Viliam Geffert Juhani Karhumäki
Alberto Bertoni Bart Preneel
Pavol Návrat Mária Bieliková (Eds.)

SOFSEM 2008: Theory and Practice of Computer Science

LNCS 4910

34th Conference on Current Trends
in Theory and Practice of Computer Science
Nový Smokovec, Slovakia, January 2008, Proceedings

Springer

Lecture Notes in Computer Science 4910

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Viliam Geffert Juhani Karhumäki
Alberto Bertoni Bart Preneel
Pavol Návrat Mária Bieliková (Eds.)

SOFSEM 2008: Theory and Practice of Computer Science

34th Conference on Current Trends
in Theory and Practice of Computer Science
Nový Smokovec, Slovakia, January 19-25, 2008
Proceedings

Springer

Volume Editors

Viliam Geffert
P. J. Šafárik University, 04154 Košice, Slovakia
E-mail: villiam.geffert@upjs.sk

Juhani Karhumäki
University of Turku, 20014 Turun Yliopisto, Finland
E-mail: karhumak@utu.fi

Alberto Bertoni
Università degli Studi, 20135 Milano, Italy
E-mail: bertoni@dsi.unimi.it

Bart Preneel
Katholieke Universiteit Leuven, 3001 Leuven-Heverlee, Belgium
E-mail: bart.preneel@esat.kuleuven.be

Pavol Návrat
Mária Bieliková
Slovak University of Technology, 81243 Bratislava, Slovakia
E-mail: {navrat,bielik}@fiit.stuba.sk

Library of Congress Control Number: 2007942547

CR Subject Classification (1998): F.2, F.1, D.2, H.3, H.2.8, H.4, F.3-4

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-77565-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-77565-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12212293 06/3180 5 4 3 2 1 0

Preface

This volume contains the invited and the contributed papers selected for presentation at SOFSEM 2008, the 34th Conference on Current Trends in Theory and Practice of Computer Science, which was held January 19–25, 2008, in the Atrium Hotel, Nový Smokovec, High Tatras in Slovakia.

SOFSEM (originally SOFtware SEMinar), as an annual international conference devoted to the theory and practice of computer science, aims to foster cooperation among professionals from academia and industry working in all areas in this field. Developing over the years from a local event to a fully international and well-established conference, contemporary SOFSEM continues to maintain the best of its original Winter School aspects, such as a high number of invited talks and in-depth coverage of novel research results in selected areas within computer science. SOFSEM 2008 was organized around the following tracks:

- Foundations of Computer Science (Chair: Juhani Karhumäki)
- Computing by Nature (Chair: Alberto Bertoni)
- Networks, Security, and Cryptography (Chair: Bart Preneel)
- Web Technologies (Chair: Pavol Návrat)

The SOFSEM 2008 Program Committee consisted of 75 international experts, representing active areas of the SOFSEM 2008 tracks with outstanding expertise and an eye for current developments, evaluating the submissions with the help of 169 additional reviewers.

An integral part of SOFSEM 2008 was the traditional Student Research Forum (chaired by Mária Bieliková), organized with the aim of presenting student projects in the theory and practice of computer science and to give students feedback on both originality of their scientific results and on their work in progress. The papers presented at the Student Research Forum were published in a separate local proceedings.

In response to the call for papers, SOFSEM 2008 received 162 submissions. After a careful reviewing process (mostly with three or four reviewers per paper), followed by a detailed electronic discussion, a total of 57 papers were selected for presentation at SOFSEM 2008, following the strictest criteria of quality and originality. In addition, this volume contains full texts or extended abstracts of 10 invited papers.

From among 44 papers falling into the student category, 13 were accepted for this volume. The Track Chairs also selected the best student paper: "Computing Longest Common Substring and all Palindromes from Compressed Strings," by W. Matsubara, S. Inenaga, A. Ishino, A. Shinohara, T. Nakamura, and K. Hashimoto. Furthermore, 13 student papers were presented at the SOFSEM 2008 Student Research Forum, based on the recommendation by the PC members and approved by the Track Chairs.

As the editors of these proceedings, we are indebted to all the contributors to the scientific program of the conference, especially to the invited speakers and all authors of the contributed papers. We also thank all authors who responded promptly to our requests for minor modifications and corrections in their manuscripts. Last but not least, we would like to thank the members of the Program Committee and all additional reviewers for their work.

November 2007

Viliam Geffert
Juhani Karhumäki
Alberto Bertoni
Bart Preneel
Pavol Návrat
Mária Bieliková

Organization

Program Committee

Foundations of Computer Science

Juhani Karhumäki	Turku, Finland, Chair
Marie-Pierre Béal	Marne la Valée, France
Wit Foryś	Kraków, Poland
Viliam Geffert	Košice, Slovakia
Lane A. Hemaspaandra	Rochester/NY, USA
Hendrik J. Hoogeboom	Leiden, The Netherlands
Oscar H. Ibarra	Santa Barbara/CA, USA
Pawel M. Idziak	Kraków, Poland
Rastislav Královič	Bratislava, Slovakia
Mojmir Křetínský	Brno, Czech Republic
Luděk Kučera	Prague, Czech Republic
Markus Lohrey	Stuttgart, Germany
Markus Nebel	Kaiserslautern, Germany
Damian Niwinski	Warszawa, Poland
Alexander Okhotin	Turku, Finland
Pekka Orponen	Helsinki, Finland
Dana Pardubská	Bratislava, Slovakia
Michel Rigo	Liège, Belgium
Sebastian Seibert	Zürich, Switzerland
Imrich Vrťo	Bratislava, Slovakia
Jiří Wiedermann	Prague, Czech Republic
Detlef Wotschke	Frankfurt a/M., Germany
Sheng Yu	London/ON, Canada
Wiesław Zielonka	Paris, France

Computing by Nature

Alberto Bertoni	Milan, Italy, Chair
Gabriela Andrejková	Košice, Slovakia
Bartlomiej Beliczynski	Warsaw, Poland
Christian Choffrut	Paris, France
Marco Dorigo	Brussels, Belgium
Rūsiņš Freivalds	Rīga, Latvia
Rudolf Freund	Vienna, Austria
Pascal Koiran	Lyon, France
Věra Kůrková	Prague, Czech Republic

Giancarlo Mauri	Milan, Italy
Carlo Mereghetti	Milan, Italy
Ashwin Nayak	Waterloo/ON, Canada
Bernardete Ribeiro	Coimbra, Portugal
John G. Taylor	London, UK
György Vaszil	Budapest, Hungary
Ingo Wegener	Dortmund, Germany

Networks, Security, and Cryptography

Bart Preneel	Leuven, Belgium, Chair
Tuomas Aura	Cambridge, UK
Michael Backes	Saarbrücken, Germany
Dan Bailey	Bedford/MA, USA
Bruno Crispo	Trento, Italy
George Danezis	Leuven, Belgium
Ian Goldberg	Waterloo/ON, Canada
Marc Joye	Cesson-Sévigné, France
Aggelos Kiayias	Storrs/CT, USA
Vlastimil Klíma	Prague, Czech Republic
Chris Mitchell	London, UK
David Naccache	Paris, France
Mats Näslund	Stockholm, Sweden
Frank Piessens	Leuven, Belgium
Ahmad Sadeghi	Bochum, Germany
Pim Tuyls	Eindhoven, The Netherlands

Web Technologies

Pavol Návrat	Bratislava, Slovakia, Chair
András Benczur	Budapest, Hungary
Mária Bieliková	Bratislava, Slovakia
Paul Brna	Glasgow, UK
Peter Brusilovsky	Pittsburgh/PA, USA
Alexandra I. Cristea	Warwick, UK
Peter Dolog	Aalborg, Denmark
Ladislav Hluchý	Bratislava, Slovakia
Geert-Jan Houben	Brussels, Belgium
Wolfgang Nejdl	Hannover, Germany
Ján Paralič	Košice, Slovakia
Dimitris Plexousakis	Heraklion, Greece
Jaroslav Pokorný	Prague, Czech Republic
Karel Richta	Prague, Czech Republic

Petr Šaloun	Ostrava, Czech Republic
Július Štuller	Prague, Czech Republic
Peter Vojtáš	Praha, Czech Republic
Bartosz Walter	Poznań, Poland
Jaroslav Zendulka	Brno, Czech Republic

List of Additional Reviewers

Stefano Aguzzoli, Gorjan Alagic, Jürgen Albert, Rajeev Alur, András Antos, Vikraman Arvind, Marian Babik, Zoltán Balogh, Jiří Barnat, Simone Bassis, Dietmar Berwanger, Hans-Joachim Böckenhauer, Frank de Boer, Bernard Boigelot, Olivier Bournez, Tomáš Brázdil, Robert Brijder, Václav Brožek, Ivana Budinská, Martin Burger, Thierry Cachat, Peter Cameron, Alexandre Campo, Christophe De Cannière, Olivier Carton, Elena Casiraghi, Jérémie Chalopin, Jan Chomicki, Robin Cockett, Elena Czeizler, Eugen Czeizler, Lieven Desmet, Dwight Deugo, Volker Diekert, Didier Dubois, Eric Duchêne, Alain Durand, Szilárd-Zsolt Fazekas, Thomas Fernique, Vojtěch Forejt, Dominik Freydenberger, Cesare Furlanello, Fabio Gaducci, Jean-Marie Gaillourdet, Leo Galamboš, Yuan Gao, Emil Gatial, Stefan Göller, Rodrigo Gonzalez, Tomasz Gorazd, Sylvain Gravier, Jarosław Grytczuk, Giovanna Guaiana, Grzegorz Gutowski, Harri Haanpää, Julien Hendrickx, Kevin Henry, Alain Hertz, Mika Hirvensalo, Jan Holeček, Antti Hyvärinen, Sebastiaan Indesteege, Damien Jamet, Petr Jančar, Ryszard Janicki, Artur Jeż, Ranjit Jhala, Galina Jirásková, Raphaël Jungers, Christos Kapoutsis, Sabine Kappes, Jarkko Kari, Petteri Kaski, Julia Kempe, Alexei Khvorost, Jetty Kleijn, Joachim Kneis, Eryk Kopczyński, Maciej Koutny, Jakub Kozik, Stanislav Krajči, Richard Královič, Matthias Krause, Danny Krizanc, Sven Krumke, Alexander Kulikov, Michal Kunc, Dietrich Kuske, Martin Kutrib, Michal Laclavik, Jeroen Laros, Frédéric Lefèbfre, Pierre Lescanne, Hing Leung, Hans Löhr, Tamás Lukovszki, Andreas Malcher, Dario Malchiodi, Adam Malinowski, Sebastian Maneth, Wim Martens, Pavel Martyugin, Ayoub Massoudi, Grzegorz Matecki, András Méhes, Larissa Meinicke, Alexandre Miquel, Tobias Mömke, Filip Murlak, Uwe Nestman, Frank Neven, Gregory Neven, Linh Anh Nguyen, Samuel Nicolay, Rolf Niedermeier, Jan Obdržálek, Marco Montes de Oca, Peter Ochsenschläger, Elizaveta A. Okol'nishnikova, Alina Oprea, Krzysztof Pancerz, Geneviève Paquin, Panos Pardalos, Radek Pelánek, Wojciech Penczek, Milan Petković, Zoltán Petres, Justus Piater, Lecomte Pierre, Jean-Éric Pin, Tomáš Plachetka, Wojciech Plandowski, Igor Potapov, Vladimir Prus, Sanguthevar Rajasekaran, Bala Ravikumar, Heiko Röglin, Andrea Roli, Dan Romik, Miklos Santha, Martin Sauerhoff, Elisa Schaeffer, Karsten Schmidt, Georg Schnitger, André Schumacher, Martin Seleng, David Shmoys, Detlef Sieling, Marek Skomorowski, Boris Skorić, Sergey Sosnovsky, Fred Spiessens, Zdenko Staníček, Marcin Stefaniak, Patrick Stewin, Jan Strejček, Jianwen Su, Alexander Szabari, Nathalie Sznajder, Paul Tarau, Torsten Tholey, Marco Trubian, Yih-Kuen Tsay, Paweł Waszkiewicz, Frank Weinberg, Philipp Wölfel, Hsu-Chun Yen, Koen Yskout, Hong-Sheng Zhou.

Steering Committee

Július Štuller	Institute of Computer Science, Czech Republic, Chair
Mária Bieliková	Slovak University of Technology, Slovakia
Bernadette Charron-Bost	École Polytechnique, France
Keith Jeffery	CCLRC, UK
Antonín Kučera	Masaryk University Brno, Czech Republic
Jan van Leeuwen	Utrecht University, The Netherlands
Branislav Rovan	Comenius University in Bratislava, Slovakia
Petr Tůma	Charles University in Prague, Czech Republic

Organizing Institutions

Institute of Computer Science, P. J. Šafárik University, Košice, Slovakia
Slovak Society for Computer Science
Institute of Computer Science of the Czech Academy of Sciences
Czech Society for Cybernetics and Informatics

Organizing Committee

Gabriela Andrejková, Jozef Gajdoš, František Galčík, Viliam Geffert, Peter Gurský, Tomáš Horváth, Michal Mati, Peter Mlynárčik, Marián Novotný, Dana Pardubská, Gabriel Semanišin (Chair), Eva Trenklerová

Sponsoring Institutions

Asseco Slovakia
Ditec — Data Information Technology & Expert Consulting
ERCIM — European Research Consortium for Informatics and Mathematics
Hewlett-Packard Slovakia
IBM Slovakia
Ness Slovakia
Siemens Slovakia
SOFTEC

Table of Contents

Invited Talks

Foundations of Computer Science

Computing by Nature

Networks, Security, and Cryptography

Web Technologies

Quantum Random Walks – New Method for Designing Quantum Algorithms

Andris Ambainis*

Department of Computer Science, University of Latvia, Raina bulv. 19, Riga, LV-1586, Latvia
andris.ambainis@lu.lv

Abstract. Quantum walks are quantum counterparts of random walks. In the last 5 years, they have become one of main methods of designing quantum algorithms. Quantum walk based algorithms include element distinctness, spatial search, quantum speedup of Markov chains, evaluation of Boolean formulas and search on "glued trees" graph. In this talk, I will describe the quantum walk method for designing search algorithms and show several of its applications.

1 Quantum Algorithms: An Overview

Quantum computing (and, more broadly, quantum information science) is a new area at the boundary of computer science and physics. The laws of quantum mechanics are profoundly different from conventional physics. Quantum computing studies how to use them for the purposes of computer science and information processing.

The area of quantum computing was shaped by the discoveries of two major quantum algorithms in mid-1990s. The first of the two was Shor's polynomial time quantum algorithm for factoring and discrete logarithms.

Factoring and discrete logarithm are very hard number theoretic problems. The difficulty of these problems has been used to design cryptosystems (such as RSA and Diffie-Helman key exchange) for secure data transmission over an insecure network (such as Internet). The security of data transmission rests on the assumption that it is hard to factor (or find discrete logarithm of) large numbers. Until recently, this assumption was not in doubt. Mathematicians had tried to devise an efficient way of factoring large numbers for centuries, with no success.

In 1994, Shor [26] discovered a fast algorithm for factoring large numbers - on a quantum mechanical computer. This shook up the foundations of cryptography. If a quantum mechanical computer is built, today's methods for secure data transmission over the Internet will become insecure.

Another, equally strikingly discovery was made in 1996, by Lov Grover [18]. He invented a quantum algorithm for speeding up exhaustive search problems. Grover's algorithm solves a generic exhaustive search problem with N possible solutions in time $O(\sqrt{N})$. This provides a quadratic speedup for a range of search problems, from ones that are solvable in polynomial time classically to NP-complete ones.

* Supported by University of Latvia Grant Y2-ZP01-100.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 1–4, 2008.

Since then, each of the two algorithms has been analyzed in great detail. Shor's algorithm has been generalized to solve a class of algebraic problems that can be abstracted to [illegible] [20]. Besides factoring and discrete logarithm, the instances of Abelian HSP include cryptanalysis of hidden linear equations [8], solving Pell's equation, principal ideal problem [19] and others.

Grover's algorithm has been generalized to the framework of [illegible] [10] and extended to solve problems like approximate counting [12,25] and collision-finding [11].

More recently, two new methods for designing quantum algorithms have emerged: quantum walks [2,21] and adiabatic algorithms [15].

2 Quantum Walks

The main quantum algorithms that use quantum walks are:

1. Quantum walk on the "glued trees" graph [13].
   Consider a problem in which we have a graph G with two particular vertices u, v, designed as the entrance and the exit. The problem is to find the vertex v, if we start at the vertex u.
   There is a special exponential size graph called "glued trees" on which any classical algorithm needs exponential time to find v but a quantum algorithm can find v in polynomial time [13].
2. An optimal quantum algorithm for element distinctness [3].
   In the element distinctness problem, we are given variables $x_1, \ldots, x_N \in \{1, \ldots, M\}$. Our task is to determine if there are two equal ones among them. The values x_i are given via a black box which answers queries. An input to a query is i, the output is x_i. Classically (i.e., for non-quantum algorithms), $\Omega(N)$ queries are required. In quantum case, we can use quantum walks to design a quantum algorithm that solves element distinctness with $O(N^{2/3})$ queries. This is known to be optimal [1,4].
3. A quantum algorithm for search on grids [6,14].
   In the spatial search problem, we have N items arranged on a d-dimensional grid. The task is to find an item with a certain property (a "marked" item). In one time step, we can perform one of two operations:
   - check if the item at the current location is marked;
   - move to an adjacent location on the grid.

   This problem is harder than Grover's search problem, since we are only permitted to move to an adjacent location in one step (instead of being able to move to an arbitrary location).
   Classical complexity of the problem is $\Omega(N)$. In the quantum case, the problem can be solved either by a discrete time quantum walk [6] or a continuous time quantum walk [14]. Both of those approaches give a quantum algorithm that runs in time $O(\sqrt{N})$ for $d \geq 3$ and time $O(\sqrt{N} \log^2 N)$ for $d = 2$.
4. A quadratic quantum speedup for hitting times of Markov chains [27,23].
   Let M be a reversible and irreducible classical Markov chain in which some states are marked. Let T be the expected time to reach a marked state, if we

start the Markov chain M in a uniformly random state. Then, a quantum algorithm of [27,23] can find a marked state in time (roughly) $O(\sqrt{T})$.

Quantum algorithms for both element distinctness problem and search on the grid are special cases of this algorithm.

5. A quantum algorithm for finding triangles in a graph [24,23].
In this problem, we have a graph on n vertices, specified by n^2 variables $a_{i,j}$ that are 1 if there is an edge between vertices i and j. Classically, $\Omega(n^2)$ queries are required. Using the ideas from the element distinctness algorithm, one can design a $O(n^{1.3}\log^c n)$ query quantum algorithm [24]. This has been refined to $O(n^{1.3})$ query quantum algorithm by [23].
6. A quantum algorithm for testing matrix identities [9,22].
We have three $n \times n$ matrices A, B and C and are allowed to query the matrix elements. We would like to test if $AB = C$. The classical complexity of the problem is $\Theta(n^2)$. (An $\Omega(n^2)$ query lower bound is easy. Querying all elements allows to solve the problem with $3n^2$ queries. There is also a probabilistic algorithm whose total running time is $O(n^2)$ [17].)

Using quantum Markov chain results of [27], Buhrman and Špalek [9] have constructed a $O(n^{5/3})$ query quantum algorithm for this problem. Magniez and Nayak [22] have shown how to use quantum walks to test if a group operation (specified by a table which is accessed by queries) is commutative.
7. A quantum algorithm for evaluating Boolean formulas [16,5].
Using quantum walks, [16,5] have shown that any Boolean formula (consisting of AND, OR, NOT gates) of size m can be evaluated using $O(m^{1/2+o(1)})$ queries to the variables in the formula.

This is nearly optimal: if the formula is read-once (every variable occurs exactly once), $\Omega(m)$ queries are required.

In this talk, we will describe the quantum walk methodology and some of the results listed above.

For a broader (although slightly outdated) survey on quantum walks, we refer the reader to [2,21].

References

1. Aaronson, S., Shi, Y.: Quantum lower bounds for the collision and the element distinctness problems. Journal of ACM 51(4), 595–605 (2004)
2. Ambainis, A.: Quantum walks and their algorithmic applications. International Journal of Quantum Information 1, 507–518 (2003)
3. Ambainis, A.: Quantum walk algorithm for element distinctness. In: Proceedings of FOCS 2004, pp. 22–31 (2004)
4. Ambainis, A.: Polynomial degree and lower bounds in quantum complexity: collision and element distinctness with small range. Theory of Computing 1, 37–46 (2005)
5. Ambainis, A., Childs, A., Reichardt, B., Spalek, R., Zhang, S.: Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. In: Proceedings of FOCS 2007, pp. 363–372 (2007)
6. Ambainis, A., Kempe, J., Rivosh, A.: Coins make quantum walks faster. In: Proceedings of SODA 2005, pp. 1099–1108 (2005)

7. Barnum, H., Saks, M.: A lower bound on the quantum complexity of read once functions. Journal of Computer and System Sciences 69, 244–258 (2004)
8. Boneh, D., Lipton, R.: Quantum cryptanalysis of hidden linear functions (extended abstract). In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 424–437. Springer, Heidelberg (1995)
9. Buhrman, H., Špalek, R.: Quantum verification of matrix products. In: Proceedings of SODA 2006, pp. 880–889 (2006)
10. Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. In: Quantum Computation and Quantum Information Science. AMS Contemporary Mathematics Series, vol. 305, pp. 53–74 (2002)
11. Brassard, G., Hoyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 163–169. Springer, Heidelberg (1998)
12. Brassard, G., Høyer, P., Tapp, A.: Quantum counting. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 820–831. Springer, Heidelberg (1998)
13. Childs, A.M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., Spielman, D.A.: Exponential algorithmic speedup by quantum walk. In: Proceedings of STOC 2003, pp. 59–68 (2003)
14. Childs, A.M., Goldstone, J.: Spatial search and the Dirac equation. Physical Review A 70, 42312 (2004)
15. Farhi, E., Goldstone, J., Gutman, S., Sipser, M.: A quantum adiabatic algorithm applied to random instances of an NP-complete problem. Science 292, 472–476 (2001)
16. Farhi, E., Goldstone, J., Gutman, S.: A Quantum Algorithm for the Hamiltonian NAND Tree. arXiv preprint quant-ph/0702144
17. Freivalds, R.: Fast probabilistic algorithms. In: Becvar, J. (ed.) MFCS 1979. LNCS, vol. 74, pp. 57–69. Springer, Heidelberg (1979)
18. Grover, L.: A fast quantum mechanical algorithm for database search. In: Grover, L. (ed.) Proceedings of STOC 1996, pp. 212–219 (1996)
19. Hallgren, S.: Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. Journal of the ACM 54, 1 (2007)
20. Jozsa, R.: Quantum factoring, discrete logarithms, and the hidden subgroup problem. IEEE Multimedia 3, 34–43 (1996)
21. Kempe, J.: Quantum random walks - an introductory overview. Contemporary Physics 44(4), 307–327 (2003)
22. Magniez, F., Nayak, A.: Quantum complexity of testing group commutativity. Algorithmica 48(3), 221–232 (2007)
23. Magniez, F., Nayak, A., Roland, J., Santha, M.: Search via quantum walk. In: Proceedings of STOC 2007, pp. 575–584 (2007)
24. Magniez, F., Santha, M., Szegedy, M.: Quantum algorithms for the triangle problem. SIAM Journal on Computing 37(2), 413–424 (2007)
25. Nayak, A., Wu, F.: The quantum query complexity of approximating the median and related statistics. In: Proceedings of STOC 1999, pp. 384–393 (1999)
26. Shor, P.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: Proceedings of FOCS 1994, pp. 124–134 (1994)
27. Szegedy, M.: Quantum speed-up of Markov chain based algorithms. In: Proceedings of FOCS 2004, pp. 32–41 (2004)

Social Information Access: The Other Side of the Social Web

Peter Brusilovsky

School of Information Sciences, University of Pittsburgh
135 N. Bellefield Ave. Pittsburgh, PA 15260, USA
peterb@pitt.edu

Abstract. Modern Web, which is frequently called Social Web or Web 2.0, celebrates the power of the user community. Most frequently it is associated with the power of users as contributors or various kinds of contents through Wikis, blogs, and resource sharing sites. However, the community power impacts not only the production of Web content, but also the access to all kinds of Web content. A number of research groups worldwide work on social information access techniques, which help users get to the right information using "community wisdom" distilled from tracked actions of those who worked with this information earlier. The paper provides an overview of this research stream focusing on social search, social navigation, and social visualization techniques.

1 Introduction

Modern Web, which is frequently called Social Web or Web 2.0, celebrates the power of the user community. Most frequently it is associated with the power of user community as content contributors through Wikis, blogs, and resource sharing sites. However, the power of community impacts not only the production of Web content, but also the access to all kinds of Web content. A number or research groups worldwide work on *social information access* technologies, which help users get to the right information using "community wisdom" distilled from the actions of those who worked with this information earlier. Social information access technologies capitalize on the natural tendency of people to follow direct and indirect cues of others' activities, e.g. going to a restaurant that seems to attract many customers, or asking others what movies to watch.

Social information access can be formally defined as a stream of research that explores methods for organizing users' past interaction with an information system (known as explicit and implicit *feedback*), in order to provide better access to information to the future users of the system. This stream has to be considered as emerging. It covers a range of rather different systems and technologies operating on a different scale - from a small closed corpus site to the whole Web. While the technologies located on the different sides of this stream may not even recognize each other as being a part of the same whole, the whole stream is driven by the same goals: to use the power of a user community for improving information access.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 5–22, 2008.

The social information access is frequently considered as an alternative to the traditional (content-based) information access. In most of the cases, social information access can run in parallel with the traditional one, helping users to find resources that would be hard to find in a traditional way. In other cases where traditional information access is hard to organize (for example, in a collection of non-indexed images), social mechanisms (such as tagging) can provide a handy replacement. However, it has been more and more frequently demonstrated that most benefits could be obtained by integrating social and traditional technologies, for example, building hybrid recommender systems, which integrate collaborative and content-based recommender mechanisms [10]. An important feature of all social navigation systems is self-organization. Social navigation systems are able to work with little or no involvement of human indexers, organizers, or other kinds of experts. They are truly powered by a community of users.

As a type of information access, which can offer multiple benefits, while being relatively easy to organize and maintain, social information access attracts more and more researchers and practitioners. The paper provides an overview of this emerging research stream while focusing primary on social search, social navigation, and social visualization techniques.

2 The Emergence of Social Information Access

The pioneer work on social information access appeared in the early 1990s, but the emergence of social information access as a research area happened between 1994 and 1996. This period brought many innovations in all areas related to information access, as research teams investigated new approaches to help users in the rapidly expanding information space. In the area of social information access, these years produced two well-defined research streams: collaborative filtering and social navigation.

Collaborative filtering [36; 45] attempted to propagate information items between users with similar interests. This technology enabled social forms of *information filtering and recommendation*. The emergence of collaborative filtering is typically traced back to the *Information Tapestry* project and a seminal paper about it, which coined the term [23]. Information Tapestry employed an approach, which was later called *pull-active* collaborative filtering [36]: to receive social guidance, its users had to actively query the community feedback left by earlier users. Another early example of "active" filtering (in this case *push-active*) was provided by Lotus Notes recommender [39], which encouraged users to send interesting documents directly to their colleagues. While both of these examples were highly influential, the mainstream work in the area of collaborating filtering focused on *automatic* approaches based on matching users with similar interests and cross-recommending positively-rated items. This direction was lead by such pioneer systems as GroupLens [43], Ringo [47], and Video Recommender [26].

Social navigation [13] in its early forms attempted to visualize the aggregated or individual actions of other community users. The motivation behind this work was that that these "footprints" can help other community users to navigate through information space. By its nature, social navigation supported user browsing activity.

The ideas of social navigation are frequently traced back to the pioneer *Read Wear and Edit Wear* system [27]. This system visualized the history of authors' and readers' interactions with a document enabling new users to quickly locate the most viewed or edited parts of the document. Social navigation in information space as well as the term *social navigation* was introduced two years later by Dourish and Chalmers as "moving towards cluster of people" or "selecting subjects because others have examined them" [16].

The pioneer systems Juggler [12] and Footprints [50] used the ideas of social navigation to help users navigating in two kinds of information spaces – a Web site and a text-based virtual environment (MOO). Both systems attempted to visualize "wear" traces left by the system users in order to guide future users. In addition to this *indirect* social navigation, Juggler also implemented several types of direct social navigation (for example, allowing users to guide each other directly through chat). This allowed Dieberger [12] to start the process of generalizing the ideas of social navigation.

Further generalization of the field of social navigation was propelled by several workshops, which gathered like-minded researchers, and publications, which steamed from these workshops [14; 28; 41]. As a result of this active idea exchange, the scope of social navigation was broadened to cover all approaches, which use social feedback as a source of knowledge in assisting the users. The notion of social feedback was also broadened to includes a variety of options – from navigation traces to rich explicit feedback and resource annotations. The newly-defined field of social navigation included two main groups of systems - collaborative filtering systems and Footprints-type systems, which were referred to as history-enriched environments.

More recently, the set of social information access technologies was extended with *social search* and *social bookmarking* systems. The research on collaborative information retrieval [33] produced some truly social search systems that attempted to help new searchers by capitalizing on past successful searches of similar users. The pioneer AntWorld system [32] was soon followed by several similar systems such as I-SPY [48] and SERF [30]. Another group of pioneer systems such as Siteseer [44], PowerBookmarks [38], and WebTagger [34] started the collaborative bookmarking research stream. Researchers working on collaborative bookmarking explored different ways of bookmark sharing to help new users locate useful information already discovered and classified by others and invented social tagging mechanism. In less than 10 years social bookmarking and tagging system popularized by such systems as *del.icio.us* or *Flickr.com* grew into a new major Internet technology [24].

It's important to stress again that each social information access technology achieved success by collecting community wisdom in a specific form, and enabling users working with a specific information access paradigm to benefit from this. Being quite different by nature, the social information access technologies have a lot in common. Unfortunately, no recent attempts were made to provide a comprehensive overview of these technologies. The important "unifying" books and papers on social navigation [14; 28; 41] appeared too early to cover and integrate more recent technologies. This paper attempts to fill this gap to some extent and provide a unified view on all technologies listed above. It looks at the problem from the prospective of information access, which is a slightly different and in some sense more narrow angle than the one taken in the books cited above. The next section suggests a framework,

which can be used to classify existing social information access technologies. The remaining sections focus on social search, browsing, and visualization, which are arguably the least known social information access technologies.

3 A Taxonomy of Social Information Access Technologies

To understand differences between modern social information access systems, it is useful to understand which type of information access they are attempting to support. In an earlier paper we distinguished four major *information access* paradigms to classify adaptive information access systems: ad-hoc information retrieval, information filtering, hypertext browsing, and information visualization [9]. In *ad-hoc information retrieval* (IR), users get access to relevant information by issuing a query to an IR system or search engine and analyzing a ranked list of documents (for example, book records), which are returned as a result. In *information filtering* (IF) an information system attempts to recommend documents, which match the user's long-term interests. Traditional IF systems match a user-provided profile against a flow of incoming documents (for example, news articles) to select the most relevant items for the user. Modern recommender systems (often considered as an extension of IF) construct dynamic user profiles by observing user interactions, and as a result can produce new recommendations even in stable document collections. In *hypertext browsing*, a user attempts to find relevant documents by browsing links that connect documents in a collection. In *information visualization*, a set of documents is presented to the user using some visualization metaphor in 2 or 3 dimensions; the user observes or, in the case of interactive visualization, interacts with the visualized set to find the most relevant documents.

The analysis of modern social information access technologies shows that different technologies were, in fact, developed in conjunction with different information access paradigms. The type of information access supported by a specific technology to a large extent determines the nature of this technology and its difference from other paradigms. For example, classic social navigation technologies (history-enriched environments) were developed to support browsing-based access. This context requires navigation support systems, which can help the users to decide, which of many links on the current page to follow. The natural approach to using the community wisdom is to show "where did the people go" [13] by augmenting links with digital "wear" indicators. The natural approach to collect this wisdom is to track user page visits [8] or link traversals [50].

Social search technologies were developed to support traditional IR information access. In this context, users expect to see a ranked list of relevant resources. The natural approach to using the community wisdom is to insert community-relevant links into the list or results [30; 48] or stress, which of the returned documents are not only relevant, but also appreciated by the community [2; 32]. A reliable approach to collecting this wisdom is to track connections between queries and items selected or rated by the community members in the context of these queries [30; 32; 48].

The presence of the context (such as current query or location in the hyperspace) helps both IR and browsing systems to identify similar users without more sophisticated tracking. These system can accumulate community wisdom by query,

by link, or by page – but not necessary by user. As a result, there is no need for a user to login, although both kinds of systems can provide better service if the user can pick up one of the sub-communities tracked by the system [19; 48]. As a result, social search and browsing system are very easy to integrate into any kind of existing Web systems.

In contrast, collaborative filtering technologies were designed to work in the most challenging situation – long-term information filtering, i.e., in a situation with no current context, As a result, collaborative filtering technologies have to track all user activity on the individual level, construct detailed user profiles and apply sophisticated approaches to match similar users.

Let's skip social visualization, since this area is not sufficiently developed, and move to social bookmaking. Social bookmarking technology presents an interesting case in social information access. Unlike other listed technologies, modern social bookmarking does not really augment any of the traditional information access paradigms. Instead, it provides an alternative mechanism to access information using community-contributed tags. In this case the social wisdom, which other social systems accumulate in some hidden form "behind the stage", becomes visible as a tightly interlinking *tag space*. The information can be accessed through this tag space using traditional access paradigms such as tag searching, tag navigation, or tag visualization in the form of *tag cloud*. So, social tagging augments several traditional information access paradigms by providing additional community-created space where these paradigms can be applied.

As could be noticed from the discussion above, there are at least three different levels on which social information technologies accumulate user information. The most relaxing is the *all-users* level where a system does not distinguish its users and accumulates all past usage activity "in a single pile". This approach was used in several early social search and browsing systems [12; 21; 32; 50] and is still appropriate in the situations where the body of users is reasonably sized and their information needs are similar [30; 37].

In the situation where the body of users is large and diverse (which is the case for the majority of "open Web" systems), the presence of context such as the current query or Web page become insufficient to reliably identify similar users. The information needs of the users passing a specific link or issuing a specific query may still be too diverse. This situation caused several recent social information access projects such as I-SPY [48], Knowledge Sea II [17], Conference Navigator [19], and ASSIST [20] to start accumulated social wisdom for different subsets of the whole body of users independently. The subsets could be of different nature and size – from a *community* of like-minded users [19; 48], which can include hundreds, to a small *group* of users (such as a college class) joined by the same information goal [17]. This approach can be called *community-level* or *group-level* information access. In this paper we will use the latter term, since it stresses the clear difference from "all-users" approach. Group level access provides an attractive compromise between all-users and individual level. It can provide very reliable social guidance without a requirement to authenticate, which can be a stumbling point for several reasons [48]. Comparable with all-user level, group-level approaches sort user feedback in multiple "group bins", which may result in community wisdom becoming too sparse. The challenge for the developers of group-level social systems is to engineer the groups of

the proper size to make sure that the volume of social feedback is sufficient for useful guidance. If the volume of social information provided by an average user is large and the volume of resources used by a group is not large, some good social guidance can be provided even for small groups of users. In addition, the problem of sparsity in group-level systems can be addressed by propagation of social feedback between groups [22].

The most fine-grained level of wisdom collection is *user-level*, where each piece of feedback is associated with an individual user and accumulated in the profile of this user. User-level tracking is a standard for *personalized information access* techniques such as adaptive navigation support [7] or personalized search [40]. Among social information access systems, user-level tracking is critical only for collaborative filtering systems. However, modern approaches to increasing user contributions in social systems such as incentives [11] or "do it for yourself" [18; 19] may require user-level tracking in any kind of social information access systems. User-level tracking adds another burden to the developers of a social information system: how to attribute all kinds of feedback left by an individual user to the profile of that user. To some limited extent, it can be done by tracking a user within a single session, but in general case it requires a long-term user profiling and reliable user authentication (such as password-protected login).

Due to the lack of space, this paper offers no further attempts to classify social information access systems. Instead, we refer to several collaborative filtering and social navigation papers [13; 36; 45; 49], which offer several useful dimensions to classify social systems. While each of these dimensions was suggested to classify a special group of systems, most of them can be successfully applied to classify all kinds of social information access approaches.

The remaining sections attempt to provide examples of social navigation technologies discussed above. Since both collaborative filtering and social tagging are rather well publicized in research literature, the presentation is focused on the least known social navigation, social search, and social visualization technologies.

4 Social Browsing

Social browsing systems use "community wisdom" to assist their users in the process of browsing a hyperspace or another virtual environment (such as MUD). As mentioned before, social browsing support systems collect the community wisdom by tracking two kinds of information – link traversals (link-centric approach) and page visits (page-centric). While these kinds of information look similar, they are quite different and each has its own advantage. Link traversal is the most browsing-specific kind of information. It allows not just counting how many users visited a specific page, but also distinguishing where they came from, i.e. the context in which this visit took place. The context in hypertext is quite important. Visiting the same page may be very relevant for the community members in one context (i.e., from one page linked to it) and much less relevant in the other context (i.e., from another linked page). While link-centric approaches take this context into account, they still do not track user activities within a page and this can't distinguish links to really useful pages from "tar pits" – low-value pages hidden behind attractive links. In contrast,

page-centric approaches do not distinguish how the community users get to a specific page. It decreases the precision of system advice and makes is necessary to group users in reasonably homogeneous communities and track navigation within each community. At the same time, it is easier for page centric approaches to take into account user behavior within each page (time spent, browsing, annotation) and thus distinguish good pages from tar pits.

Most of the early social browsing systems were link centered, although they still differ in respect to how the accumulated link traversal information was used. One kind of systems followed the traditions of *intelligent hypertext* and used link traversal information to periodically modify and expand the hyperspace link structure. For example, a system presented in [5] can add a link from page A to page C if existing links from A to B and from B to C were frequently used in succession. This kind of re-structuring typically requires global log analysis and is better performed *off-line*. The other kind of systems was inspired by the social navigation ideas and used link traversal information to dynamically generate history-enriched environments, where the behavior of past users is made visible.

A classic example of a link-centric history-enriched system is Juggler [12]. Juggler is a MOO system, a text-based virtual environment, which is conceptually similar to hypertext. A MOO system consists of rooms connected by passages. Every room exit in Juggler tracked how frequently it was used and showed this information (textually) as wear on a door mat. Wear decayed over time in order to reflect the behavior of natural environments, such as a path in the forest, which may fade and disappear if not used frequently. This feature was introduced to guide people towards popular locations and make encounters in the environment more likely. The same idea was implemented in the Web navigation context in Footprints system, which visualizes usage paths in a web site [50]. The Footprints system allows users to leave activity traces in the virtual environment and visualizes these traces to guide future users. With the Footprints system, new users can see the popularity of each link on the current page and make navigation decisions.

An example of a simple page-centric social browsing system is CoWeb [13; 15]. CoWeb is a history-enriched Wiki system. To increase user awareness of what is going on in the Wiki space and to guide the users to most recently updated or visited pages all links inside the CoWeb were annotated with activity markers (Fig. 1). An *access marker* showed access information using a metaphor of *footprints*. Small footprint symbols in three different colors (gray, orange, red) were placed right next to links to indicate the amount of traffic the page behind that link received in the past 24 hours. A *novelty marker* indicated another kind of community activity, which is specific to Wiki: page updates. Using three different novelty levels, it indicated how long ago that page was last modified.

A more sophisticated example of page-centric social browsing system is the Knowledge Sea II [8]. Knowledge Sea II uses ideas of social navigation to support both browsing and visualization access to information. The visualization-based access is provided through an 8 by 8 cell-based map of the information space. This map is assembled using Kohonen's Self-Organized Map (SOM) technology [35] from about 25,000 Web pages devoted to C programming language. Every cell on a resulting map provides access to a subset of these pages. By clicking on a cell, the user can open it and get access to the set of pages located in this cell (Fig. 2). An interesting property

of SOM technology is that it places similar pages into the same or adjacent cells on the map, so the result presents a reasonably good semantic map of the information space. The cells of the map are marked by keywords, which are most frequently found in its pages and by landmark resources located in the cell.

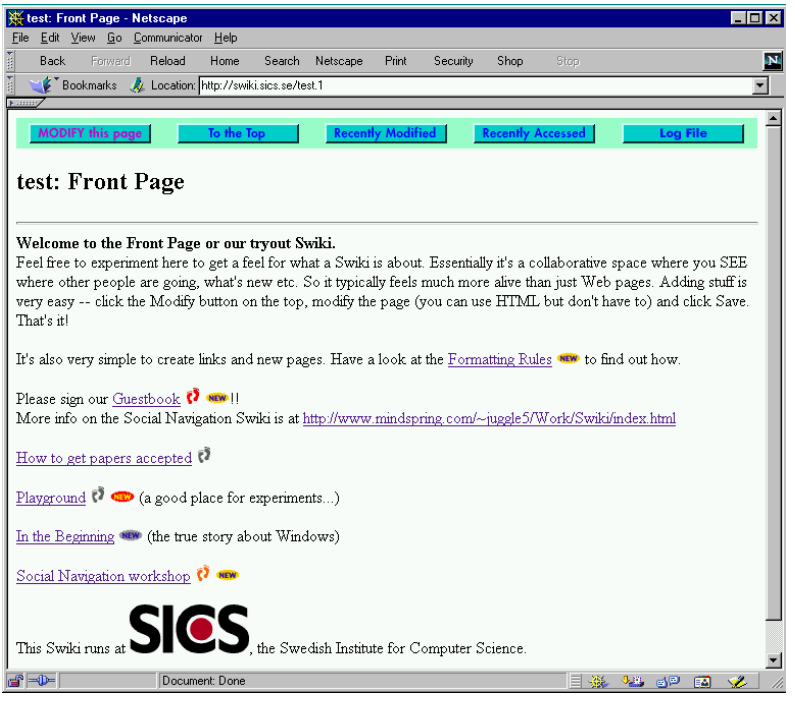


Fig. 1. Page-centric social browsing support in CoWeb. Two kinds of activity markers indicate when the page behind the link was last modified and also whether it was recently accessed. Used from [13] with the permission from the author.

The browsing-based access is provided through the hierarchical structure of the C programming tutorials assembled by the system. Each tutorial site is organized as a tree with table of contents, sections, and subsections. The home page of Knowledge Sea II provides access to the root pages of all these tutorials. Starting from that, users can navigate down to the sections or subsections of interest.

The community wisdom in Knowledge Sea II is collected by tracking two kinds of page-centric user information: timed page visits (traffic) and page annotations. This information is used to generate a history-enriched environment with two types of visual cues, which change the appearance of links on the pages and map cells presented to the user (Fig. 2). These cues are based on the two kinds of tracked information and are known respectively as traffic- and annotation-based social navigation support. The system generates appropriate cues individually for each user by analyzing past individual activities of the user and other users belonging to the same group.

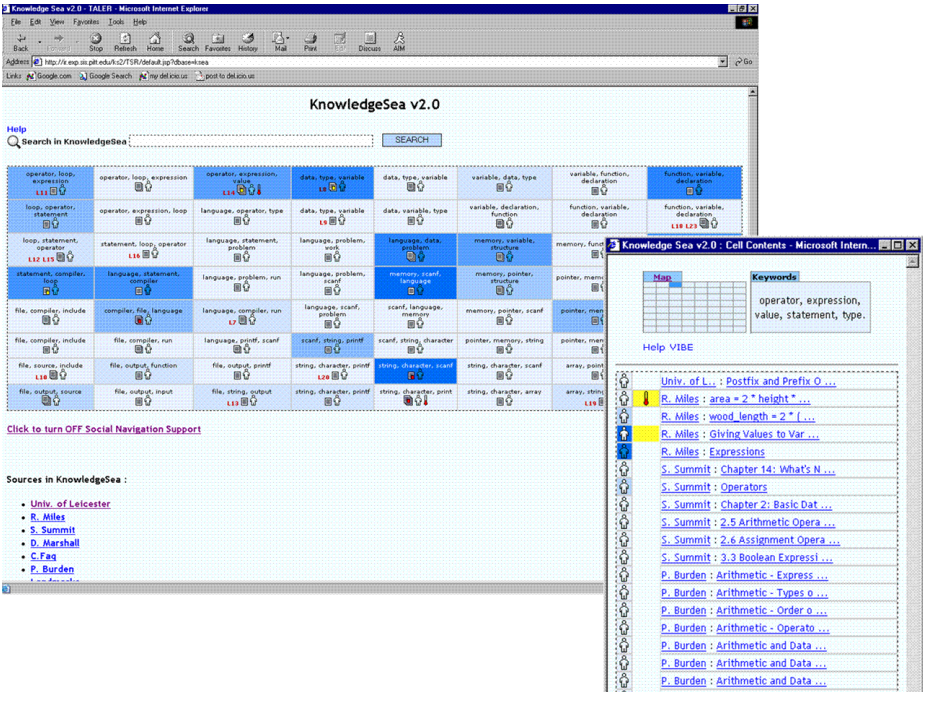


Fig. 2. Social navigation support in the Knowledge Sea II system. The knowledge map is shown on the top left and an opened cell on the right. The list of links to the tutorial roots is shown on the bottom left. A darker blue background indicates documents and map cells that have received more attention from users within the same group. Human icons with darker colors indicate documents and cells that have received more attention from the user herself. Similarly, a yellow background indicates density of annotations.

Traffic-based navigation support attempts to express how much attention the user herself and other users from the same group paid to each of 25,000 pages that the system monitors. The level of attention for a page is computed taking into account both number of visits and time spent on the page and is displayed to the user through an icon that shows a human figure on a blue background. The color saturation of the figure expresses the level of the user's own attention while the background color expresses the average level of group attention. The higher the level of attention is, the darker the color appears to the user. The contrast between colors allows the user to compare her navigation history with the navigation of the entire group. For example, a light figure on a dark background indicates a page that is popular among group members but remains under-explored by the user. The color of the map cell and the human figure shown in the cell is computed by integrating attention parameters of all pages belonging to that cell.

Annotation-based navigation support uses a similar approach to represent the number of page annotations made by the users from the same group. Users can annotate each page in the system. Users can also indicate that a note is praise (i.e., the page is good in some aspect). While users make annotations mainly for themselves,

Knowledge Sea II allows all users of the same group to benefit from collective annotation behavior. The yellow annotation icon shown next to the blue traffic icon shows the density and the "praise temperature" of annotation for each page. The more annotations a page has, the darker the yellow background color appears to the user. The temperature shown on a thermometer icon indicates the percentage of praise annotations.

5 Social Search

Let's start with narrowing down the notion of social search. Over the last 10 years, researchers and practitioners suggested a number of creative approaches to use social information for the improvement of Web search. The most well-known example is Google PageRank [6], which improves ranking by using Web link structure - a communal product of Web page authors. However, in the light of definition of social information access suggested in the introduction, only a subset of these approaches qualify as social search – those based on taking into account the past behavior of information system *users*. Similar to the case of social browsing, the approaches based on past search behavior can be further classified into *off-line* and *on-line* approaches. Off-line approaches seek to use various kinds of social information to improve indexing and ranking delivered by search engines. For example, weights of query terms may be increased in the index of documents, which were selected in response to this query [46]. On-line approaches seek to assist users in their search dynamically by accessing the accumulated social information during the very search process. By *social search*, modern sources traditionally mean on-line approaches. We will follow this tradition below.

Social search approaches do not attempt to modify the behavior of search engines, but instead apply the community wisdom before the search engine is invoked or after the results are returned. Pre-search approaches use community wisdom for social query expansion. This idea was suggested by Fitzpatrick and Dent [21] (who also coined the term *social search*) and expanded in several other projects [3; 29]. Due to the lack of space, we will not provide more details about these approaches.

Post-search approaches use the community wisdom to manipulate results returned by a search engine. It can be done in several ways: re-ranking results returned by the engine according to their social value, inserting additional results that were not in the list originally, and adding social visual cues to the listed results. A pioneer attempt to use social information for ranking search results was made at the second part of the 1990's by DirectHit (www.directhit.com) search engine. DirectHit used query logs to measure the popularity of result selections for each given query. This data was used for "social ranking" for future occurrences of this query. Unfortunately this approach turned out to bee too simplistic for a large-scale search engine: it never became a success story. The study [4] showed that DirectHit falls below the satisfaction of an average user. The most cited reason of DirectHit failure was low query repetition, which made the social data collected by it too sparse to use frequently and reliably. User diversity was another likely contribution: users with different goals and interests may prefer different results returned by the same query. Finally, the proposed

approach to link ranking was too easy to abuse by malicious users who wanted to promote their favorite pages.

A more sophisticated approach was pioneered at the same time [31] in AntWorld system [32] and later re-used in SERF [30]. AntWorld introduced the concept of a *quest*, which is an information goal pursued by a user over a sequence of queries (Fig. 3). The system successfully encouraged its users to describe their quests in natural language and used this description to determine inter-quest similarity. During their search, the users were able to rank search results by their relevance to the original quest (not a query used to obtain this result!). These innovations allowed the system to address to some extent the sparsity and reliability problems. To determine documents, which are socially relevant for a particular quest, the system looked for positively ranked documents in past similar quests. The system assisted the user by adding socially relevant documents to the list of search results and also adding a small *ant* icon to socially relevant links returned during each search within the quest.

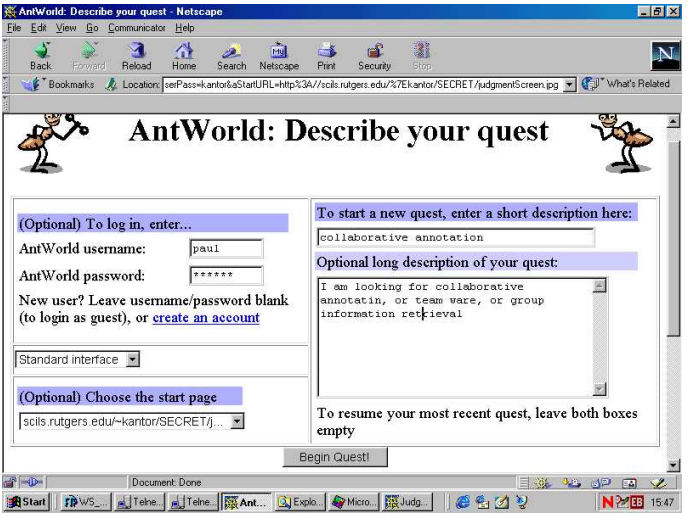


Fig. 3. AntWorld encouraged its user to describe their quests in natural language

A different approach to improve the reliability of social recommendation was suggested in I-SPY, which worked as a post-filter for regular search engines [48]. I-SPY combined and extended DirectHit and AntWord approaches. Like DirectHit, it used indirect feedback (the fact of document selection from the list of results) and accumulated social information on the level of a single query, not a multy-query quest. Thus it targeted typical search engine users who are less likely to specify wordy quests or rate search results. At the same time, I-SPY used query similarity to fight sparsity in the same way as AntWorld used quest similarity. When presented with a new query, in addition to retrieving the appropriate results from the underlying search engine, I-SPY retrieved any search sessions associated with *similar* queries and combined the results selected during these sessions. Results with high social scores were promoted ahead of the results returned by the search engine (Fig. 4).

The key innovation introduced by I-SPY was community-based search. I-SPY allowed users to join one of many communities and perform the search from the community prospect. All social feedback was collected and used independently for each community, i.e., on a group-level, which increased the reliability of search results. A more recent example of social search system with group-level collection of social feedback is the search component of Knowledge Sea II [2].

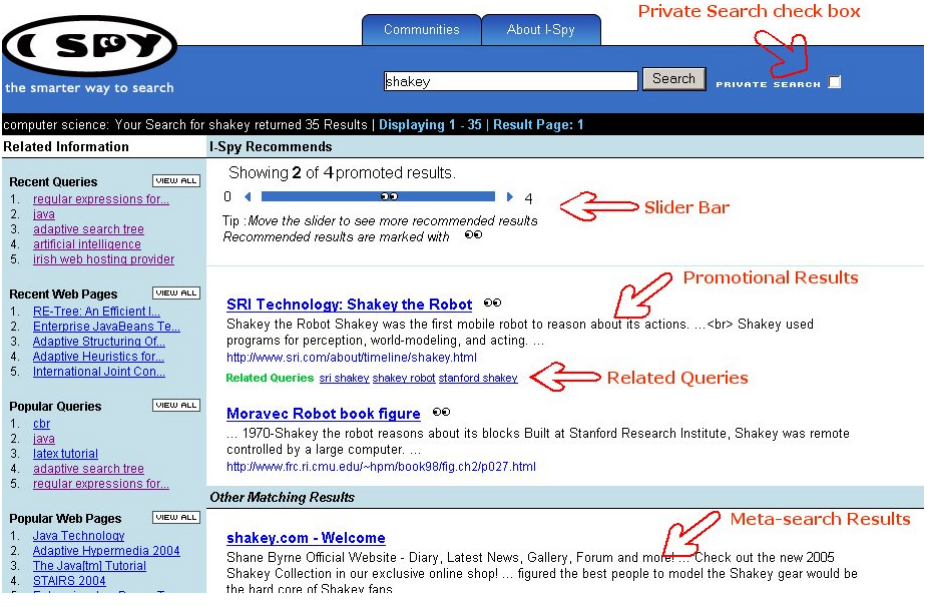


Fig. 4. Promotion of community-relevant results in I-SPY search engine. Promoted links are placed on the top of the results list and annotated with a "pair of eyes" icon.

6 Social Visualization

Social visualization is the least investigated area of social information access, however, it is one of the most promising context of the application of social access ideas, due to the highly expressive power of information visualization (IV). Information visualization allows users to see a set of information resources as a whole, while still being able to recognize individual resources, their properties, and their relationships to each other using relative positioning and visual cues such as shape, color, and size. In contrast to the 1-dimensional guidance provided by the ordered list of objects in IR and IF, a typical IV can use two spatial dimensions in addition to the item appearance to express any information important for the users. This could be critical for social information access systems, which have to present community wisdom to their users *in addition* to regular information presented in traditional information systems. Giving an individual the ability to manipulate the

relative positioning of documents and visual cues allows a social information access system to present more aspects that are valuable to their users. In addition, the higher level of interactivity supports more reliable user tracking techniques.

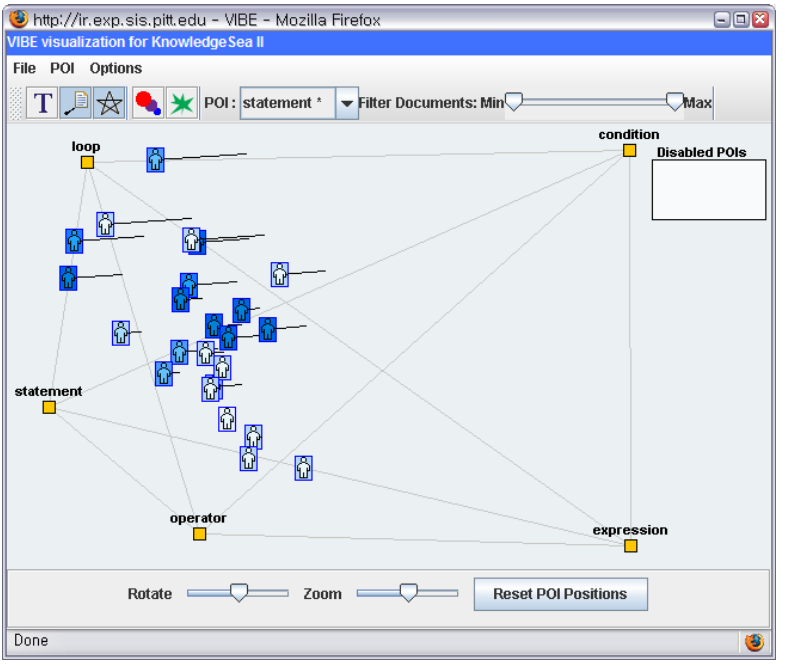


Fig. 5. A visualization of a Knowledge See II cell in Social VIBE system. Information resources are represented by "human on a background" icons. The color density of the background indicates timed page traffic for a group of users.

To some extent, social visualization resembles social browsing and social information filtering. Like in social hypertext browsing, information visualization supports user-driven exploration of information items. However, this exploration is done not by moving from item to item using links, but by exploring and manipulating the visual representation of these items. As a result the context (such as current page or query) either does not exist or is hard to define. It makes the visualization close to information filtering. These similarities hint that approaches from social browsing and collaborative filtering could be appropriate for building social visualization. Indeed, both user-profile based approaches from the area of collaborative filtering and page-centric approaches from the area of collaborative browsing are appropriate for collecting community wisdom in the context of social visualization. A benefit of page-centric approaches is that they may not require individual login, it may be sufficient for users to indicate their community as in I-SPY [48], ASSIST [20] or Conference Navigator [19] systems. If the number of visualized objects is not very large (up to a few thousands) and the groups are reasonably homogeneous, community-level tracking may work quite well. For larger information spaces or to

achieve better precision individual profiling and profile matching may be necessary. On the presentation side, the traditional ranked recommendation list used by collaborative filtering system is hardly appropriate for expressing the community wisdom in social visualization. The ideas of history-enriched environments used by social browsing looks much more relevant. I.e., the representation of information items on the visualization can be altered to express past interaction of similar users (users from the same community or with matching profiles).

Knowledge Sea II presented in section 4 provides an example of social visualization, which uses page-centric group-level activity tracking and expresses it by creating a history-enriched environments with color visual cues. However, it is not the most useful example since the visualization itself is not very typical. It is area-based, not item-based (as in the majority of visualization approaches) and the user can't explore this visualization by manipulating its parameters.

A more straightforward example can be provided by Social VIBE [1]. This system was originally developed in conjunction with the Knowledge Sea II and used to visualize documents within a SOM cell. Social VIBE is based on VIBE [42] interactive spatial visualization approach, which uses document content analysis to present a collection of documents in two dimensions, relative to the *points of interest* (POI). By manipulating the location of the POI, a user can explore the collection and locate relevant documents. In Social VIBE the top cell keywords (the cell focus) are used as POI so the visualization helps users can to discover relationships between the focal keywords and the documents located inside the cell. On Fig. 5 the POIs are shown as small orange squares and the documents are displayed using the same social icons as used by Knowledge Sea II social browsing context (the color of the human figure indicates user personal traffic and its background indicates group traffic). The document positions are determined by their similarities to POIs: the closer a document to a POI, the more similar its contents to the POI. For example, we can see that the documents that are displayed on Fig. 5 are more similar to terms like "loop," "statement," and "operator" than the terms "condition" and "expression." When a user drags a POI around the screen, related documents follow the move, according to their similarity to that POI. Therefore, the user can easily understand the related document by observing these movements. If one document moves a greater distance than the other documents do, when a POI is moved, it means that the document is more similar to that POI. Trails of the movements may optionally be displayed, as in Fig. 5. Social VIBE also provides several other ways to manipulate the visualization (see [1]), but they are not essential for the focus of this paper.

7 Conclusions

This paper reviewed social information access, a new stream of research on the crossroads of information access and social computing. It reviewed the origins of this stream, classified social information access technologies according to the supported information access paradigm, and provided examples of three less explored types of social information access systems. Due to the space limits, some interesting technologies were just briefly mentioned and some were completely left out. The results of empirical evaluation or reviewed technologies were not presented either. In

addition, there was no space to discuss two important integrative topics, which go beyond the technology-by-technology structure of the review. The first of these topics is the emergence of social information access systems, which integrate creatively several traditional technologies such as visualization+browsing in Knowledge Sea II [17] or search+browsing in ASSIST [20]. This direction of research is important since integration expands the volume of community information available for the social guidance algorithms. Another important topic is the modern stream of research on encouraging user contributions in social systems. Ensuring a reliable stream of user feedback is critical for social systems and a number of modern approaches to increase the flow of feedback seriously influence the design of social systems. More information about it can be found in [11; 20; 25]. With all these shortcomings, the author hopes, that this review will be useful to those interested in exploring and implementing social information access technologies.

Acknowledgments. This material is partially based upon work supported by the National Science Foundation under Grants No. 0310576 and 0426021. A number of systems presented in this paper were developed in collaboration with Rosta Farzan, who was supported by National Science Foundation Graduate Fellowship.

References

1. Ahn, J.-w., Farzan, R., Brusilovsky, P.: A two-level adaptive visualization for information access to open-corpus educational resources. In: Brusilovsky, P., Dron, J., Kurhila, J. (eds.) Proc. of Workshop on the Social Navigation and Community-Based Adaptation Technologies at the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 497–505 (2006)
2. Ahn, J.-w., Farzan, R., Brusilovsky, P.: Social search in the context of social navigation. Journal of the Korean Society for Information Management 23(2), 147–165 (2006)
3. Amitay, E., Darlow, A., Konopnicki, D., Weiss, U.: Queries as anchors: selection by association. In: Proc. of Proceedings of the 16th ACM Conference on Hypertext and Hypermedia, pp. 193–201 (2005)
4. Beg, M.M.S., Ravikumar, C.P.: Measuring the quality of web search results. In: JCIS 2002. Proc. of 6th Int. Conf. on Computer Science and Informatics, A Track at the 6th Joint Conference on Information Sciences, pp. 324–328 (2002)
5. Bollen, J., Heylighen, F.: A system to restructure hypertext networks into valid user models. The New Review of Multimedia and Hypermedia 4, 189–213 (1998)
6. Brin, S., Page, L.: The anatomy of a large-scale hypertextual (Web) search engine. In: Ashman, H., Thistewaite, P. (eds.) Proc. of Seventh International World Wide Web Conference, vol. 30, pp. 107–117. Elsevier Science B.V., Amsterdam (1998)
7. Brusilovsky, P.: Adaptive navigation support. In: Brusilovsky, P., Kobsa, A., Neidl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization. LNCS, vol. 4321, pp. 263–290. Springer, Heidelberg (2007)
8. Brusilovsky, P., Chavan, G., Farzan, R.: Social adaptive navigation support for open corpus electronic textbooks. In: De Bra, P., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, pp. 24–33. Springer, Heidelberg (2004)
9. Brusilovsky, P., Tasso, C.: Preface to special issue on user modeling for Web information retrieval. User Modeling and User Adapted Interaction 14(2-3), 147–157 (2004)

10. Burke, R.: Hybrid Web recommender systems. In: Brusilovsky, P., Kobsa, A., Neidl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg (2007)
11. Cheng, R., Vassileva, J.: Design and evaluation of an adaptive incentive mechanism for sustained educational online communities. User Modelling and User-Adapted Interaction 16(2/3), 321–348 (2006)
12. Dieberger, A.: Supporting social navigation on the World Wide Web. International Journal of Human-Computer Interaction 46, 805–825 (1997)
13. Dieberger, A.: Where did all the people go? A collaborative Web space with social navigation information (2000), http://homepage.mac.com/juggle5/WORK/publications/SwikiWriteup.html
14. Dieberger, A., Dourish, P., Höök, K., Resnick, P., Wexelblat, A.: Social navigation: Techniques for building more usable systems. Interactions 7(6), 36–45 (2000)
15. Dieberger, A., Guzdial, M.: CoWeb - experiences with collaborative Web spaces. In: Lueg, C., Fisher, D. (eds.) From Usenet to CoWebs: Interacting with Social Information Spaces, pp. 155–166. Springer, Heidelberg (2003)
16. Dourish, P., Chalmers, M.: Running out of space: Models of information navigation. In: Cockton, G., Draper, S.W., Weir, G.R.S. (eds.) Proc. of HCI 1994, Cambridge University Press, Cambridge (1994)
17. Farzan, R., Brusilovsky, P.: Social navigation support through annotation-based group modeling. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 463–472. Springer, Heidelberg (2005)
18. Farzan, R., Brusilovsky, P.: Social navigation support in a course recommendation system. In: Wade, V., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 91–100. Springer, Heidelberg (2006)
19. Farzan, R., Brusilovsky, P.: Community-based Conference Navigator. In: Dimitrova, V., Tzagarakis, M., Vassileva, J. (eds.) UM 2007. Proc. of 1st Workshop on Adaptation and Personalisation in Social Systems: Groups, Teams, Communities at the 11th International Conference on User Modeling, pp. 30–39 (2007)
20. Farzan, R., Coyle, M., Freyne, J., Brusilovsky, P., Smyth, B.: ASSIST: adaptive social support for information space traversal. In: HT 2007. Proc. of 18th conference on Hypertext and hypermedia, pp. 199–208. ACM Press, New York (2007)
21. Fitzpatrick, L., Dent, M.: Automatic feedback using past queries: social searching? In: Proc. of ACM SIGIR 1997, pp. 306–313 (1997)
22. Freyne, J., Smyth, B.: Cooperating search communities. In: Wade, V., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 101–111. Springer, Heidelberg (2006)
23. Goldberg, D., Nichols, D., Oki, B., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications of the ACM 35(2), 61–70 (1992)
24. Hammond, T., Hannay, T., Lund, B., Scott, J.: Social Bookmarking Tools (I): A General Review. D-Lib Magazine 11(4) (2005)
25. Harper, F.M., Li, X., Chen, Y., Konstan, J.: An economic model of user rating in an online recommender system. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 307–316. Springer, Heidelberg (2005)
26. Hill, W., Stead, L., Rosenstein, M., Furnas, G.: Recommending and evaluating choices in a virtual community of use. In: CHI 1995. Proc. of SIGCHI conference on Human factors in computing systems, pp. 194–201 (1995)
27. Hill, W.C., Hollan, J.D., Wroblewski, D., McCandless, T.: Edit wear and read wear. In: CHI 1992. Proc. of SIGCHI Conference on Human Factors in Computing Systems, pp. 3–9. ACM Press, New York (1992)

28. Hook, K., Benyon, D., Munro, A.J. (eds.): Designing Information Spaces: The Social Navigation Approach. Springer, Berlin (2003)
29. Huang, C.K., Chien, L.F., Oyang, Y.J.: Relevant term suggestion in interactive web search based on contextual information in query session logs. Journal of the American Society for Information Science and Technology 54(7), 638–649 (2003)
30. Jung, S., Harris, K., Webster, J., Herlocker, J.L.: SERF: Iintegrating human recommendations with search. In: CIKM 2004. Proc. of ACM 13th Conference on Information and Knowledge Management, pp. 571–580 (2004)
31. Kantor, P.B.: Historical Note on the archaeology and the history of recommender systems (2007), http://www.scils.rutgers.edu/ kantor/AntWorld/OCLC_ANTWORLD_HistoricalNote.doc
32. Kantor, P.B., Boros, E., Melamed, B., Meñkov, V., Shapira, B., Neu, D.J.: Capturing human intelligence in the net. Communications of the ACM 43(8), 112–116 (2000)
33. Karamuftuoglu, M.: Collaborative information retrieval: toward a social informatics view of IR interaction. Journal of the American Society for Information Science 49(12), 1070–1080 (1998)
34. Keller, R.M., Wolfe, S.R., Chen, J.R., Rabinowitz, J.L., Mathe, N.: A bookmarking service for organizing and sharing URLs. In: Proc. of Sixth International World Wide Web Conference. Computer Networks and ISDN Systems, vol. 29(8-3), pp. 1103–1114 (1997)
35. Kohonen, T.: Self-organizing maps. Springer, Berlin (1995)
36. Konstan, J., Riedl, J.: Collaborative filtering: Supporting social navigation in large, crowded infospace. In: Höök, K., Benyon, D., Munro, A.J. (eds.) Designing Information Spaces: The Social Navigation Approach, pp. 43–82. Springer, Berlin (2003)
37. Lehikoinen, J., Salminen, I., Aaltonen, A., Huuskonen, P., Kaario, J.: Meta-searches in peer-to-peer networks. Personal and Ubiquitous Computing 10(6), 357–367 (2006)
38. Li, W.-S., Vu, Q., Agrawal, D., Hara, Y., Takano, H.: PowerBookmarks: a system for perzonalizable Web information organization, sharing, and management. In: Proc. of 8th International World Wide Web Conference, pp. 297–311. Elsevier, Amsterdam (1999)
39. Maltz, D., Ehrlich, K.: Pointing the way: active collaborative filtering. In: CHI 1995. Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 202–209. ACM Press, New York (1995)
40. Micarelli, A., Gasparetti, F., Sciarrone, F., Gauch, S.: Personalized search on the World Wide Web. In: Brusilovsky, P., Kobsa, A., Neidl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization. LNCS, vol. 4321, pp. 195–230. Springer, Heidelberg (2007)
41. Munro, A.J., Hook, K., Benyon, D. (eds.): Social Navigation of Information Space. Springer, Berlin (1999)
42. Olsen, K.A., Korfhage, R.R., Sochats, K.M., Spring, M.B., Williams, J.G.: Visualisation of a document collection: The VIBE system. Information Processing and Management 29, 1 (1993)
43. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: An open architecture for collaborative filtering of netnews. In: Proc. of ACM 1994 Conference on Computer Supported Cooperative Work, pp. 175–186. ACM Press, New York (1994)
44. Rucker, J., Polano, M.J.: Siteseer: Personalized navigation for the Web. Communications of the ACM 40(3), 73–75 (1997)
45. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Brusilovsky, P., Kobsa, A., Neidl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization. LNCS, vol. 4321, pp. 291–324. Springer, Heidelberg (2007)

46. Scholer, F., Williams, H.E.: Query association for effective retrieval. In: CIKM 2002. Proc. of ACM 11th Conference on Information and Knowledge Management, pp. 324–331 (2002)
47. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating "word of mouth". In: Katz, I., Mack, R., Marks, L. (eds.) Proc. of CHI 1995, pp. 210–217. ACM Press, New York (1995)
48. Smyth, B., Balfe, E., Freyne, J., Briggs, P., Coyle, M., Boydell, O.: Exploiting Query Repetition and Regularity in an Adaptive Community-Based Web Search Engine. User Modeling and User-Adapted Interaction 14(5), 383–423 (2004)
49. Svensson, M., Höök, K., Cöster, R.: Designing and evaluating kalas: A social navigation system for food recipes. ACM Transactions on Computer-Human Interaction 12(3), 374–400 (2005)
50. Wexelblat, A., Mayes, P.: Footprints: History-rich tools for information foraging. In: CHI 1999. Proc. of ACM Conference on Human-Computer Interaction, pp. 270–277 (1999)

Designing Adaptive Web Applications

Peter Dolog

Aalborg University, Computer Science Department
Selma Lagerlöfs Vej 300, DK-9220 Aalborg-East, Denmark
dolog@cs.aau.dk

Abstract. The unique characteristic of web applications is that they are supposed to be used by much bigger and diverse set of users and stakeholders. An example application area is e-Learning or business to business interaction. In eLearning environment, various users with different background use the eLearning system to study a discipline. In business to business interaction, different requirements and parameters of exchanged business requests might be served by different services from third parties. Such applications require certain intelligence and a slightly different approach to design. Adpative web-based applications aim to leave some of their features at the design stage in the form of variables which are dependent on several criteria. The resolution of the variables is called adaptation and can be seen from two perspectives: adaptation by humans to the changed requirements of stakeholders and dynamic system adaptation to the changed parameters of environments, user or context. Adaptation can be seen as an orthogonal concern or viewpoint in a design process. In this paper I will discuss design abstractions which are employed in current design methods for web applications. I will exemplify the use of the abstractions on eLearning web applications as well as on applications for business to business interaction based on web services.

1 Introduction

Adaptive Web-based applications provide an alternative to the traditional "one-size-fits-all" applications [3]. Such applications try to address diverse requirements of different stakeholders by leaving some of their features at the design stage in the form of variables which are dependent on several criteria. The resolution of the variables is called adaptation and can be seen from two perspectives:

- Adaptation by humans to the changed requirements of stakeholders;
- Dynamic system adaptation to the changed parameters of the environment or context.

User-centered adaptive applications utilize user features to resolve the variability; i.e. to determine appropriate information presentation and navigation sequences for exploring a sufficiently complete set of information. They update a user model in accordance with user interaction and the information which he or she has provided. There are several application domains where such adaptive web applications have been found useful such as education, eCommerce, and news.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 23–33, 2008.

Adaptive eLearning Applications. The UML-Guide [7] is an example of an adaptive web application. It geneates a map of an information space designed for a particular information or learning goal. The map itself is adaptive; some of the links and symbols are annotated according to knowledge about a user which is maintained by the UML-Guide.

One approach to the visualization of the navigation map is depicted in Fig. 1. The navigation map displays a composite hierarchy of information nodes in information space (folder symbols with subfolders and document nodes) and sequencing relations between the nodes (arrow symbol).

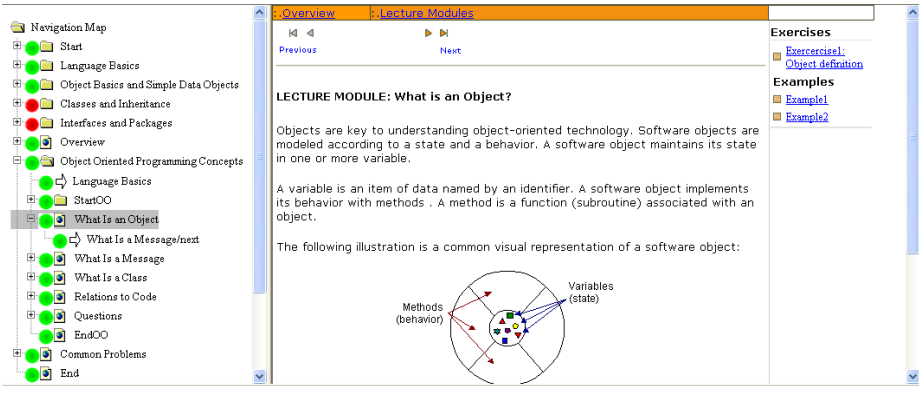


Fig. 1. Visualization of the navigation graph for the Java e-lecture [4]

Besides the adaptation mentioned here, the content served by a web application can be adapted as well. Some fragments of presented content can be hidden, some can be displayed. Composition and placement of the fragments can change according to preferences and user abilities.

The adaptation in such applications like the UML-Guide is usually a decision for a particular information item or function based on knowledge about the items being recommended. The knowledge about items usually comprises what particular information items or functions have in common and where they differ. The properties which differ from item to item determine the source for adaptation. The selection of an appropriate item is based on results from matchmaking between user and information properties. The differences between users in terms of properties and their values determine a selection of different information items or functions which best fit to particular user features according to a chosen selection strategy. As the user's behavior pattern evolves, recommended items may change. This is ensured by continuous updating and evolution of a user's profile based on his or her behavior as traced by the application. In this way, the user always receives up to date information items or functions matching the current state of her profile.

Business to Business Interaction. Supply chain management or financial applications are other examples of application which could benefit from adaptation. In such applications, the adaptivity is considered beyond the user profile and is based on rather different requirements profiles matched with service profiles satisfying a business function. Consider for example a company's monthly payroll processing from [21]. A company has to calculate a salary for each employee. In the next step, the payment of the salary is performed, which comprises several operations. First of all, the salary is transferred from the company's account to the employee's account. Then the company transfers the employee's income tax to the account of the tax authorities. Finally, the company prints the payslip and sends it to the employee. On the employee has only one task which he has to perform each month in this scenario: He transfers the monthly installment for his new car to the car dealer's account.

The company's and the employee's operations are each controlled by a business process, and are implemented using Web services from multiple providers. The business transactions are used in order to guarantee a consistent execution of all required operations. This is depicted in Figure 2. Only the services of transaction T1 are shown.

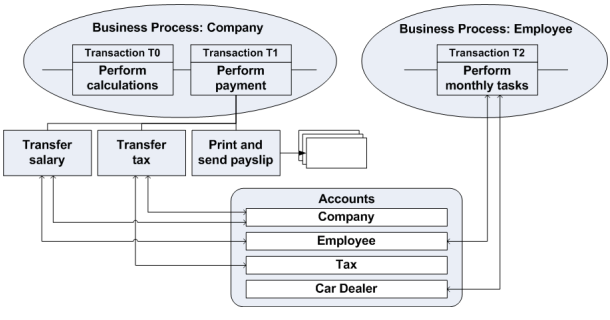


Fig. 2. The motivating scenario

Examples where adaptation/replacement can occur are the following:

1. A service which participates in the transaction fails (for example transfer of the salary fails due to an internal error). Instead of aborting and rolling back to the previous state, a different service can be selected to compensate the failed one. The compensating service is selected based on matchmaking between requested and offered capability. Such a *replacement* is encouraged by the fact that usually multiple services exist that have the same capabilities.
2. A mistake has been made regarding the input data of an operation. In this scenario, it could be that the calculation of the salary is inaccurate, and too much has been transferred to the employee's account. The flaw is spotted by an administrator, and the system offers a compensating service which will correct the mistake instead of aborting the whole transaction.

In the above examples, the adaptation is considered as a replacement either during the business transaction execution or as a post process compensation execution.

2 Web Application Design

The Web-based application development is usually characterized as an integrated set of activities producing three products of a Web application: application domain, navigation, and presentation and their engineering.

Application Domain Engineering deals with analysis, design, implementation, authoring of concepts which are related to the information content to be made accessible through the Web application, and functions to process, access, and guide through. *Navigation Engineering* deals with activities related to analysis, design, implementation and testing of the modality through which users will navigate through the available information and services. In particular, the navigation engineering is concerned with grouping information fragments and functions into navigation nodes (hypertext nodes, contexts, views) and interconnects them by links. *Presentation Engineering* is concerned with analysis, design, implementation, and testing of appearance of information fragments, functions and their results to a user. The presentation model defines spatial layout and content of information fragments related to the user interface. It also defines presentation classes or objects, spatial relationships between them and content associated with them.

There have been several proposals for Web development methods, describing specific activities for Web application development, like OOHDM [22], WebML [5], UML-based Web Engineering [11], and HERA [13] or reference models like the IMPACT-A method [16]. All of them provide design abstractions for above mentioned activities.

Adaptivity is another concern in web application design which is orthogonal to those mentioned above. Systematic analysis and design of adaptive application features require following requirements to be met [6]:

- *Common and Variable Features* — A method and technique is needed, which will support analysis of *common (nonadaptive)* and *variable (adaptive)* parts of developed applications.
- *Multiple Domains* — A Web application usually serves information from several domains and uses different environments to do so. The adaptation is influenced by parameters from user or client constraints domains. Separation of these concerns, according to domains to which they belong to, helps a designer to focus on features which are important for a particular domain.
- *Dynamic connectors between domains* — The separation between several domains allows for better decision making about features in a particular domain. When designing a particular (instance of) a Web application, the domain features have to be connected (configured) in a certain manner suited to the context of the (instantiated) Web application. Appropriate design

technique which supports connectors (compositions) and collaborations between domains is needed. Such a technique helps reason about connections between domains which might encourage their reuse. The connections can be further constrained where the constraints are to be evaluated at run time.
– *Support for adaptive navigation design in connected domains* — the composed information fragments should be further linked to form possible navigation paths. The navigation paths can be further constrained where constraints are to be evaluated at run time.

2.1 Common and Variable Features in Multiple Domains

Adaptation components in adaptive web applications usually recommend one of the several options for links, operations, or content fragments in a content composition. Furthermore, information items can be adaptively configured in the web application based on user profile or abilities of an environment. The options which are planed to be available for adaptation are described using feature models.

Feature Modeling. There have been several proposals for techniques modeling variability and commonality in software systems such as Feature Oriented Domain Analysis (FODA) [15] and or the extended UML structural modeling package with stereotypes for feature modeling (see [6]). Other proposals use different techniques for capturing the variability in software systems like the story boards [2], variation points in assets and components in [1].

Feature model is a set of models which represent configuration aspects of concepts from domains analyzed in web application engineering. Each model in a feature model has one concept and its belonging features. The concept and features are connected to each other by composition relationship. Configuration relations between features and concept are represented as variation points. The concepts and features in feature models are mapped on the concepts and relationships from the conceptual model.

Multiple Domains. The feature models prescribe the parts of the content, environment, and software components, which are stable or common for any user or customer and parts which are variable depending in general on some factors (mostly values of user features or client profiles with requirements for business operations).

The common and variable features can be described separately in:

– *Application Domain* — a domain of information (or content) and user task supporting functions which are served by a web application;
– *Environment Domain* — a domain for representation and organization of information and task supporting functions in a delivery platform;
– *User Domain* — a domain of user features and constrains which are relevant for matchmaking with content, functions and environment to decide for particular option to be offered by a web application.

In application domain, different content can be used to communicate the same information to people with different background. Moreover, the same content can be represented by different media and this content can evolve in time. The content can be also presented in different environment, e.g. as a book, lecture, or an article. Also overall access to the content can be managed through different patterns such as digital library, e-course (virtual university), on-line help, etc.

Each user group may require different information fragment to browse, different composition of presented information (local navigation), and different order and interconnections between information chunks (global navigation). Different navigation styles can also be determined by the target environment where the information is served to a user.

Similarly, different audience may require different appearance and layout of information chunks and different presentation of organization of read information. Target environment can also restrict possibilities to presentation. Thus, it is important to also capture this kind of variability.

Feature Models for Content Intensive Adaptive Web Applications. A feature, as a prominent or distinctive user-visible aspect, quality, or characteristic of a software system [12], in feature models of adaptive web application represents:

- **in an application domain model** — information fragments, which are needed to communicate effectively a concept of a feature model,
- **in an environment/information model** — supporting structural units of a content in particular web-based application,
- **in a user domain model** — qualitative and quantitative features which are needed for decisions about certain adaptation strategy within adaptation process (e.g. a competence acquired within learner performance to decide whether a user is able to grasp particular content item or exercise or metrics of the performance for finer recommendations of next learning steps).

Mandatory features, *Optional features*, and *variation points* are means to analyse and plan a variability of adaptive web applications in the domains mentioned above. The variation points are means to model the dependencies between features and concepts in feature models and can specify *mutually exclusive (XOR),* mutually required (AND), and *mutually inclusive features (OR).*

Feature Models for Web Services and Business Transactions. The domain engineering activities in Web service environments are realized by different independent service providers. The application engineering activities are realized by different parties as well, employing service selection mechanisms and matchmaking to fit particular business activities when utilizing Web services from different providers. Some of the variable features of the Web services can be considered at runtime. Therefore, the software product line engineering process can be tailored to the Web service environment with extended compensation capabilities as follows. Service provider tasks are:

- *Service Domain Analysis* — is a domain engineering process where variabilities and commonalities between service variants are designed to support compensations based on failures or based on different constraints and requirements;
- *Service Domain Design and Implementation* — different service features are mapped onto an implementation and an architecture for service provisioning where some of the features need not to be exposed to the public and some of the variabilities may be left to runtime adaptation.

Client/service consumer tasks are:

- *Business Application Analysis and Design* — is an application engineering task which may be performed by a party external to the service provider and involves the definition of requirements for and constraints on the Web service compensations;
- *Retrieving the Abstract Web Services* — is an application engineering task in which a designer looks for and retrieves Web services which are required to perform business to business conversations;
- *Defining Client Side Compensations* — is an application engineering task in which a designer defines a variability for compensations which will be exploited at runtime if more Web services with similar capabilities have been found, or an alternative Web service has been defined by an application developer;
- *Implementing Client Side Compensations and Functionalities* — is an application engineering task in which the additional compensations are implemented at the client side, as well as additional operations for which there was no Web service found are realized by an application developer.

Web service capabilities or client requirements are placed into the *application domain conceptual model* and the *compensation concepts* are placed into the *environment conceptual model*. The *configuration view* on the concepts in the application domain model and the environment model is described by means of feature modelling. Therefore, the *functionality feature model* as well as the *compensation feature model* are created. Subsequently, the functionality and compensation models are merged to describe the offered capabilities by a service provider, or requested functionalities and restrictions regarding compensations by a service consumer.

Similarly, [10] studies product lines in the context of adaptive composite service oriented systems. A pattern based variability has been employed for development of composite service-oriented systems in [14].

2.2 Dynamic Connectors between Domains

The collaboration diagrams provide useful abstraction to link together several application domains, where the content or service capabilities are comming from, with an environment through which they are provided.

A Story Collaboration Model is a set of collaboration diagrams which define dynamic content chunks or interaction spaces with business functions accessible in particular environments constrained by conditions evaluating partial restricting profile state (user profile or client requirements profile). The story collaboration diagrams contain collaborations between roles created as instance roles of features and concepts from an application domain feature model linked to instance roles of features and concepts from an environment feature model.

At runtime, the feature instances collaborate to create a content which can be modeled as active information objects providing a defined interface to access their content and presentation. In web services domain, features from capabilities domain interact with each other to deliver a requested service together with environment features such as the compensations.

Roles are used to model different purposes of a particular feature or concept in an environment component. Roles terminology can form a complex structures. The UML class diagram can be employed to model such a structure. This model can be used similarly to the domain and environment conceptual models.

2.3 Adaptivive Navigation in Connected Domains

State machines provide a useful abstraction for adaptive navigation design in a web application where the navigation is seen as a guidence through a certain path in a hypertext graph.

A *Navigation State* for a user is an information chunk or an interaction space [9] observed by a user at a hypertext node at a given time. In the UML state diagrams, atomic states can be grouped into *superstates*. States usually refer to concepts of an application domain. The superstate may compose substates in alternate or parallel fashions. Concurrency in web presentations can be handled by *Concurrent regions*, *Fork* and *Join* pseudostates, and *SyncState*.

A *Transition in Navigation Trail* is a transition between one navigation state and another. The transition is usually caused by a user interaction *event* or by another event (e.g. time event). When the transition is fired it leads to a production of a new hypertext node for a user — the new navigation state. *Guards* can be used to constrain transitions, entry, and exit actions of states by adaptation rules. Usually, they consist of a predicate over user profile attributes or context information. The transitions and events on states are useful abstractions for assigning sensors observing user evolution. Each transition can have a side effect *action*. Actions can be performed also at entry, exit and as an internal transition side effect of state. The side effect can be, for example, the modification of a user profile, or the choice of presentation styles for a given chunk of information. Actions can also process parameters used in guards of outgoing part of branches.

The *variability in the navigation trails* is supported by the alternate (OR) states and by decision symbols which can split transition to several alternative transitions. In this way, the navigation trails can have alternate navigation paths and information chunks constrained by conditions referring to certain user, content, device, or environment features. From user point of view it means that

each trail can be adapted by taking into account the user background, level of knowledge, preferences and so on [7].

Similar principles apply to business operations. Those business operations which are dependent on each other may similarly be linked from the user perspective. State machines or transition systems can then be applied in a similar fassion. [18,20], for example, describe the semi-automatic adaptation of a workflow in case of errors. A change of the workflow process can, for example, consist of a deletion or jump instruction, or the insertion of a whole new process segment. The change can either be done on a running instance, or it can be performed on the scheme which controls the workflow, and which results in a change in all running instances. Refer to [19] for details. Labeled transition systems are also used in context based substitution of web services [17].

3 Further Challanges

As the web evolves new opportunities for innovative applications occur. These opportunities, however, also raise many challenges for design. Here I have mentioned just three categories, which I think are very relevant today.

Rich Internet Applications. Rich internet applications are applications which try to enhance user experience on the web and bring it as close as possible to desktop applications. Such applications become popular especially when multimedia capabilities and programming models behind Adobe/Macromedia products and AJAX were introduced. However, the challenge for web engineering methods is how to deal with asynchronous communication, functionality on server side as well as client side, possibly independent autonomous servers, synchronization and so on. In this setting, computation of links is becoming more complex. Issues such as availability of content, which have not been so crucial in closed, one server environments, are now also becomming important.

Social Web Applications. Open adaptive web applications where dynamic groups of users exist pose other challenges on design. The open question is, for example, how to compose user profiles into group profiles. It is also interesting to study in which context a single individual profile is more useful over the group profile. Furthermore, it is also interesting to study how different activities of different groups and different individuals contribute to an effective personalized access to information and operations. This multilevel interaction of various profiles adds a complexity to web applications, thus influencing their design methods as well.

Composition Models. There are two mainstream approaches to handle business transactions, commits, locking, composition, interaction as well as coordination of adaptive web services and applications followed in web services community. On the one hand, there are plan-based design approaches, for example, based on BPEL, which prescribe composition and interaction between participating services. On the other hand, there are middleware approaches with autonomous protocols focusing on environments where services can join and leave on an ad-hoc manner. It

is interesting to study the tradeoffs between them as well as various design techniques either for compositions or for middleware in combination with algorithms, protocols, and computation models for transactions.

References

1. Bachmann, F., Goedicke, M., do Prado Leite, J.C.S., Nord, R.L., Pohl, K., Ramesh, B., Vilbig, A.: A meta-model for representing variability in product family development. In: van der Linden, F.J. (ed.) PFE 2003. LNCS, vol. 3014, pp. 66–80. Springer, Heidelberg (2004)
2. Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J.-M.: Pulse: A methodology to develop software product lines. In: SSR, pp. 122–131 (1999)
3. Brusilovsky, P.: Adaptive hypermedia. User Modeling and User-Adapted Interaction 11(1-2), 87–100 (2001)
4. Ceri, S., Dolog, P., Matera, M., Nejdl, W.: Adding client-side adaptation to the conceptual design of e-learning web applications. Journal of Web Engineering 4(1), 21–37 (2005)
5. Ceri, S., Fraternali, P., Matera, M.: Conceptual modeling of data-intensive web applications. IEEE Internet Computing 6(4) (August 2002)
6. Dolog, P.: Engineering Adaptive Web Applications. Doctoral thesis, Leibniz University of Hannover, Hannover, Germany (March 2006)
7. Dolog, P., Nejdl, W.: Using UML and XMI for generating adaptive navigation sequences in web-based systems. In: Stevens, P., Whittle, J., Booch, G. (eds.) UML 2003. LNCS, vol. 2863, pp. 205–219. Springer, Heidelberg (2003)
8. Dolog, P., Nejdl, W.: Using UML-based feature models and UML collaboration diagrams to information modelling for web-based applications. In: Baar, T., Strohmeier, A., Moreira, A., Mellor, S.J. (eds.) UML 2004. LNCS, vol. 3273, pp. 425–439. Springer, Heidelberg (2004)
9. Dolog, P., Stage, J.: Designing interaction spaces for rich internet applications with uml. In: Fraternali, P., Baresi, L., Houben, G.-J. (eds.) ICWE2007. LNCS, vol. 4607, pp. 358–363. Springer, Heidelberg (2007)
10. Hallstein, S., Stav, E., Solberg, A., Floch, J.: Using product line techniques to build adaptive systems. In: SPLC 2006. 10th Intl. Software Product Line Conference (2006)
11. Hennicker, R., Koch, N.: A UML-based methodology for hypermedia design. In: Evans, A., Kent, S., Selic, B. (eds.) UML 2000. LNCS, vol. 1939, Springer, Heidelberg (2000)
12. American Heritage: The american heritage dictionary. Houghton Mifflin, Boston, MA (1998)
13. Houben, G.-J., Barna, P., Frasincar, F., Vdovjak, R.: Hera: Development of semantic web information systems. In: Lovelle, J.M.C., Rodríguez, B.M.G., Gayo, J.E.L., Ruiz, M.d.P.P., Aguilar, L.J. (eds.) ICWE 2003. LNCS, vol. 2722, pp. 529–538. Springer, Heidelberg (2003)
14. Jiang, J., Ruokonen, A., Systä, T.: Pattern-based variability management in web service development. In: ECOWS 2004 (2005)
15. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E.: Feature-oriented domain analysis (foda) feasibility study. Technical Report CMU/SEI-90-TR-21, ESD-90-TR-222, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 (1990)

16. Lowe, D.B., Bucknell, A.J., Webby, R.G.: Improving hypermedia development: a reference model-based process assessment method. In: Hypertext 1999. Proceedings of the ACM International Conference on Hypertext and Hypermedia, Darmstadt, Germany, pp. 139–146 (February 1999)
17. Pathak, J., Basu, S., Honavar, V.: On context-specific substitutability of web services. In: ICWS 2007. IEEE International Conference on Web Services, Salt Lake City, Utah, USA, pp. 192–199 (July 2007)
18. Reichert, M., Dadam, P.: ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. Journal of Intelligent Information Systems 10(2), 93–129 (1998)
19. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive Process Management with ADEPT2. In: ICDE, pp. 1113–1114. IEEE, Los Alamitos (2005)
20. Rinderle, S., Bassil, S., Reichert, M.: A Framework for Semantic Recovery Strategies in Case of Process Activity Failures. In: Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J. (eds.) ICEIS, pp. 136–143 (2006)
21. Schäfer, M., Dolog, P., Nejdl, W.: Engineering compensations in web service environment. In: Fraternali, P., Baresi, L., Houben, G.-J. (eds.) ICWE 2007. LNCS, vol. 4607, pp. 32–46. Springer, Heidelberg (2007)
22. Schwabe, D., Rossi, G.: An object-oriented approach to web-based application design. Theory and Practise of Object Systems (TAPOS), Special Issue on the Internet 4(4), 207–225 (1998)

Best of Both: Using Semantic Web Technologies to Enrich User Interaction with the Web and Vice Versa

Martin Dzbor

Knowledge Media Institute, The Open University, UK
mdzbor@acm.org

1 Background

Human-computer interaction is a well-established and rich subject that has an impact not only on those who develop computational systems, but also on the users of such systems, the vendors, maintainers, and many more stakeholders who are normally involved in designing and delivering software and computer-based tools. Interaction in this context is seen broadly – in general, it involves three constituting parts: *the user*, *the technology*, and *the way they work together*. One can then study such phenomena as how the users work with a particular technology, what the users prefer, how the technology addresses given issues, etc. In this contribution I want to look at a specific type of user interaction – with semantically enriched content in general, and with ontologies in particular.

Human-ontology interaction can be seen as a subset of user interaction issues that apply to specific tasks and specific technologies. On the level of task we can see the emergent importance of *inference* and *reasoning*, while on the level of technologies we are aware of markup languages for denoting semantic content (e.g. RDF or OWL), but also of an increasing number of techniques to manage, learn or evaluate semantic content of different kind. We will return to some of these techniques later.

While the awareness of human-computer interaction is a subject almost as old as the computer science, the specifics of interacting with ontologies were not considered in much depth. Tools supporting ontological engineering are considered to be primarily software artifacts, and thus, it is presumed that general user interaction rules apply to ontologies and other semantic content. To some extent, this is true; however, design and subsequently application of ontologies in our everyday activities are indeed specific ways to interact with the technology.

In this contribution, we look at several different aspects of how a user may interact with ontologies in a varied sort of ways. In the first part (sections 2 and 3) we will explore a range of existing web interaction techniques and tools with the discussion of how they may apply to facilitate user interaction with semantic content. In the second part (section 4) we will look at two techniques that aim to intertwine the Web with the Semantic Web, and thus simplify the navigation through semantic spaces on one hand and the interpretation of semantic content on the other hand.

2 User Interactions with the Semantic Web

In the previous publications [13] we analyzed the state of the art tools for engineering ontologies based on the paradigm of reusing an existing content. The main

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 34–49, 2008.

conclusions from that study were that the user group that engages in ontology design faces numerous issues with expressing their modeling intentions in the tools they have at disposal. One of the observations was the confirmation of the gap between the language people think in and the language they have to encode their ontological commitments in.

This is an important group of users; however, these are not necessarily the only users who may have a need to interact with networked ontologies. The issue of interacting with ontologies effectively and efficiently is much more pressing with less experienced users, who carry out an ad-hoc, occasional ontology-related task. An example of such a task may be to use the ontological content to annotate and interpret a textual document.

2.1 Familiarity of Tools Supporting Users

Although the key interaction with the Semantic Web is the interaction with a single type of software artefact: ontology, there are significant difference that need to be kept in mind. Users involved in ontology-driven production of information and knowledge need to be equipped with a range of software configurations and diverse user interfaces to deliver the outcomes of their work as effectively and efficiently as possible. There are two broad strategies how one can match the tools to the needs:

1. different tools for different users and different purposes and
2. different configurations of one tool or toolkit for different users or purposes

The two strategies are not mutually exclusive; very often we find users rely on a limited range of tools, and then may have different, specialized configurations for some of those tools. Let us briefly consider the key advantages and disadvantages of the above approaches:

In the former situation, tools are well defined but apparently independent of each other. This may lead to a proliferation of a large number of highly specialized tools, something that is unlikely to alleviate the user's confusion and overwhelming. Moreover, with specialized tools, there is an increasing risk of them being mutually less compatible or compatible on a rather cumbersome level (e.g. import/export mechanism of various graphical editors is a good example of this compatibility issue). The main advantage is that the user will only get to work with tools and interfaces s/he necessarily needs to carry out given tasks, and nothing more.

In the latter situation, we tend to see more complex and multi-functional tools that can exhibit a variety of user interfaces and user interaction components in different situations. In many tools of this type, we see an aggregation of functionalities and a fairly seamless switching between many tasks the user may carry out at some point. This is essentially a "one-stop shop" approach where the user has (almost) everything they may ever need already inside the tool, and only needs to activate different configurations. A typical example of this would be editors like Microsoft Word, and its 'rich document editor' face, as opposed to (say) 'content revision' face or 'mail merge and distribution' face.

As we suggested in this brief discussion, the assumption that a new technology (e.g. Semantic Web) has to introduce new tools and user interaction metaphors may be actually incorrect. In line with the initial definition of user interaction as a triangle

comprising users, technologies and the application patterns, we can even argue that the three nodes of the triangle interact with each other and have impact on the overall user's experience with the applied technology. However, they may have different weight in different interactive situations. So, if the user is introduced to a new technology and at the same time to a new tool to facilitate his or her interaction with that technology, unnecessary confusion may arise.

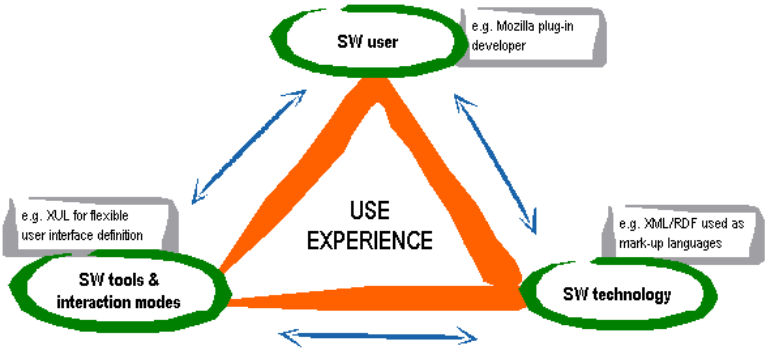


Fig. 1. User – User Interaction Paradigm – Technology triangle

Thus, intuitively it seems reasonable to argue for reusing the existing, familiar user interfaces to introduce novel technologies – through the "back door". A similar observation has been formally made in the aforementioned study with the ontology designers, where users missed common features they got used to – e.g. bookmarks to flag the ontology section they work on, keyboard vs. mouse interactions, etc.

In terms of our simplified triangle this can be generalized so as to limit the number of unknowns in the triangle, or to introduce the innovations in steps rather than disruptively. Take the semantic web technology such as RDF for annotating documents that we want to push to the user. That makes one node in the triangle a variable. As the RDF technology is unfamiliar, it brings in uncertainty for the user, who may not have had any experience with it, may not know its benefits, etc. Now, if we decide to realize this technological change in a radically different user interface from what the user is familiar with, we are introducing another unknown into the interactive experience triangle.

2.2 Towards Universal Usability

The idea of taking advantage of the existing approaches has been explored by the user interaction scientists in the past with a notable contribution made by Shneiderman in terms of his argument for a *universal usability* [27]. In his seminal work, Shneiderman points to several factors that may affect the tool usability. These are factors that vary from one user to another, and hence trigger a degree of adaptation to the user interface. Importantly, Shneiderman highlights many common factors that are not always recognized as valid reasons for UI customization. For example, he talks about technological variety (i.e. the need to support a range of software and hardware

platforms, networks, etc.), about gaps in user knowledge (what users know, what they should know, etc.), or about demographic differences (skills, literacy, income) or environmental effects (light, noise, etc.)

One approach to achieving more universal usability of a tool is to introduce user interface (UI) adaptation into the loop. The rationale is that while a standard UI may not fit the user completely, it might be tweaked so that it gets as closely as possible to the user needs. There are two distinct strategies of how UI adaptation may be accomplished. Since this differentiation may have impact on what is actually modified in the tool, we decided to include this brief detour to generic issues of adaptation. The two strategies distinguish between the following types [21]:

- **adaptive UI:** These are systems and user interfaces that are capable of monitoring its users, their activity patterns, and automatically adjust the user interface or content to accommodate these local differences in activity patterns (which may be due to user's skill, preference, etc.).
- **adaptable UI:** These are systems and user interfaces that allow the users to control and specify adjustments, and often come with the provision of some guidance or help.

According to the informal definitions, the difference is in the actor; who performs the adaptation act. In adaptive UI-s it is the tool, applications or the system that takes the active role; whereas in adaptable UI-s it is the human – typically the actual user of the system, but possibly another user (such as system administrator).

Why do we mention user interface adaptation in this context? Ontologies are highly structured, formalized artefacts that have sufficient expressiveness to describe the structure of a system, tool, or its user interface. Considering that such common tools as web browsers already make use of ontological formalisms to support customization and thus make life easier for the user, it is rather surprising that very little of a similar approach is used to improve the interaction with ontologies for the end users such as bioinformaticians, students, analysts and indeed general public.

Let us therefore start with a brief review of how familiar web interaction metaphors may find their use in the context of a new paradigm of the Semantic Web.

3 Familiar Tools for the (Unfamiliar) Semantic Web

In this section we look at the characteristics of a Semantic Web browsing application with respect to support some form of enriched or enhanced navigation through a rich, semantically encoded space. Quan and Karger [24] suggested that the primary functionality an application for the Semantic Web should satisfy is "to separate the content – the proper purview of the publisher serving the information – from the presentation – an issue in which the end user or their local application should have substantial say."

Since the discussion of generic aspects of what may constitute a Semantic Web browser and navigation support tool were published earlier in [15], here we limit ourselves to reviewing and discussing different types of tools that the researchers and developers borrowed from the Web and applied to the Semantic Web. In particular we

look at the use of graph metaphor, the use of multidimensional facets, the role of styles and templates, and finally the metaphor of semantic layering.

3.1 Navigation in Semantic Graphs

One example of a user interface that extends a familiar metaphor of navigating in semantic graphs by means of expansible nodes and arcs is Tabulator [2]. The project started to demonstrate the "serendipitous re-use" and to address the "explore vs. analyze" tension in user interface design in an open-world of interlinked semantic data. Tabulator was designed as a generic browser for linked data, without the expectation of providing domain-specific interfaces. However, it permits domain-specific functionality (such as calendar, money or address book management) to be loaded transparently from the web.

For the Tabulator the logical, semantic graph is the primary source of data; the web documents are optional and secondary. Hence, the user can explore the graph of data as a conjunction of all the graph documents that have been read (in a particular browsing session). While this approach may not allow directly browsing the documents, it allows the user to check the provenance or source of any piece of information included in the browseable graphs.

Tabulator operates in two modes: exploration and analysis. In the exploration (open world) mode it allows the user to explore the data graph, without the user having to provide all the data – Tabulator implicitly follows links that may contain RDF data about relevant nodes. Data is typically presented as traditional graphs of nodes and arcs, which in Tabulator are called outline views. In the analysis mode, the user may select some nodes or arcs to define patterns of a query, which is then executed by the tool. Query results may be displayed in different views and may be mashed together.

While this approach is sufficiently generic and *universally usable*, it has some shortcomings when it comes to interpreting and making sense of the data. The focus on the individual nodes and immediate links (relationships) may be often reductive. It treats knowledge as an abstract artifact that can be removed and expressed independently of any specific context. Thus, the metaphor may suit better those who want to use ontological data in the abstract, conceptual sense – i.e. without the need to ground the ontology in a document or another data set.

3.2 Navigation in Multidimensional Facets

One typical feature of semantic content is that its entities can be formally assigned many properties and may participate in numerous relations with other entities. If each relationship and property is treated as a potential "dimension" along which one may navigate, the metaphor of multidimensional navigation becomes applicable. One interaction strategy for such a type of data – in addition to simple searching and browsing – is faceted browsing, where users filter an item sub-set by progressively selecting from valid dimensions of an appropriate classification. On a non-semantic level, the strategy was piloted in Flamenco [28] that used metadata to guide users through the choices of views, thus helping them to organize the underlying collection.

Although without ontological support, Flamenco used the notion of lateral links to complement the standard hierarchies.

For semantic (i.e. RDF-encoded) data, Longwell is an out-of-the-box faceted browser intended to be used for viewing arbitrary, complex RDF datasets in a user-friendly way. It was deployed in various contexts both domain independent and dependent [22]. One of its most important features is that of facet extraction from RDF literals and support for inference. A similar approach can be seen in `/facet` – a browser for heterogeneous RDF data that handles collections of different types of items unlike most other, more specialized faceted browsers [16]. It explores the facets of items related by `subClassOf` relation, uses intersections, and can also collapse facets related through `subPropertyOf` relation.

In mSpace paradigm [26], the Semantic Web is seen as a hypertext system. Hence, mSpace extends the faceted browsing paradigm and its functional operations like slicing, sorting, swapping, adding, or subtracting to the semantic content. mSpace relies on ontological knowledge as a basis on which the above operations can be realised in a domain-independent manner. It supports direct manipulation of the ontology representation and the selection of instances associated with its current configuration. Its logic also provides for automatic reasoning to ensure that only meaningful attribute ordering/selections occur. In addition to traditional facets, mSpace adds a few experimental semantic add-ons – e.g. its numeric volume indicators are interesting as they help the user reflect on what (volume of data) to expect behind a particular facet.

Unlike the metaphor of semantic graphs, faceted navigation makes the semantic content much easier accessible to the ordinary users. For example, it completely hides the complexity of interpreting all the relationships an entity engages in at once. Instead, its step-by-step approach to unfolding the space seems simple and intuitive. On the other hand, the user can meaningfully process a small number of facets – e.g. the tools mentioned above usually demonstrate their functionality with three to five facets. This has as implications in terms of hiding too much semantic information from the user, which to some extent violates the motivation for using ontologies in the first place – to facilitate knowledge sharing in explicit, well-defined and formal terms.

3.3 Navigation Using Styles and Templates

This approach to browsing relates to the definition from Quan and Karger mentioned at the beginning of this section arguing the complete separation of content from presentation. It emerged from the fact that much information present on the Web is already stored in relational form, in the database-driven web sites. Therefore, another way to resolve the knowledge acquisition bottleneck is to take advantage of the structural clues of this structured web content to re-create the original information stored in the databases backing this content.

Thresher [18] is a system build on top of the Haystack platform [23], which allows non-technical users, rather than content providers, to “unwrap” the semantic structures buried inside human-readable Web. It provides users with an interface to “demonstrate” the extraction of semantic meanings, and through such demonstrations it learns mappings between the regularities in the document structure and the semantics. Thresher then automatically applies the mappings to similar documents.

Thus, extraction of semantic data is separated from its presentation, and is accessible for further reuse via the Haystack platform.

A notable trend related to templates is the push away from the heavy, specialized clients towards lightweight clients – often in a form of plugins or bookmarklets. A lightweight equivalent of Haystack is PiggyBank [19], which runs as a browser plugin, allowing the user to collect and browse found semantic information. Solvent, another web browser plugin, performs Thresher's role for PiggyBank, allowing the user to visually annotate the document with common vocabularies (e.g. Doublin Core) and generate "scrapers" that extract, convert and store in PiggyBank the "unwrapped" semantic information.

Exhibit [20] is another tool from the same family as the above tools, is a JavaScript-based approach that exposes structured data to the Web using styles and templates. Rather than directly showing graph structures, Exhibit, PiggyBank and others emphasize the need to make the semantic content human-friendly – hence, templates and styles serve to 'prettify' the graphs and show them in a variety of familiar metaphors (e.g. timelines, maps, tables, etc.) One shortcoming of this approach is its focus on presenting simple annotations — they rely on the fact that inference support and additional personalization by the users are not needed in most browsing scenarios.

Similarly as with the previous metaphors, the issue of removing knowledge from its original context is also applicable to styles and templates. In fact, the use of templates in conjunction with formal knowledge may be seen as an introduction of new context for that knowledge, which may not always be desirable.

3.4 Navigation by Means of Semantic Layering

Semantic layering is a notion that was introduced in connection with browsing and navigation in early papers about Magpie [9, 10]. However, this metaphor has been used previously by annotating and entity discovery algorithms (e.g. [6]) to present outcomes of the text analysis. In the context of Magpie, its browser extension (plugin) contains a small HTML parser that finds and highlights the entities from a particular ontology in the current web page. A pre-condition of a successful parsing within the Magpie-enabled web browser is the definition or download of an ontology-derived lexicon. Lexicon- or gazeteer-based parsing is extremely fast approach to carry out an entity recognition task in plain text [17]; in case of Magpie it helps keeping the time overheads of semantic layering comparable to those for network latency.

The outcome of this text processing by Magpie plugin is annotation of an entity instantiation in the web page using custom tags, which can then be activated using Magpie toolbar. This process creates a semantic layer over the original document. The original content remains unchanged; only the interesting concepts and the corresponding text are highlighted. This approach to visualizing the semantic layers puts the users in control of what knowledge is visible at any time, which in turn reduces a common problem of overwhelming the users with too much information (as in many link recommenders, annotators or data fed systems.

Annotated and highlighted concepts become 'hotspots that allow the user to request a menu with a set of actions for a relevant item. More on this is in the next section. Here it suffices to say that web service choices depend on the ontological classification of a particular concept in the selected ontology and on what services are

available for the concepts in a given ontology. Magpie plugin is wrapping the requests for a particular service into a URI, which is then unwrapped to communicate with the actual web service using SOAP (or correctly SOAP over HTTP). The results from the individual web service may be fed into appropriate forms and visualizations based on data retrieval, knowledge-level inference, statistical correlation, or their combination.

Unlike the previous three metaphors that focus on capturing abstract knowledge and acknowledge context as a secondary thing, semantic layering treats knowledge in situ, in its original surrounding. It also acknowledges the fact that the same piece of text can be interpreted using different ontological commitments – every ontology resulting in different entities being relevant and annotated. Since this approach is the core of our research, we will devote more space to analyzing pluses and minuses of this user interaction metaphor.

4 Enhancing Semantic Layering

In the scenario discussed in the previous publications [11] we considered Magpie used as a dynamic educational resource supporting undergraduate students at The Open University (UK). In this scenario, Magpie assisted these students by facilitating a course-specific perspective on scientific texts, analyses and publications. Since Magpie is a generic framework, several demonstrators have been built by using different sets of ontologies, different web services and inference methods. In one such large-scale example Magpie uses a 50k term large thesaurus of terminology related to agriculture reused from resources of the Food and Agriculture Organization of the United Nations (Fig. 2). Web services developed for FAO include term translations (in conceptual and natural languages) and the conceptual navigation through semantically close entities.

4.1 Positive Experiences with Semantic Layering in Magpie

The Magpie framework is open and transparent to these types of ontological commitments. For the user the choice of ontological perspective alters the appearance of the graphical user interface – the Magpie toolbar. Users can at any time switch to a different ontology (if available) – e.g., if they want to re-interpret the document in the context of agriculture instead of climate science. This is an important design feature, as it allows one to circumscribe the scope of web pages in an open world. The control over ontological viewpoints is up to the user, rather than the knowledge engineer.

Another positive aspect is the capability to interact with the user via semantic web services – these can even be derived automatically based on a given ontology. Services can be obviously composed, and thus a more natural and richer user experience can be achieved. For example, the content of service response in Fig. 2 is computed from several independent web services (e.g., term code to label transformation and a calculation of a semantic neighbourhood). Hence, the user's single click gives a more comprehensive result that would normally require more complex, manual composition of partial results, their interpretation, etc. The degree of sophistication of the web services is independent of the Magpie architecture, which sees services as black boxes.

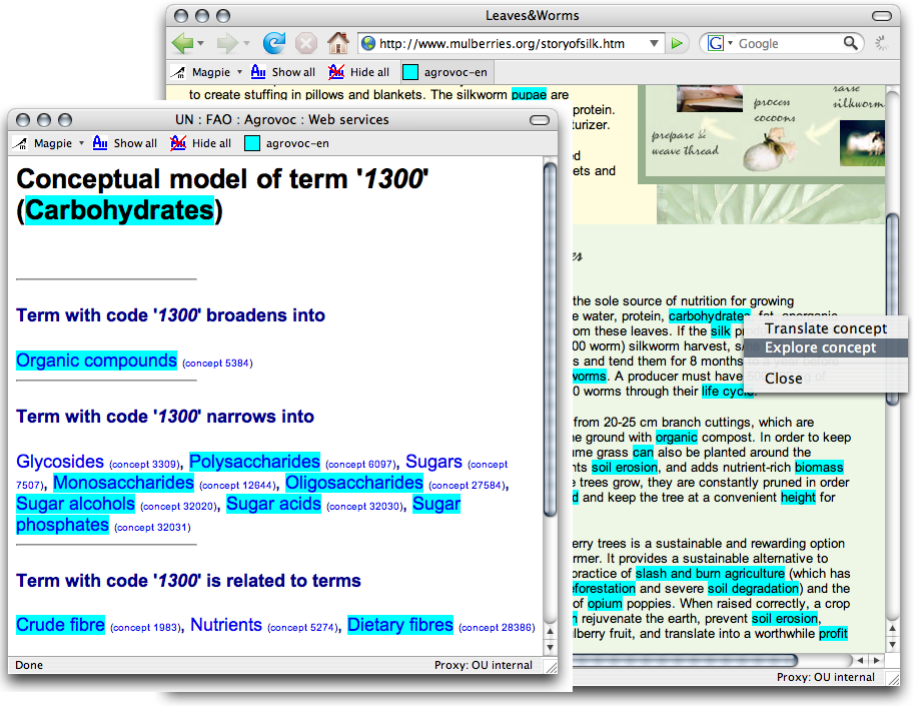


Fig. 2. Magpie demonstrator: (a) applying the Agrovoc-derived lexicon to a web page, and (b) an associated web service facilitating conceptual navigation

Since an answer to a semantic query/service may be a web resource in its own right, it can also be semantically browsed and annotated. Here Magpie merges the mechanisms for recognizing semantic relevance with browsing the results of semantic inferences similarly as some other open hypermedia systems.

4.2 Negative Experiences with Semantic Layering in Magpie

Although the "single ontology = single interpretative perspective" paradigm used by Magpie reduces the size of the problem space, this reduction is not always helpful. Although it focuses the user's attention (as intended), it also unduly restricts the breadth of the acquired knowledge (this was clearly not intended). For instance, during a study session a student may come across a few similar but semantically not entirely identical study materials.

This means that at each page, the student would benefit from minor tuning of the used ontology, glossary and/or service menu. These tunings reflect slight shifts within a broader problem space, which is a fairly common tactic we use everyday to deal with the open situations. Thus, Magpie's design actually features a gap between the inherent notion of a single, formal, sound but 'semantically closed ontology guaranteeing a certain precision within the domain, and the desire to open up the

interaction by supporting multiple services, as well as multiple ontologies (at the same time, without explicit user's reloading step).

4.3 Future of Semantic Browsing

As we suggested in the previous section, one experience from semantic layering that seems to hamper the full potential of the approach is the actual selection of the ontological perspective. A single ontology can hardly capture the diverse contexts and variations in the meaning of resources found on the Web. Hence, as argued also in [15], one of the key issues to enable full-power semantic browsing is the capability to apply *multiple ontological perspectives in multiple user contexts*.

This may seem a straightforward requirement to extend the user interface and simply include several ontologies in place of a single one. However, in practice, this is heavily dependent on the capability of the semantic browser to find and/or retrieve the relevant ontologies in the distributed setting of the Web. In this section, we present our view of strategy that uses the existing Web and the existing partial semantic resources as a background knowledge to enable automatic use of multiple ontologies during browsing.

The idea of relying on the redundancy of information published on the Web and using it as a large-scale repository of background knowledge to helps solve hardly tractable problems is not novel. It has been applied e.g. in ontology learning and relation population scenario in [4]. Another use of a similar motivation has been in [25] for the purposes of ontology mapping. However, both these approaches could be labeled as "back-office" support, i.e. the Web is interacting with techniques, algorithms and heuristics before the any result is proposed to the user. The difference of our approach is in stepping toward the front-end support and exploit the Web as a source of ontologically captured knowledge that can help other users to navigate in a semantically meaningful way.

First, we explain how we use the Web as a source of ontologies and semantic content; alongside with justifying why the existing infrastructures (such as e.g. Google) are not sufficiently expressive to do this task. Second, we propose how the user can interact with a potentially large number of ontologies semantically applicable to a particular web document.

4.3.1 Finding Distributed Ontologies on the Web

The requirement of coping with multiple ontologies on the user level depends, to some extent, on an existence of a broader infrastructure supporting quick and efficient selection and application of ontologies. It has been observed that the number of ontologies and semantically marked up data is growing at a rapid pace. However, the present knowledge of the quality of this generated and designed content in the distributed Semantic Web resources is very sparse. To that extent, we are investigating an infrastructure known as Watson [7], whose primary functions are:

- to enable advanced Semantic Web applications access and use ontologies that may be distributed throughout the Web,
- to enrich access to networked ontologies by taking into account their quality, the relationships and dependencies among, and

- to improve our understanding of the nature of the ontologies on the Semantic Web, so that these can be effectively and economically re-used by the new generation of Semantic Web applications (e.g. question answering tools, ontology mapping engines, flexible semantic browsers, etc.)

Watson infrastructure offers a scalable mechanism for discovering, selecting, and accessing ontologies distributed over the Web. It is a standalone (i.e. semantic browsing independent) infrastructure with several benefits over similar competitors. For instance, Swoogle [8] – the best known 'ontology search engine' has a broad coverage of semantic content, but it suffers from two issues that make it less reliable for advanced Semantic Web applications:

- indexed semantic content is often of mixed quality; Swoogle does not use any stronger notion of ontology quality there are several independent works in the area of ontology evaluation, but these do not benefit the Swoogle users;
- its primary criterion is akin to Google's PageRank, i.e. reflecting popularity of ontologies rather than more practical aspects such as domain coverage or similarly; furthermore, Swoogle's emphasis on web-like treatment of ontologies overlooks their semantic content

One effect of this web-view in Swoogle cited and discussed in [7] is existence of semantic duplicates or near- duplicates – i.e. files having different URLs but semantically being equivalent. The lack of treatment of such a simple issues as semantic duplication then may lead to skewing the performance of advanced applications like semantic browsers. While duplication is a useful measure to tell something about the popularity of a certain opinion, it is a misleading factor if it starts influencing e.g. the meaning disambiguation and similarly.

Hence, infrastructure like in Watson is needed to create more in-depth knowledge about the distributed semantic content available for the purposes of browsing, question answering, etc. Watson's initial contribution is in discovering ontologies and other semantic content by means of commonly used data harvesting techniques. These may produce a large quantity of input, so the actual added value of this envisaged framework is the *semantic and qualitative analysis* of the harvested content.

In addition to simple analytic information (e.g. languages, formats, expressivity, similarity, versions, etc.) it is important to know about the *topological and networked relationships* among the individual ontologies and semantic data sets as wholes. Once such information is acquired, we may consider tackling the second criterion in the semantic browsing applications – the support for multiple ontological frames.

4.3.2 Semantic Browsing Using Multiple Ontologies

The new semantic browser ('PowerMagpie') we are developing relies on the generic Watson framework introduced in the previous section. The role of Watson in this semantic 'power browsing' tool is to identify and make available ontologies that are interesting or otherwise relevant to the content of the web page visited by the user. Watson enables us to extend the semantic layering to multiple ontologies, where in turn, we need to distinguish the ontologies of different quality, topic coverage, expressivity, etc., rather than merely finding any semantic content containing a given keyword.

Once the external (semantic) content provider, such as Watson, provides ontologies or other semantic content applicable to the visited web page, this is passed onto the user interaction component, and is presented to the user. Contrary to the previous version of Magpie, where the user was restricted to one mode of interaction (i.e. semantic layering with ontology-specific categories), the new framework is more flexible. Apart from offering various interpretative perspectives on a single web page, the PowerMagpie browser is able to discover further semantically related resources, which are not directly or indirectly referred by the selected ontology.

The key principle of this functionality is the fact that each ontology models a certain aspect of the world and does it from a very particular perspective, which is by no means exhaustive.

This strategy is common in information search and retrieval – e.g. under such names as query expansion. However, a vast majority of search engines bases the similarity on the lexical proximity of the resources, which, in turn, draws upon the underlying search index. When such a service is implemented outside the search engine scope – i.e. it cannot make direct use of the document index to explore the resource neighbourhood – a dynamic, document-specific descriptive vector of terms needs to be computed. In our framework, this capability is referred to as document fingerprint, and it is somewhat resembling a summary of the document, a set of key defining terms.

We are experimenting with various possibilities and methods to compute document fingerprints, which are to be modularized and parameterized by the chosen user perspective. A sub-set of such terms can be submitted to a search engine, or to our semantic our alternative of a search engine – Watson Semantic Gateway (described in section 4.1. The key idea of interfacing Watson rather than standard search engines such as Google is to find and reuse other, already formalized and represented conceptual commitments that may be captured in several ontologies that Watson discovered on the web.

This technique is inherently iterative: the web browser plugin starts with an initial document fingerprint and tests its conceptual fitness against the existing ontologies. From the most relevant ontologies5 one can calculate semantic neighbours of the matched concepts, which, when returned to the PowerMagpie plugin serve as candidate fingerprint extensions. The plugin attempts to find matches to these fingerprint extensions, thus disambiguating between the different perspectives in which a document might be interpreted.

The discrimination between the different perspectives not only facilitates different navigational paths for the document interpretation, but it also offers opportunities for an implicit annotation of the resource or for an implicit population of the ontologies. In the previous versions of the semantic web browsers, such annotations were merely visual and from the reuse point of view transient. Now, with discovering new ontologies it makes sense to store the annotations locally. One formalism that has been recently agreed upon to facilitate this reuse is RDFa [1], which brings in either ad-hoc or ontology driven annotation to the level that can be easily parsed and reused – e.g. for the purposes of social semantic annotations or tagging applications.

5 Conclusions and Discussion

The Semantic Web is gaining momentum and more semantic data is available online. This has an impact on the application development strategies. The original Magpie, which started our interest in user interfaces for navigating and browsing the Semantic Web, appeared in the time before this momentum became visible. Therefore, one of the key assumptions behind the 'classic' Magpie of having no or little semantic mark-up available is now becoming obsolete. In fact, there are nowadays many ontologies available, which brings in new challenges.

The above-mentioned momentum also implies that the new generation of Semantic Web application needs to work with more heterogeneous and distributed semantic data. Hence, another design principle typically occurring in the applications from late 1990s and early 2000s (a single ontology) is challenged by this environment consisting of distributed and networked ontologies.

The idea tested by us in the early applications of the Magpie technology of interlinking semantic annotation, semantic browsing and semantic services is gaining popularity. Annotation is no longer a separate objective in its own right; most of the new annotation tools aim to offer additional services; e.g. validation or consistency checking. A major challenge stems from the need to make the association between semantic services and semantic mark-up more open, flexible and simpler. Clearly, more usable techniques are needed for marking up, discovering and deploying such services, so that these are more readily available to the users.

A potentially interesting input is likely to come from the deployment of modular ontologies and specialized services as opposed to monolithic ontologies with tightly integrated web services. Namely, the modular approach may allow some of the services to be involved in evolving the general knowledge captured on the Semantic Web. Some methods may use statistical techniques, where other services may rely more on social trust. It seems that knowledge evolution would need to be investigated from two complementary perspectives – (i) formal knowledge evolution based on logical inference with an aim to assure consistency in knowledge bases, and (ii) social knowledge evolution based on the input from the Semantic Web users and tools with an aim to inspect the validity and applicability of knowledge.

New techniques for IE, text analysis, knowledge validation or relationship discovery will surely emerge in the near future. However, the open – web browser and service-based Magpie framework proved to be capable of facilitating the low-cost upgrade to these future technologies without any major re-design of the existing user components, such as ontologies or services. The open architecture of Magpie is thus becoming a bridge to enable a shift from closed, single perspective application development to a smarter, on-demand knowledge construction.

On a similar note, our early experimentation with the PowerMagpie, the successor of the original idea of a Semantic Web browser points to the feasibility of the user interaction with multiple ontologies at the same time. The challenge at this point, which I will address in more depth in the talk at the conference, is to improve our understanding of how different views on the content coming from multiple ontologies affect the interpretation of semantic annotations. For example, with having several ontologies found to be potentially relevant to a single web page, it is not unusual to have mutually inconsistent conceptualizations of a particular term – how can we

(i) communicate this difference to the user, while at the same time (ii) associate terms with appropriate web services in the case where consistency is no longer guaranteed?

This leads to some tentative suggestions regarding the styles of semantically enriched browsing and navigation. One can take the ontology-driven pathway, whereby the text is annotated in a manner that is consistent with a particular core ontology. Other, potentially relevant ontologies would form a contextual fringe; they may only be used, if they do not conflict with the core, only enhance it. On the other hand, we can imagine a more 'lateral navigation' strategy, whereby the users would be enabled to explore the consequences of alternative conceptual commitments with respect to the same, vague terms.

The difference between the two strategies is not in the underlying technologies; the problem is on the level of user interaction – how to convey different meanings using essentially a single channel, which is fairly constrained by its reliance on the standard web browsing tools. We have piloted this exploratory approach to interacting with distributed resources in the domain of supporting learners to interpret complex, interlinked educational materials [14]. Specifically we tested two distinct modes of exploratory learning: (i) convergent, 'spotlight' browsing of semantically enriched resources [5], and (ii) divergent, 'serendipitous' browsing into conceptually related parts of the problem space [3, 12].

Applying Semantic Web to construct multiple exploratory paths and attending to different aspects of the exploration, rather than to the individual nodes of the semantically enriched space, has several side effects. For instance, from the user experience viewpoint, the application becomes more flexible. A semantically enriched application does not confine its user to one specific activity or role. With PowerMagpie, the user would be assisted in both modes of interaction mentioned above, and likely several additional modes would be developed, as we get more understanding of the roles played by different relationships among multiple ontologies.

Another side effect of pursuing a more exploratory support for semantic navigation in PowerMagpie is the dynamics of the semantic application. Ontology-driven solutions are often brittle; often based on closed worlds that enable reasoning solely about the known concepts. Linking the process of concept discovery and the concept association with terms to their presentation using multiple modalities overcomes this brittleness, and also helps to bypass the knowledge acquisition bottleneck. In other words, with the new generation of Semantic Web applications, the concern of the user would be far less on having a good ontology to start with – the ontology would become more fluid, modules and concepts can be added to it based on their *utility* and *usefulness* in particular user interaction situation.

Acknowledgements

The author and his Magpie research have been over the years supported by several projects and research initiatives. In particular, by the climateprediction.net, Dot.Kom, KnowledgeWeb, Advanced Knowledge Technologies (AKT), OpenKnowledge and NeOn pro jects. Climateprediction.net was sponsored by the UK Natural Environment Research Council and UK Department of Trade e-Science Initiative. Dot.Kom

(Designing Adaptive Information Extraction from Text for Knowledge Management) by the IST Framework 5 grant no. IST-2001-34038. KnowledgeWeb is an IST Framework 6 Network of Excellence (grant no. FP6-507482). AKT is an Interdisciplinary Research Collaboration (IRC) sponsored by the UK Engineering and Physical Sciences Research Council by grant no. GR/N15764/01. OpenKnowledge and NeOn are research projects partly supported by grants no. IST-2005-027253 and no. IST-2005-027595, respectively, provided by the European Commission's IST Framework 6 Programme.

References

[1] Adida, B., Birbeck, M.: RDFa Primer 1.0: Embedding RDF in XHTML, World Wide Web Consortium. p. W3C Working Draft (2007)
[2] Berners-Lee, T.: Tabulator: Exploring and analyzing linked data on the Semantic Web. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006. LNCS, vol. 4273, Springer, Heidelberg (2006)
[3] Brusilovsky, P., Rizzo, R.: Map-Based Horizontal Navigation in Educational Hypertext. Journal of Digital Information 3(1), 156 (2002)
[4] Ciravegna, F., Dingli, A., Guthrie, D., et al.: Integrating Information to Bootstrap Information Extraction from Web Sites. In: IJCAI Workshop on Information Integration on the Web, Mexico (2003)
[5] Collins, T., Mulholland, P., Zdrahal, Z.: Semantic Browsing of Digital Collections. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 127–141. Springer, Heidelberg (2005)
[6] Cunningham, H., Maynard, D., Bontcheva, K., et al.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: ACL. 40th Anniversary Meeting of the Association for Computational Linguistics, Pennsylvania, US (2002)
[7] d'Aquin, M., Sabou, M., Dzbor, M., et al.: WATSON: A Gateway for the Semantic Web. In: Posters of the 4th European Semantic Web Conf., Austria (2007)
[8] Ding, L., Finin, T.: Characterizing the Semantic Web on the Web. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 242–257. Springer, Heidelberg (2006)
[9] Domingue, J., Dzbor, M., Motta, E.: Semantic Layering with Magpie. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies in Information Systems, Springer, Heidelberg (2003)
[10] Dzbor, M., Domingue, J., Motta, E.: Magpie: Towards a Semantic Web Browser. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 690–705. Springer, Heidelberg (2003)
[11] Dzbor, M., Motta, E., Domingue, J.: Opening Up Magpie via Semantic Services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 635–649. Springer, Heidelberg (2004)
[12] Dzbor, M., Motta, E., Stutt, A.: Achieving higher-level learning through adaptable Semantic Web aplications. Int. J. of Knowledge and Learning 1(1/2), 25–43 (2005)
[13] Dzbor, M., Motta, E., Buil Aranda, C., et al.: Developing ontologies in OWL: An observational study. In: OWL: Experiences & Directions workshop, Georgia, US (online) (2006)

[14] Dzbor, M., Motta, E.: Semantic Web Technology to Support Learning about the Semantic Web. In: AIED. Proc. of the 13th Intl. Conf. on Artificial Intelligence in Education, IOS Press, California, US (2007)
[15] Dzbor, M., Motta, E., Domingue, J.: Magpie: Experiences in supporting Semantic Web browsing. Journal of Web Semantics 5(3), 204–222 (2007)
[16] Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous semantic web repositories. In: Proc. of the 5th Intl. Semantic Web Conf., Georgia, US (2006)
[17] Hirschman, L., Chinchor, N.: Named Entity Task Definition. In: 7th Message Understanding Conf (MUC-7) (1997)
[18] Hogue, A., Karger, D.R.: Thresher: automating the unwrapping of semantic content from the World Wide Web. In: Proc. of the 14th Intl. WWW Conf., ACM Press, Japan (2005)
[19] Huynh, D., Mazzocchi, S., Karger, D.R.: Piggy Bank: Experience the Semantic Web Inside Your Web Browser. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 413–430. Springer, Heidelberg (2005)
[20] Huynh, D., Karger, D., Miller, R.: Exhibit: Lightweight Structured Data Publishing. In: Proc. of the 16th Intl. WWW Conf., Alberta, Canada (2007)
[21] Kules, B.: User modeling for adaptive and adaptable software systems. UUGuide: Practical design guidelines for Universal Usability 2000 (April 2007), http://www.otal.umd.edu/UUGuide
[22] Mazzocchi, S., Garland, S., Lee, R.: SIMILE: Practical Metadata for the Semantic Web. O'Reilly (2005), http://www.xml.com/pub/a/2005/01/26/simile.html
[23] Quan, D., Huynh, D., Karger, D.R.: Haystack: A Platform for Authoring End User Semantic Web Applications. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 738–753. Springer, Heidelberg (2003)
[24] Quan, D., Karger, D.R.: How to make a semantic web browser. In: Proc. of the 13th International Conference on World Wide Web, ACM Press, New York (2004)
[25] Sabou, R.M., d'Aquin, M., Motta, E.: Using the Semantic Web as Background Knowledge for Ontology Mapping. In: Proc. of the Ontology Mapping wksp (collocated with ISWC 2006) (2006)
[26] Schraefel, M.C., Smith, D.A., Owens, A., et al.: The evolving mSpace platform: leveraging the Semantic Web on the Trail of the Memex. In: Proc. of the Intl. Conf. on Hypertext, Austria (2005)
[27] Shneiderman, B.: Universal Usability: pushing human-computer interaction research to empower every citizen. Communications of the ACM 43(5), 84–91 (2000)
[28] Yee, P., Swearingen, K., Li, K., et al.: Faceted Metadata for Image Search and Browsing. In: CHI. Proc. of the ACM Conf. on Computer-Human Interaction (2003)

On the Hardness of Reoptimization*

Hans-Joachim Böckenhauer, Juraj Hromkovič, Tobias Mömke, and Peter Widmayer

Department of Computer Science, ETH Zurich, Switzerland
{hjb,juraj.hromkovic,tobias.moemke,widmayer}@inf.ethz.ch

Abstract. We consider the following reoptimization scenario: Given an instance of an optimization problem together with an optimal solution, we want to find a high-quality solution for a locally modified instance. The naturally arising question is whether the knowledge of an optimal solution to the unaltered instance can help in solving the locally modified instance. In this paper, we survey some partial answers to this questions: Using some variants of the traveling salesman problem and the Steiner tree problem as examples, we show that the answer to this question depends on the considered problem and the type of local modification and can be totally different: For instance, for some reoptimization variant of the metric TSP, we get a 1.4-approximation improving on the best known approximation ratio of 1.5 for the classical metric TSP. For the Steiner tree problem on graphs with bounded cost function, which is APX-hard in its classical formulation, we even obtain a PTAS for the reoptimization variant. On the other hand, for a variant of TSP, where some vertices have to be visited before a prescribed deadline, we are able to show that the reoptimization problem is exactly as hard to approximate as the original problem.

Keywords: reoptimization, approximation algorithms, inapproximability.

1 Introduction

In algorithmics and operations research, we deal with many optimization problems whose solutions are of importance in everyday applications. Unfortunately, most of these problems are computationally hard, and so we use different approaches such as heuristics, approximation algorithms, and randomization in order to compute good (not necessarily optimal) solutions. Nevertheless, we are often not even able to give any reasonable guarantee on the solution quality or on the efficiency of the applied algorithm. In this difficult situation, we propose to make use of a very natural idea: Don't start from scratch when confronted with a problem, but try to make good use of prior knowledge about similar problem instances whenever they are available. Traditionally, there is no such

* This work was partially supported by SBF grant C 06.0108 as part of the COST 293 (GRAAL) project funded by the European Union.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 50–65, 2008.

prior knowledge because problem inputs are considered as isolated instances. In reality, prior knowledge is often at our disposal, because a problem instance can arise from a small modification of a previous problem instance. As an example, imagine that an optimal timetable (for some objective function, under some constraints) for a given railway network is known, and that now a railway station is closed down. It is intuitively obvious that we should profit somehow from the old timetable when we try to find a new timetable. It is this idea that we pursue in some generality: Given a problem instance with an optimal (or approximate) solution, and a variation of the problem instance that we obtain through small, local modifications, what can we learn about the new solution? Does the old solution help at all? Under what circumstances does it help? How much does it help? How much does it help for the runtime, how much for the quality of the output?

In this paper, we investigate the hardness of solving locally modified problem instances when an optimal (or a good) solution to the original instance is for free. We survey some results showing that this additional information helps to get solutions with better quality guarantees for some problems and does not help to find the solution more easily for other problems. The core of our approach is to investigate which algorithmic problems become easier when moving from the classical problem formulation to the version with locally modified instances, and which don't. Our hopes should not be unrealistic: In general, NP-hardness of a problem implies that the solution to a locally modified problem cannot be found in polynomial time, given the solution of the original problem. But it is, on the other hand, obviously a drastic advantage to know a solution of a problem instance, as compared to not knowing it, and this is reflected in the fact that verification for problems in NP is polynomial, while finding a solution is not (unless P=NP). For polynomial-time solvable problems, such as maintaining a substructure in a graph under local changes (e.g., edge weight changes, insertions, deletions), there are plenty of positive research results.

In this paper, we limit ourselves to hard problems; amazingly, they have been relatively little studied under this natural perspective. Our study aims at contributing to the understanding of the computational hardness of locally modified problems, and can furthermore lead to developing new algorithms for solving algorithmic tasks that are not only efficient, but also provide reasonable guarantees on the solution quality.

The general idea of problem solving through repeated modification is also reflected on a somewhat higher level of generality in the idea of solving a broad variety of problems through an "iterated" approach: In [20], four solutions to very important problems (a polynomial time approximation of the permanent of non-negative matrices; a construction of expander graphs; a log-space algorithm for undirected connectivity; an alternative proof of the PCP theorem) are found to start from a trivial construct, and to apply an ingeniously designed sequence of iterations that yields the desired result by modifying the construct in a moderate manner at each step.

We now define our notion of *reoptimization problems* more technically: Given an instance of an optimization problem together with an optimal solution, we consider the scenario in which the instance is modified locally. In graph problems, e.g., the cost of one edge might be varied or a single vertex or edge may be removed or added, etc. For a problem U and a local modification lm, we denote the resulting problem by lm–U. Obviously, lm–U may be easier than U because we have the optimal solution for the original problem instance for free.

A research question related to reoptimization was also considered in operations research [19,25,26,31,32], where one studies how much a given instance of an optimization problem may be varied if it is desired that optimal solutions to the original instance retain their optimality. In contrast with this so called "postoptimality analysis", we allow also modifications causing the loss of the optimality of the solution to the original instance for the modified instance and look for efficient algorithms for computing new high-quality solutions.

Note that, for some optimization problems, knowing an optimal solution to the original instance trivially makes their reoptimization versions easy to solve because the given optimal solution is itself a very good approximate solution to the modified instance. For example, adding an edge in the instance of a coloring problem can increase the cost of an optimal solution at most by one. But such obvious cases are not a matter of our investigation. They are only useful for illustrating the possible different outcomes of this study. On the one hand, lm–U may become very easy relative to a problem U, either in terms of an additive approximation as for the coloring problem, or it may allow for a PTAS whereas the non-reoptimization variant of the problem is APX-hard. On the other hand, we will show an example where lm–U is exactly as hard as U from the approximability point of view.

To illustrate these possibilities in this paper, we will employ several variants of the traveling salesman problem (TSP) and the Steiner tree problem. The TSP is the problem of finding a minimum-cost Hamiltonian cycle in a complete edge-weighted graph; as local modifications we consider changing the cost of exactly one edge. In the Steiner tree problem, we are given a complete edge-weighted graph and a subset of vertices, called terminals, and the goal is to find a minimum-cost subtree connecting all terminals (and possibly containing some of the non-terminals). For the Steiner tree problem, we consider the change of the terminal set by adding or removing exactly one vertex as local modifications.

The paper is organized as follows: Section 2 contains some basic definitions. In Section 3 we show that, for many optimization problems, the reoptimization variants are still NP-hard. In Section 4 we present a 7/5-approximation algorithm for a reoptimization variant of the metric TSP and 3/2-approximation algorithms for some Steiner tree reoptimization variants. Section 5 contains a PTAS for a reoptimization variant of the Steiner tree problem on graphs with bounded edge weights. In Section 6, we present a reoptimization variant of the TSP with deadlines which is exactly as hard as the non-reoptimization problem. We conclude the paper in Section 7 with some remarks on possible research directions.

2 Problem Definitions

In this section, we will formally define the reoptimization problems which we will use as examples in the subsequent sections of this paper.

We start with the classical traveling salesman problem (TSP). An input instance for the TSP is a complete edge-weighted graph $G = (V, E)$, if the edge cost function $c : E \to \mathbb{Q}^+$ satisfies the *triangle inequality*

$$c(\{u, v\}) \leq c(\{u, w\}) + c(\{w, v\})$$

for all vertices $u, v, w \in V$, we call this instance *metric*. For the TSP, we will consider two different types of local modifications in this paper, namely increasing or decreasing the cost of a single edge.

Definition 1. *The* traveling salesman reoptimization problem with increasing (decreasing, resp.) edge costs*, IncEdge-TSP (DecEdge-TSP, resp.) for short, is the following optimization problem: Given an undirected complete graph $G = (V, E)$ with two edge cost functions $c_O, c_N : E \to \mathbb{Q}^+$, which coincide for all but one edge $e \in E$ where $c_O(e) < c_N(e)$ ($c_O(e) > c_N(e)$, resp.), and a minimum-cost Hamiltonian tour T_O in G according to c_O, find a minimum-cost Hamiltonian tour in G according to c_N.*

The subproblem where both (V, E, c_O) and (V, E, c_N) are metric graphs is denoted by IncEdge-ΔTSP (DecEdge-ΔTSP, respectively).

We will also consider a variant of the TSP where additionally a subset of vertices get assigned deadlines, and a feasible Hamiltonian tour has to visit these deadline vertices before the expiry of their respective deadlines, starting from a prespecified start vertex. For this deadline TSP, we will consider the local modifiations of increasing or decreasing the value of a deadline.

Definition 2. *The* deadline traveling salesman reoptimization problem with increasing (decreasing, resp.) deadline*, IncDead-DLTSP (DecDead-DLTSP, resp.) for short, is the following optimization problem: The input consists of a complete graph $G = (V, E)$ with edge cost function $c : E \to \mathbb{Q}^+$, a set $D \subset V$ of deadline vertices, a start vertex $s \in V$, two deadline assignments $d_O : D \to \mathbb{Q}^+$ and $d_N : D \to \mathbb{Q}^+$ which coincide for all but one deadline vertex v where $d_O(v) < d_N(v)$ ($d_O(v) > d_N(v)$, respectively), a minimum-cost Hamiltonian tour in G obeying the deadline restrictions according to d_O, and a feasible Hamiltonian tour in G with respect to the deadline restrictions d_N. The goal is to find a minimum-cost Hamiltonian tour in G obeying the deadline restrictions d_N.*

The subproblem where (V, E, c) is a metric graph is denoted by IncDead-ΔDLTSP (DecDead-ΔDLTSP, respectively).

Note that, for some deadline TSP instances, even finding a feasible solution might already be a hard task. Since we are not interested in this hardness aspect of the problem here, we assume that a feasible solution also for the new instance with the changed deadline value is given as part of the input.

The last problem which we will consider as an example in this paper is the Steiner tree problem. For the Steiner tree problem, we are given a complete edge-weighted graph and a subset of the vertex set, whose elements are called *terminals*. The goal is to construct a minimum-cost subtree containing all terminals and possibly any of the non-terminal vertices. For the Steiner tree problem, we consider the local modifications of adding a vertex to the set of terminals or removing a vertex from the terminal set.

Definition 3. *The Steiner tree reoptimization problem with increasing (decreasing, resp.) terminal set, IncTerm-STP (DecTerm-STP, resp.) for short, is the following optimization problem: Given a complete graph $G = (V, E)$ with edge cost function $c : E \to \mathbb{Q}^+$, two terminal sets $S_O \subseteq V$ and $S_N \subseteq V$ where $S_O \subset S_N$ and $|S_O| + 1 = |S_N|$ ($S_N \subset S_O$ and $|S_O| - 1 = |S_N|$, resp.), and a minimum-cost Steiner tree T_O for (G, c, S_O), find a minimum-cost Steiner tree T_N for (G, c, S_N). We denote the variant of IncTerm-STP (DecTerm-STP, resp.) where the edge costs are restricted to values from $\{1, 2, \ldots, r\}$ for a constant r by r-IncTerm-STP (r-DecTerm-STP, resp.).*

3 NP-Hardness of Reoptimization

In this section, we show that for the reoptimization problems as defined in Section 2 there is no hope for polynomial-time exact algorithms.

In [4,5], it has been shown that TSP reoptimization in graphs with general cost functions is still as hard to approximate as the classical non-reoptimization variant of general TSP.

Theorem 1. *Unless* P = NP*, there is no polynomial-time $p(n)$-approximation algorithm for IncEdge-TSP or DecEdge-TSP for any polynomial p.* □

The proof of Theorem 1 employs a diamond graph construction similar to the one which was used in [29] to exhibit worst-case examples for local search TSP algorithms.

We will in the following focus on the metric case for the TSP reoptimization. The following result was proved in [4,5].

Theorem 2. *The problem IncEdge-ΔTSP is NP-hard.*

Proof. We use a reduction from the restricted Hamiltonian cycle problem (RHC) which can be described as follows. The input is an unweighted, undirected graph G and a Hamiltonian path P in G which cannot be trivially extended to a Hamiltonian cycle by joining its end-points. The goal is to decide whether G contains a Hamiltonian cycle. This problem is well-known to be NP-complete (see, for example, [24]).

The idea of the reduction is similar to the well-known reduction from the Hamiltonian cycle problem to the TSP. Let (G, P) be an instance of RHC where $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, and $P = (v_1, \ldots, v_n)$. From this RHC instance, we construct an instance for IncEdge-ΔTSP as follows. Let $\tilde{G} = (V, \tilde{E})$ be a

complete graph on vertex set V, and let $c_O(e) = 1$ for all $e \in E \cup \{\{v_n, v_1\}\}$ and $c_O(e) = 2$ otherwise, and let $c_N(\{v_n, v_1\}) = 2$. Let $T_O = v_1, v_2, \ldots, v_n, v_1$ be the given optimal Hamiltonian tour in $\tilde{G}$ according to c_O.

This reduction can obviously be performed in polynomial time, the constructed cost functions are metric, and moreover it is easy to see that G contains a Hamiltonian cycle if and only if $\tilde{G}$ contains a Hamiltonian cycle of cost n, according to c_N. □

In the previous examples, we used the usual Karp reductions. If our aim is to show NP-hardness, however, a polynomial-time Turing reduction is already satisfactory [18]. Using an oracle polynomially many times turns out to be helpful for proving hardness of reoptimization problems. In the following, we present a general framework for proving NP-hardness of reoptimization problems. Then we apply this framework for showing the NP-hardness of two variants of the Steiner tree reoptimization problem (see [12]).

Lemma 1. *Let U be an NP-hard optimization problem and let lm be a local modification for U such that a deterministic algorithm can transform some efficiently solvable input instance I' for U into any input instance I for U using a polynomial number of local modifications of type lm. Then, also the problem lm–U is NP-hard.*

Proof. We reduce U to lm–U using a polynomial-time Turing reduction. Since U is NP-hard, this implies that also lm–U is NP-hard.

Let l be the number of local modifications of type lm needed to transform the instance I' into I. Now suppose that there is a polynomial-time algorithm A for lm–U. Then, applying A exactly $l - 1$ times, starting from I', is sufficient for finding an optimal solution for I. Therefore, both the number of computations and the runtime of each computation are polynomial in the size of U which implies that the reduction runs in polynomial time, too. □

For the problems IncTerm-STP and DecTerm-STP, we are able to show even stronger results by restricting the edge costs to one and two only.

Theorem 3. *The problems IncTerm-STP and DecTerm-STP are strongly NP-hard.*

Proof. We show the stronger result that 2-IncTerm-STP and 2-DecTerm-STP are NP-hard. Since the edge costs are restricted to one and two, according to [18], the NP-hardness implies that these problems are also strongly NP-hard. Obviously, IncTerm-STP and DecTerm-STP are at least as hard as their restricted versions.

Let (G, S, c) be an arbitrary input instance for the problem 2-MinSTP where $G = (V, E)$ and $S = \{s_1, s_2, \ldots, s_l\}$. It is well known that 2-MinSTP is NP-hard (and even APX-hard, see [3]). For the 2-IncTerm-STP, we use the efficiently solvable input instance $(G, \{s_1\}, c)$. The graph $(\{s_1\}, \emptyset)$ is the unique optimal solution. Now we can transform $(G, \{s_1\}, c)$ into (G, S, c) by adding the $l - 1$

vertices s_2 to s_l successively to the terminal set, each time solving an instance of 2-IncTerm-STP. Hence, according to Lemma 1, 2-IncTerm-STP is NP-hard.

Analogously, the problem 2-DecTerm-STP is NP-hard. Here, the instance (G, V, c) is efficiently solvable since we only have to compute a minimum spanning tree. Obviously, successively removing all vertices from $V \setminus S$ from the terminal set is sufficient for applying Lemma 1. □

4 Improving Constant-Factor Approximations

We have seen in the preceding section that we cannot hope for polynomial-time exact algorithms for many reoptimization problems. Thus, we will focus on approximation algorithms in the remainder of this paper. We start with a 1.4-approximation algorithm for IncEdge-ΔTSP and DecEdge-ΔTSP which was presented in [4,5]. This algorithm shows that reoptimization can indeed help for finding approximative solutions since the best known approximation algorithm for the metric TSP is Christofides' algorithm [14] which achieves an approximation ratio of 1.5.

Theorem 4. *There exists a polynomial-time approximation algorithm for both IncEdge-ΔTSP and DecEdge-ΔTSP which achieves an approximation ratio of* $7/5$.

The proof of Theorem 4 is based on the observation that, if both the old and the new cost function are metric, the local change can only be moderate if the edge with changed cost is adjacent to an edge with small cost. More precisely, the following lemma from [4,5] holds.

Lemma 2. *Let $G(V, E)$ be a complete graph, let $c_O, c_N : E \to \mathbb{Q}^+$ be two metric cost functions which coincide, except for one edge e. Then, every edge adjacent to e has a cost of at least $1/2 \cdot |c_O(e) - c_N(e)|$.*

Proof. We first assume that $c_O(e) > c_N(e)$, and let $\delta = c_O(e) - c_N(e)$. Let $f \in E$ be any edge adjacent to e, and for any such f, let $f' \in E$ be the one edge that is adjacent to both e and f. The triangle inequalities according to c_O and c_N then imply $c_O(e) \leq c_O(f) + c_O(f') = c_N(f) + c_N(f')$ and $c_N(f') \leq c_N(f) + c_N(e)$. Hence, $c_O(e) - c_N(e) \leq 2 \cdot c(f)$. The proof in the cost-increasing case is analogous. □

We now prove Theorem 4 for the case of IncEdge-ΔTSP. The proof for DecEdge-ΔTSP is similar and can be found in [4,5].

Proof of Theorem 4. Let (G, c_O, c_N, T_O) be an input instance for IncEdge-ΔTSP. Let e be the edge of G where $c_O(e) < c_N(e)$, let $\delta = c_N(e) - c_O(e)$. We distinguish two cases depending on the size of the edge cost change δ. Let T_N denote one optimal Hamiltonian tour in G according to c_N.

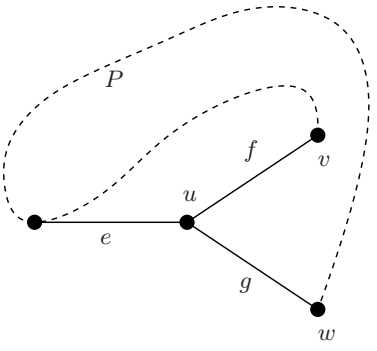


Fig. 1. The structure of an optimal tour according to c_N in case 2 of the proof of Theorem 4

Case 1: Assume $\delta/c_N(T_N) < 2/5$. In this case, already the given old solution T_O constitutes a 7/5-approximation, since

$$\frac{c_N(T_O)}{c_N(T_N)} \leq \frac{c_O(T_O)+\delta}{c_N(T_N)} \leq \frac{c_O(T_N)+\delta}{c_N(T_N)} \leq \frac{c_N(T_N)+\delta}{c_N(T_N)} = 1 + \frac{\delta}{c_N(T_N)} \leq \frac{7}{5}.$$

Case 2: Assume $\delta/c_N(T_N) \geq 2/5$. In this case, we will construct an approximate Hamiltonian tour. We may assume that the edge e is not part of any new optimal solution, otherwise T_O obviously would be optimal also according to c_N. Let T_N denote one arbitrary (but fixed) optimal tour according to c_N, let u denote one of the endpoints of e. Then T_N contains two edges f and g incident to u, let v and w denote the other endpoints of f and g, respectively. By P we denote the subpath of T_N from v to w not containing u. In other words, T_N consists of a path P from v to w and the two edges f and g as shown in Figure 1.

We now consider the following algorithm. For any pair $\tilde{f} = \{\tilde{v}, u\}$ and $\tilde{g} = \{\tilde{w}, u\}$ of edges incident to u, compute an approximate Hamiltonian path on $G - u$ with start vertex $\tilde{v}$ and end vertex $\tilde{w}$ using the 5/3-approximation algorithm from [21,22] and augment it with the edges $\tilde{f}$ and $\tilde{g}$ to yield a Hamiltonian tour for G. The algorithm then outputs the cheapest of these $O(|V|^2)$ tours.

Since all possible pairs $\tilde{f}, \tilde{g}$ of edges are considered, one of the constructed tours contains exactly the same edges f and g as T_N does. Hence, the algorithm constructs a tour of cost at most

$$c(f) + c(g) + \frac{5}{3}c(P) = (c_N(T_N) - c(P)) + \frac{5}{3}c(P) = c_N(T_N) + \frac{2}{3}c(P)$$

and thus achieves an approximation ratio of

$$1 + \frac{2}{3} \cdot \frac{c(P)}{OT_{G_N}}.$$

Due to Lemma 2, $c(f), c(g) \geq \delta/2$ and thus

$$\frac{c(P)}{c_N(T_N)} \leq 1 - \frac{\delta}{c_N(T_N)} \leq \frac{3}{5}.$$

Using this, we achieve an approximation ratio of $1 + 2/5 = 7/5$ also in this case. □

We have seen above that reoptimization can help for the metric subproblem of TSP. All of these results can be generalized to TSP instances satisfying a relaxed form of triangle inequality. For a complete graph $G = (V, E)$ with edge cost function c and for any $\beta > 1/2$, we say that c satisfies the β-triangle inequality if

$$c(\{u, v\} \leq \beta \cdot (c(\{u, w\}) + c(\{w, v\}))$$

for all vertices $u, v, w \in V$. If $\beta > 1$, we speak of a *relaxed triangle inequality*; if $1/2 < \beta < 1$, we speak of a *sharpened triangle inequality*. For the case of a relaxed triangle inequality, approximation algorithms were designed for TSP reoptimization with increasing or decreasing edge costs in [4,5]. These algorithms have constant approximation ratios (depending on the parameter β only) and again improve over the best known approximation algorithms for the non-reoptimization variant of TSP on instances with relaxed triangle inequality.

Also for the local modification of inserting a new vertex into the graph, approximation algorithms improving over Christofides' algorithm have been designed in [1,2]. The best currently known result is a 4/3-approximation as presented in [2].

We proceed with approximation algorithms for Steiner tree reoptimization. The best currently known approximation ratio achievable by an approximation algorithm for the Steiner tree problem is $1 + (\ln 3)/2 \approx 1.55$, see [30]. For the reoptimization variants with the local modification of increasing and decreasing the number of terminals, we present a 1.5-approximation algorithm from [12]. We will focus especially on the algorithm for IncTerm-STP because its time complexity is linear in the number of terminals. Thus, we do not only achieve a better approximation ratio in this case, but also an improved runtime compared to using the algorithm from [30] which iteratively solves minimum spanning trees.

For simplicity, we assume that the cost functions of all instances for IncTerm-STP and DecTerm-STP are metric. This is no restriction because it is sufficient to solve these problems on the metric closure of the input graph, see [12,28]. The metricity allows us to ignore nonterminals of degree two in the given solution since any such vertex can be replaced by connecting its two adjacent vertices directly without increasing the cost.

Theorem 5. *There exists a 1.5-approximation algorithm for IncTerm-STP, and its runtime is linear in the number of terminals.*

Proof. Consider the following algorithm A. If the new terminal t is contained in the given optimal Steiner tree T_O, then A keeps the old solution. Otherwise, A adds the cheapest edge that connects T_O with t.

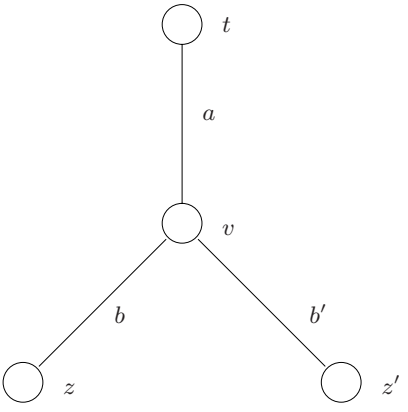


Fig. 2. The situation in the proof of Theorem 5 where the degree of t is one and the neighbor of t is a non-terminal

Now we show that the approximation ratio of A is at most 1.5. If t is in T_O, then we get an optimal solution for the new instance since any new optimal tree T_N cannot be cheaper than T_O. In the case where the added edge is cheaper than $c(T_O)/2$, the claimed approximation ratio can also be easily guaranteed. Since T_N is at least as expensive as T_O, the cost of the computed solution, which is at most $1.5 \cdot c(T_O)$, suffices for a 1.5 approximation. Hence, in the following we assume that t is not in T_O and that any cheapest edge from t to a terminal costs at least $c(T_O)/2$. Let T_N denote an optimal tree for the new instance; whenever it is possible to get a new optimal tree by augmenting T_O with an additional edge from t, let T_N be of this kind.

For now, let us assume that the degree of t in T_N is one; afterwards we will show that higher degrees of t are even easier to deal with.

Let $a = \{t, v\}$ be the only edge that is incident to t in T_N. If v is a terminal, then, due to the optimality of the subtree without t (see [27]), A computes an optimal solution. Hence, we assume v to be a nonterminal. We further assume that the degree of v is at least three since we can replace every nonterminal of degree two by an edge connecting the adjacent vertices.

Since each leaf of an optimal Steiner tree is a terminal, in T_N there are two terminals z and z' such that a path b connects v to z and a path b' connects v to z'. Note that, due to the metricity, the paths b and b' are at least as expensive as the edges $\{v, z\}$ and $\{v, z'\}$ (see Figure 2).

Since $T_N - a$ is a solution for (G, S_O, c),

$$c(T_N) \geq c(T_O) + c(a). \tag{1}$$

Both $a + b$ and $a + b'$ connect t with a terminal. Let T_A be the Steiner tree computed by A. Then

$$c(T_A) \leq c(T_O) + c(a) + c(b). \tag{2}$$

Let us assume that, without loss of generality, $c(b) \leq c(b')$. Since $c(T_N) \geq c(b) + c(b')$, we know that

$$c(b) \leq c(T_N)/2, \tag{3}$$

and thus we get

$$c(T_A) \underset{(2)}{\leq} c(T_O) + c(a) + c(b) \underset{(1)}{\leq} c(T_N) + c(b) \underset{(3)}{\leq} 3c(T_N)/2,$$

proving that T_A is a 1.5-approximative solution.

If the degree of t is at least two, we can use the same proof as above, but with $c(a) = 0$, i.e.,

$$c(T_A) \leq c(T_O) + c(b) \leq c(T_N) + c(b) \leq 3c(T_N)/2.$$

It is easy to verify that the runtime of A is linear in the number of terminals. □

As mentioned above, the knowledge of an optimal solution also helps us to find an approximative solution in the case where a terminal is declared to become a nonterminal.

Theorem 6. *There is a 1.5-approximation algorithm for DecTerm-STP.*

For the 1.5-approximation algorithm, the degree of the altered vertex t is crucial. If the degree of t is at least three, then the old optimal tree provides a good approximation. The hardest case appears to be the one where the degree of t is two. The main idea here is to remove the shortest path from t to a terminal in each of the two subtrees that are connected by t. Afterwards, the algorithm computes an optimal Steiner tree for connecting the components that were created by removing the paths. The branching of the two subtrees ensures that the length of the paths is logarithmic in the number of vertices, which allows the algorithm to run in polynomial time.

If the degree of t is one, removing t and the only corresponding edge either yields an optimal solution for the reoptimization problem, if t is adjacent to a terminal, or it allows us to apply one of the first two cases to compute an approximative solution.

5 Polynomial-Time Approximation Schemes

In the preceding section, we have seen that the concept of reoptimization can help in lowering a constant approximation ratio. In this section, we will present two examples, where the improvement is even larger. More precisely, we will present APX-hard problems whose reoptimization variants allow for a polynomial-time approximation scheme (PTAS).

Theorem 7. *Let r be an arbitrary positive integer. There exist polynomial-time approximation schemes for r-IncTerm-STP and r-DecTerm-STP.*

Proof. We have to show that, for every $\varepsilon > 0$, there is a polynomial-time $(1+\varepsilon)$-approximation algorithm for both r-IncTerm-STP and r-DecTerm-STP. For this purpose, we consider the following algorithm A.

Let $k = \lceil 1/\varepsilon \rceil$. If the number of terminals in the new input instance, $|S_N|$, is at most $r \cdot k$, then A computes an optimum solution for r-IncTerm-STP or r-DecTerm-STP, respectively, using the Dreyfus-Wagner algorithm [16]. Otherwise, for r-DecTerm-STP, A keeps the given old solution; for r-IncTerm-STP, A keeps the old solution and, if the new terminal is not contained in the old solution, A adds an arbitrary edge that connects the old tree with the new terminal.

In the following we assume without loss of generality that the input instance has more than $k \cdot r$ terminals, i.e., $|S_N| > r \cdot k$, since otherwise A computes an optimal solution.

Therefore, an optimal solution contains at least $r \cdot k$ edges, each of cost at least one.

Adding one edge to some solution increases its costs by at most r. Hence, for IncTerm-STP, $c(T_A) \leq c(T_O) + r$ and the approximation ratio of A is at most

$$\frac{c(T_A)}{c(T_N)} \leq \frac{c(T_O) + r}{c(T_N)} \leq \frac{c(T_N) + r}{c(T_N)} = 1 + \frac{r}{c(T_N)} \leq 1 + \frac{r}{r \cdot k} \leq 1 + \varepsilon.$$

For r-DecTerm-STP, adding one edge to T_N yields a feasible solution for (G, S_O, c). Hence, we know that $c(T_O) \leq c(T_N) + r$ and the approximation ratio of A is

$$\frac{c(T_A)}{c(T_N)} \leq \frac{c(T_N) + r}{c(T_N)} = 1 + \frac{r}{c(T_N)} \leq 1 + \frac{r}{r \cdot k} \leq 1 + \varepsilon.$$

For both problems, whenever the number of terminals $|S_N|$ is at most $k \cdot r$, the time complexity of A is bounded from above by $O(n^2 \cdot 3^{r \cdot k})$ — the time to compute an optimal Steiner tree using the Dreyfus-Wagner algorithm (see [16]) — which is only exponential in the constants $\lceil 1/\varepsilon \rceil$ and r. The remaining parts of the algorithm only require constant time complexity. Hence, all requirements for a PTAS are satisfied. □

A similar result has been proven in [5] for TSP reoptimization with increasing or decreasing edge costs on instances satisfying some sharpened triangle inequality. Moreover, a PTAS for a reoptimization variant of the metric maximum TSP, where the local modification is the insertion of a new vertex into the graph, was presented in [2].

6 Approximation Hardness of Reoptimization

In the preceding sections, we have seen some examples of problems where the reoptimization approach helps to achieve a better approximation guarantee. In this section, we will demonstrate that there are also problems for which reoptimization does not help at all. As an example, we will use the reoptimization variant of the TSP with deadlines as presented in Definition 2.

We start with the subproblem where the number of deadline vertices is bounded by a constant. The classical non-reoptimization variant k-ΔDLTSP of this problem with metric edge cost function and a constant number k of deadline vertices was shown to be 2.5-approximable in [11,10]. Furthermore, for every small $\varepsilon > 0$, a lower bound of $2 - \varepsilon$ on the approximability of k-ΔDLTSP was

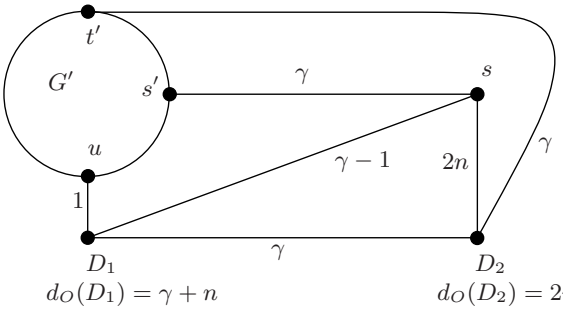


Fig. 3. The construction in the proof of Theorem 8. All vertices $v \in V' - \{s', t'\}$ are connected like u.

proven in [11,10]. We will in the following present a result from [4] showing that exactly the same lower bound also holds for k-IncDead-ΔDLTSP and k-DecDead-ΔDLTSP.

Theorem 8. *Let $\varepsilon > 0$. There is no polynomial-time $(2-\varepsilon)$-approximation algorithm for k-IncDead-ΔDLTSP or k-DecDead-ΔDLTSP, unless* P = NP.

Proof. We prove the claim for 2-IncDead-ΔDLTSP only. The proof for 2-DecDead-ΔDLTSP is similar and can be found in [4].

For the proof, we use a reduction from the restricted Hamiltonian path problem (RHP) which is defined as follows. The input is an unweighted, undirected graph G and a Hamiltonian path P in G with start vertex s and end vertex t. The goal is to decide whether G contains a Hamiltonian from s to some vertex $u \neq t$. This problem is well-known to be NP-complete (see, for example, [24]).

Let $\varepsilon > 0$. We show that the RHP could be solved using a $(2-\varepsilon)$-approximation algorithm for k-IncDead-ΔDLTSP.

Let (G', P') be an input instance for RHP where $G' = (V', E')$ is an unweighted undirected graph, $|V'| = n + 1$, and P' is a Hamiltonian path in G' from start vertex $s' \in V'$ to end vertex $t' \in V'$.

We choose a $\gamma > (5n+3)/(2\varepsilon)$; this implies $(4\gamma+n-1)/(2\gamma+3n+1) > 2-\varepsilon$.

Now we construct a complete weighted graph $G = (V, E)$ with edge cost function c as part of an input instance for 1-IncDead-ΔDLTSP as shown in Figure 3. We add three new vertices s, D_1, and D_2 to V', i.e., $V = V' \cup \{s, D_1, D_2\}$, and, for any edge e between two vertices $v_1, v_2 \in V'$, we set $c(e) = 1$ if $e \in E'$ and $c(e) = 2$ otherwise. All edges depicted in Figure 3 have the indicated costs while non-depicted edges obtain maximal possible costs such that the triangle inequality is still satisfied. The only deadline vertices are D_1 and D_2, We set the deadlines $d_O(D_1) = \gamma + n$ and $d_O(D_2) = 2\gamma + n + 1$. The given optimal solution for this ΔDLTSP instance is the tour $T_O = s, D_1, D_2, t', \ldots, s', s$ which uses the Hamiltonian path P' from s' to t' in G'.

The cost of this given solution is exactly $\gamma-1+\gamma+\gamma+n+\gamma = 4\gamma+n-1$. Any solution not using such Hamiltonian path from s' to t' may visit some vertices in V' between s and D_1, but costs at least the amount of 1 more.

Moreover, we define the new deadline function d_N by increasing the deadline of D_1 by the amount of x, i.e., $d_N(D_1) = d_O(D_1) + x$. If the graph G' contains a Hamiltonian path P from s' to some vertex $u \neq t'$, an optimal solution for G with respect to d_N is the tour s, P, D_1, D_2, s which costs $\gamma+n+1+\gamma+2n = 2\gamma+3n+1$.

On the other hand, if G' does not contain such a Hamiltonian path, it is not possible to visit all vertices in V' before reaching D_1 and D_2. As $c(\{t', D_1\}) \geq 2$, we cannot follow the given Hamiltonian path P because this would violate the deadline $d(D_2)$. Similar arguments hold for every other possibility. Hence, the given tour T_O remains an optimal solution also with respect to d_N in this case.

Thus, we could use any approximation algorithm with an approximation guarantee better than

$$\frac{4\gamma + n - 1}{2\gamma + 3n + 1} > 2 - \varepsilon$$

to solve the RHP. Hence, approximating 2-IncDead-ΔDLTSP within a ratio of $2 - \varepsilon$ is NP-hard which obviously implies the same lower bound also for every $k \geq 2$. □

Note that we have actually proven even a stronger result than claimed in Theorem 8: The claimed lower bounds even hold for problem instances containing only two deadline vertices.

In the more general case of an unbounded number of deadline vertices, even a linear lower bound on the approximation ratio was shown for ΔDLTSP in [11,10]. Also this lower bound carries over to the reoptimization case as shown in [4].

Theorem 9. *Let $\varepsilon > 0$. There is no polynomial-time $O(|V|)$-approximation algorithm for IncDead-ΔDLTSP or DecDead-ΔDLTSP, unless* P = NP.

The reduction needed for proving Theorem 9 is technically rather involved, so we refer to [4] for the proof.

Similar results as in Theorems 8 and 9 can also be shown for the local modification of increasing or decreasing the cost of a single edge instead of changing a deadline.

7 Conclusion

In this paper, we have presented an overview of the current techniques and results in the area of approximation algorithms for reoptimization problems. We have shown that the concept of reoptimization can in many cases help to improve the approximability of hard optimization problems by lowering constant approximation ratios or even constructing a PTAS for problems which are APX-hard in their classical non-reoptimization variant. On the other hand, we have exhibited some problems where reoptimization does not help at all.

But there are also several interesting open questions in this area of research. For example, the approximation hardness of all reoptimization problems presented in Section 4 is open. Moreover, the model of reoptimization considered so far is rather restricted and could be generalized in two ways: First, instead of

requiring an exact solution to the unaltered instance as part of the input, a more advanced model of reoptimization could also consider a good approximative solution here. Closely connected with this is a second possible way of generalizing the model by allowing for not only one local modification of the input instance, but for a sequence of local modifications. First results in this direction were presented in [2] for TSP reoptimization with insertion of new vertices as local modification.

References

1. Archetti, C., Bertazzi, L., Speranza, M.G.: Reoptimizing the traveling salesman problem. Networks 42, 154–159 (2003)
2. Ausiello, G., Escoffier, B., Monnot, J., Paschos, V.Th.: Reoptimization of minimum and maximum traveling salesman's tours. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 196–207. Springer, Heidelberg (2006)
3. Bern, M.W., Plassmann, P.E.: The Steiner problem with edge lengths 1 and 2. Information Processing Letters 32(4), 171–176 (1989)
4. Böckenhauer, H.-J., Forlizzi, L., Hromkovič, J., Kneis, J., Kupke, J., Proietti, G., Widmayer, P.: Reusing optimal TSP solutions for locally modified input instances (extended abstract). In: IFIP TCS 2006. Proc. of the 4th IFIP International Conference on Theoretical Computer Science, pp. 251–270. Springer, Norwell (2006)
5. Böckenhauer, H.-J., Forlizzi, L., Hromkovič, J., Kneis, J., Kupke, J., Proietti, G., Widmayer, P.: On the approximability of TSP on local modifications of optimally solved instances. Algorithmic Operations Research 2(2), 83–93 (2007)
6. Böckenhauer, H.-J., Hromkovič, J., Klasing, R., Seibert, S., Unger, W.: Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem (extended abstract). In: Bongiovanni, G., Petreschi, R., Gambosi, G. (eds.) CIAC 2000. LNCS, vol. 1767, pp. 72–86. Springer, Heidelberg (2000)
7. Böckenhauer, H.-J., Hromkovič, J., Klasing, R., Seibert, S., Unger, W.: Approximation algorithms for TSP with sharpened triangle inequality. Information Processing Letters 75, 133–138 (2000)
8. Böckenhauer, H.-J., Hromkovič, J., Klasing, R., Seibert, S., Unger, W.: An improved lower bound on the approximability of metric TSP and approximation algorithms for the TSP with sharpened triangle inequality (extended abstract). In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, pp. 382–394. Springer, Heidelberg (2000)
9. Böckenhauer, H.-J., Hromkovič, J., Klasing, R., Seibert, S., Unger, W.: Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. Theoretical Computer Science 285, 3–24 (2002)
10. Böckenhauer, H.-J., Hromkovič, J., Kneis, J., Kupke, J.: On the parameterized approximability of TSP with deadlines. Theory of Computing Systems (to appear)
11. Böckenhauer, H.-J., Hromkovič, J., Kneis, J., Kupke, J.: On the approximation hardness of some generalizations of TSP. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 184–195. Springer, Heidelberg (2006)
12. Böckenhauer, H.-J., Hromkovič, J., Královič, R., Mömke, T., Rossmanith, P.: Reoptimization of Steiner trees: changing the terminal set (submitted)
13. Böckenhauer, H.-J., Seibert, S.: Improved lower bounds on the approximability of the traveling salesman problem. RAIRO Theoretical Informatics and Applications 34, 213–255 (2000)

14. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh (1976)
15. Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F.: VRP with time windows. In: Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem, SIAM 2001, pp. 157–193 (2001)
16. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks 1, 195–207 (1971/72)
17. Forlizzi, L., Hromkovič, J., Proietti, G., Seibert, S.: On the stability of approximation for Hamiltonian path problems. Algorithmic Operations Research 1(1), 31–45 (2006)
18. Garey, M., Johnson, D.: Computers and Intractability. W. H. Freeman and Co., New York (1979)
19. Greenberg, H.: An annotated bibliography for post-solution analysis in mixed integer and combinatorial optimization. In: Woodruff, D.L. (ed.) Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search, pp. 97–148. Kluwer Academic Publishers, Dordrecht (1998)
20. Goldreich, O.: Bravely, moderately - A common theme in four recent works. In: SIGACT News, vol. 37, pp. 31–46. ACM, New York (2006)
21. Guttmann-Beck, N., Hassin, R., Khuller, S., Raghavachari, B.: Approximation algorithms with bounded performance guarantees for the clustered traveling salesman problem. Algorithmica 28, 422–437 (2000)
22. Hoogeveen, J.A.: Analysis of Christofides' heuristic: Some paths are more difficult than cycles. Operations Research Letters 10, 178–193 (1978)
23. Hromkovič, J.: Stability of approximation algorithms for hard optimization problems. In: Bartosek, M., Tel, G., Pavelka, J. (eds.) SOFSEM 1999. LNCS, vol. 1725, pp. 29–47. Springer, Heidelberg (1999)
24. Hromkovič, J.: Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Springer, Heidelberg (2003)
25. Libura, M.: Sensitivity analysis for minimum Hamiltonian path and traveling salesman problems. Discrete Applied Mathematics 30, 197–211 (1991)
26. Libura, M., van der Poort, E.S., Sierksma, G., van der Veen, J.A.A.: Stability aspects of the traveling salesman problem based on k-best solutions. Discrete Applied Mathematics 87, 159–185 (1998)
27. Mölle, D., Richter, S., Rossmanith, P.: A faster algorithm for the Steiner tree problem. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 561–570. Springer, Heidelberg (2006)
28. Prömel, H.J., Steger, A.: The Steiner Tree Problem. Friedr. Vieweg & Sohn, Braunschweig (2002)
29. Papadimitriou, Ch., Steiglitz, K.: Some examples of difficult traveling salesman problems. Operations Research 26, 434–443 (1978)
30. Robins, G., Zelikovsky, A.: Improved Steiner tree approximation in graphs. In: Proc. of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 770–779. ACM, New York (2000)
31. Sotskov, Y.N., Leontev, V.K., Gordeev, E.N.: Some concepts of stability analysis in combinatorial optimization. Discrete Appl. Math. 58, 169–190 (1995)
32. Van Hoesel, S., Wagelmans, A.: On the complexity of postoptimality analysis of 0/1 programs. Discrete Applied Mathematics 91, 251–263 (1999)

Describing Self-assembly of Nanostructures

Natasha Jonoska* and Gregory L. McColm

Department of Mathematics
University of South Florida
Tampa, FL 33620
{jonoska,mccolm}@math.usf.edu

Abstract. We outline an algebraic model for describing complex structures obtained through self-assembly of molecular building blocks and show that, with this model, the assembled structure can be associated to a language and it can be determined up to congruence.

Recently models for self-assembly have been proposed for describing various phenomena ranging from nano-scale structures, material design, crystals, bio-molecular cages such as viral capsids and for computing (see for example [3,4,7,8]). There is an increased necessity for mathematical study of these phenomena. With this abstract we sketch an informal description of an algebraic system for describing and characterizing nanostructures built by a set of molecular building blocks. This algebraic approach connects the classifcal view of crystal dissection with a more modern system based on algebraic automata theory. A two dimensional treatment of the same problem with a more formal description can be found in [5]. Here we concentrate on a simple three-dimensional case with polyhedral building blocks.

1 The General Set-Up

For simplicity we take that the molecular building blocks can be modeled as a set of polyhedra, for example a pyramid and a cube (see Figure 1). The building blocks have specific chemical properties on their faces (presented as labels or colors). These bonds may be strong covalent or weak ionic (hydrogen) types of bonds and specify which two faces can be superimposed or "glued" together. Say, the connection or bonding is allowed only along compatible faces, where the "compatibility" is defined by a binary relation on the set of bond types.

We concentrate on two general, essentially geometric problems,

- how can structures that can be built characterized or classified?
- how can two non-congruent structures be distinguished?

In this note, we outline a method for answering the latter question.

For these questions, one utilizes classical symmetry isometries: suppose that the cube was taken from an archetypic "cube type" in "standard position"

* Supported in part by the NSF grants CCF-0726396 and CCF-0523928.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 66–73, 2008.

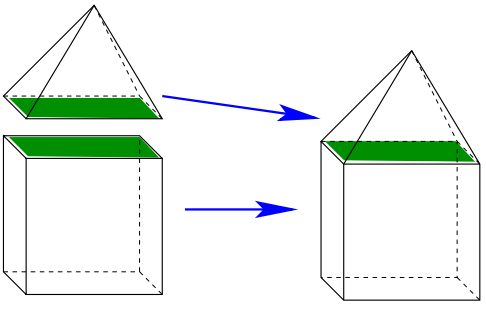


Fig. 1. An example of assembly of two polyhedral blocks together by bonding across compatible faces. In this example, two faces are compatible if they are congruent and of the same color.

(in a particular orientation, centered at the origin). If this block is used in an assembly of a complex structure, it will be displaced by a vector $\mathbf{x}$ and then rotated by an angle θ about a line ℓ. Its new placement can be described by the triple $(\mathbf{x}, \ell, \theta)$. Attaching a pyramid across a face of the cube requires it to be displaced and rotated as in Figure 2. All block types that take part in assembly of the structure are assumed to be taken from a standard position (centered at the origin in a particular orientation) and then displaced to their appropriate position in the built structure. For every assembled structure we assume that there is a building block which is in its standard position which we call a *reference block*.

We also assume that there is a finite set of building block types, there are a finite number of bond types, and a finite number of relative rotations that one block can take with respect to another while bonding.

It is desirable that a computer program be used to design and investigate these structures. For this purpose one needs to convert the geometry into a compact (algebraic) system for representing the geometric articulation. We propose to do this by using finite state automata and borrowing a notion from turtle geometry (from the "turtle graphics" of the computer language Logo developed by Wally Feurzeig and Seymour Papert; see, e.g., [1]).

2 Automata

A structure can be described by knowing the positions of all building blocks, and those positions are known if a path that a "bug" could take from the reference block to each block in the structure. Hence, we can describe a structure (composed of building blocks) by describing all the paths that a bug could take to traverse it. The bug starts at the center of the reference block, and moves from block to block across bonds. Its motion is tracked by the vectors from the center of a block to the center of the next block. During this motion, the bug keeps track of the re-orientations it needs to make arriving at a block: orienting itself, and then moving to the next block. Each re-orientation is accomplished

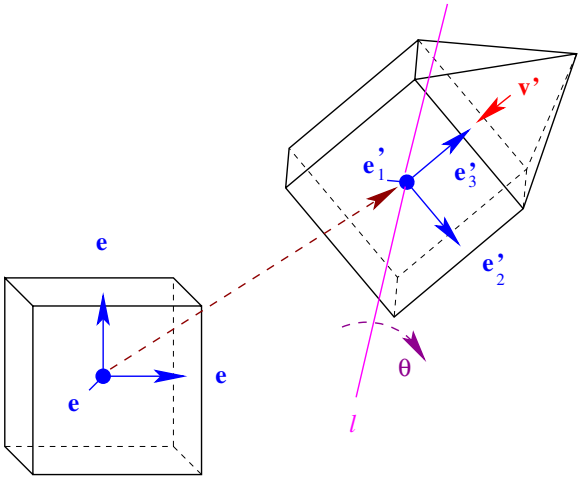


Fig. 2. Displacing the cube by the vector $\mathbf{x}$ and rotating by angle θ around an oriented line ℓ allows connection the pyramid's base to the cube's top face. If $\mathbf{e}_3$ is the vector from the center of the cube (in standard position) to the center of the top face (which we treat as the "reference point" for the top face), then the corresponding vector in the displaced and rotated cube is $\mathbf{e}'_3 = \mathrm{Rot}_{\ell,\theta}(\mathbf{e}_3)$ where $\mathrm{Rot}_{\ell,\theta}$ is the corresponding matrix for the given transformation. If the vector from the center of the pyramid to its base is denoted $\mathbf{v}$ (when in standard position), after rotation, that vector becomes $\mathbf{v}' = \mathrm{Rot}_{\ell,\theta}(\mathbf{v})$. Hence after displacing, the center of the pyramid is at $\mathbf{x} + \mathbf{e}'_3 - \mathbf{v}'$, and its orientation is determined by (ℓ, θ).

by a single rotation around a line (from a fixed list of such orientations) relative to the block it is on.

A finite state automaton can be used to generate the possible bug paths for a given set of block types within an assembled structure. The states of the automaton are the block types: in the example of Figure 1, denote for the state representing the cube block type and $\triangle$ for the state representing pyramid block type. The automaton also has a zero state 0 to represent a state of the bug having attempted an illegal move. All states of the automaton are initial and terminal. The input alphabet of the automaton consists of all possible movements a bug can make from one block (of a specific type) to another block (of a specific type). The walks follow the bonding properties of the abutting faces. For example, a transition from a block $\mathbf{b}$ to a block $\mathbf{b}'$ labeled by symbol $\sigma = (\mathbf{u}, \imath, \varphi)$ represents the following movement of the bug. From a block $\mathbf{b}$, rotate through an angle φ about a line $\imath$ (for $\imath$ displaced to go through the center of $\mathbf{b}$) and translate the bug through the vector $\mathbf{u}$ to a new block $\mathbf{b}'$. To a walk of the bug within the structure we associate a walk in the directed graph represented by the finite state automaton. At every step of the walk, we associate a *placement* of the bug. If the bug starts a walk in the structure at block $\mathbf{b}$, then its placement at the starting time is at the origin, and the block $\mathbf{b}$ is considered to be in the standard position and is the "reference block". Now, inductively, if the bug is on a block $\mathbf{b}$

whose center is at $\mathbf{x}$, and if that block was rotated from its standard orientation through an angle θ about the oriented line ℓ, we say that its *placement* is at block $\mathbf{b}$ with position $(\mathbf{x}, \ell, \theta)$. A transition moves the bug from one placement to another. For example, if the bug is at placement $(\mathbf{x}, \ell, \theta)$ at block $\mathbf{b}$ (of type $\mathfrak{b}$), an automaton transition from state $\mathfrak{b}$ to state $\mathfrak{b}'$ labeled $\sigma = (\mathbf{u}, \imath, \varphi)$ moves the bug to a block $\mathbf{b}'$ with position $(\mathbf{x} + \mathrm{Rot}_{\ell,\theta}(\mathbf{u}), \jmath, \psi)$, where $\mathrm{Rot}_{\ell,\theta}$ is the rotation operator (for the rotation about ℓ by angle θ) and $\mathrm{Rot}_{\jmath,\psi} = \mathrm{Rot}_{\imath,\varphi} \circ \mathrm{Rot}_{\ell,\theta}$ as computed by the Euler-Oliveires formulas (see, e.g., [6], (4.3), p. 82]). Hence a path in the automaton $\mathfrak{b}\sigma_1\sigma_2\cdots\sigma_k\mathfrak{b}'$ starting at block type (state) $\mathfrak{b}$ and ending at block type (state) corresponds to a path in the assembled structure starting at $\mathbf{b}_0$ (of type $\mathfrak{b}$) following displacements defined with σ_1, σ_2, ..., and σ_k, and ending at block $\mathbf{b}_k$ (of type $\mathfrak{b}'$). Such a path in the automaton (or in the structure) represents a composition of translations and rotations moving the bug from block $\mathbf{b}_0$ to block $\mathbf{b}_k$.

This automaton is *symmetric* in the sense that a bug can always reverse direction: for any transition σ representing a movement from a block of type $\mathfrak{b}$ to an adjacent block of type $\mathfrak{b}'$, there is a "local inverse" σ^{-1} representing a bug movement from a block of type $\mathfrak{b}'$ to a block of type $\mathfrak{b}$.

Hence, given a structure made of building blocks, any traversal of the structure by the bug can be represented by a "framed walk" $\mathfrak{b}\sigma_1\sigma_2\cdots\sigma_n\mathfrak{b}'$, where $\mathfrak{b}$ and $\mathfrak{b}'$ encode the types of the initial and terminal blocks, and $\sigma_1, \ldots, \sigma_n$ encode n movements by the bug. Note that the walks do not encode particular blocks, just their types: transfixing (identifying or distinguishing) between blocks requires an additional notion. In this sense, a framed walk can be seen as a walk through the automaton's digraph.

Even for cubes and pyramids, if we allowed all possible transitions across abutting square faces, the computational details could be burdensome. For example, if the bug started at a cube in standard position, and moved to an adjacent block, there are 6 faces of the cube (hence six bonding sites), and for each face, there are two possible next blocks: the cube (via any of the six faces, and for each choice of articulating faces, there are four orientations of the adjacent cube) and the pyramid (one face, with four orientations): thus there are $6 \times 7 \times 4 = 168$ possible adjacencies, and so 168 possible transitions (even though in any particular structure, there will be at most six possible transitions from this initial cube).

For simplicity, we restrict the example to only one building block, a cube, where the only bonding is Left-Right, Up-Down, and Back-Front, and hence each complex formed has each block in the same orientation, as shown in Figure 3. So the bug moving from cube to cube can only move from a Right face of one cube to the (abutting) *L*eft face of an adjacent cube (a transition labeled R), or from the *F*ront face to the *B*ack face of an adjacent cube (a transition labeled F), or from the *U*pper face to the *D*own face of an adjacent cube (a transition labeled F),and so on. In this example, there are only six possible transitions, and the resulting automaton is presented in Figure 3. The transitions labeled $L = R^{-1}$, $B = F^{-1}$, and $D = U^{-1}$ are (local) inverses of R, F, and U, respectively. We

have the following correspondence of the symbols with the geometric movements of the bug: $F = (\langle 1,0,0\rangle, \ell, 0)$, $F = (\langle 0,1,0\rangle, \ell, 0)$, and $U = (\langle 0,0,1\rangle, \ell, 0)$, for any ℓ. And thus $B = F^{-1} = (\langle -1,0,0\rangle, \ell, 0)$, and so on.

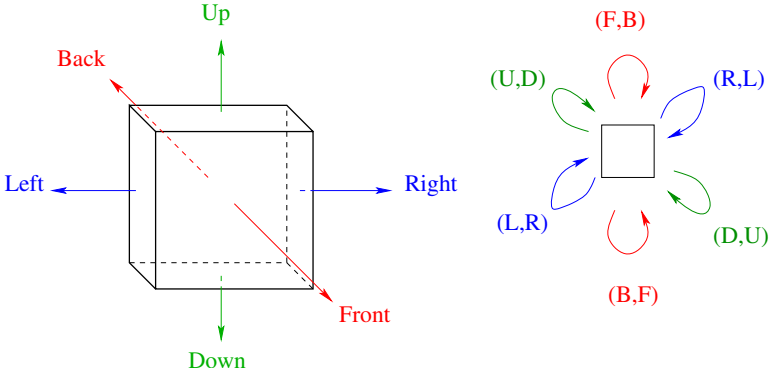


Fig. 3. To the left, a cube in standard position, with its faces labeled; a cube is allowed to connect to its neighbor only via a complementary face, as dictated by the automaton represented by the digraph to the right. For example, L is an instruction: "go through the Left face of the current block into the next block, entering that next block via its Right face." On a particular block in a particular assembled structure, the bug can obey this instruction *if and only if* there is a next block beyond the Left face of the current block, and the Right face of that next block abuts the Left face of the current block.

So we start with a fixed set of *block types* (cubes, pyramids, etc.) that can connect in certain ways defined by transitions of some automaton. These transitions can be concatenated into words. Given such an automaton, call the set of all words it can generate its *walkspace*. In order to identify structures that can be built from building blocks, we assume that a bug is traversing a particular structure, (using a framed walk as a list of instructions to follow). Hence, we look at particular structures through the walks of the bug.

3 Structures through Words

Intuitively, a particular structure is an assembly of blocks (each one of a given block type), all arranged so that they are connected across articulating faces.

A framed walk is a list of instructions for walking through a structure, with the following complication: it may not be possible for the bug to follow the instructions. For example, imagine a structure of five cubes from Figure 3, lined up and articulating Right face to Left face in a row. From the leftmost cube, but not from the middle cube, the bug could perform the walk $RRRRR$; but there is no cube from which it could perform the walk FRB. In the rest of the section, intuitively we take that "structure" is a possible assembled structure

that can be obtained by gluing building blocks of given types according to the bonding relation.

Definition 1. *Let* $\mathfrak{C}$ *be a structure, and let* $\mathbf{b}$ *be a block of* $\mathfrak{C}$. *Let* $W(\mathfrak{C}, \mathbf{b})$ *be the set of framed walks in the automaton that can be walked in* $\mathfrak{C}$ *starting at* $\mathbf{b}$ *as a reference block. Furthermore, the* length *of a framed walk is the number of transitions, i.e.,* $|\mathfrak{b}\sigma_1\sigma_2\cdots\sigma_n\mathfrak{b}'| = n$. *Let* $W_n(\mathfrak{C}, \mathbf{b}) = \{w \in W(\mathfrak{C}, \mathbf{b}) \colon |w| \leq n\}$.

We now convert geometry to algebra by defining an equivalence that identifies pairs of framed walks that lead the bug from the same initial block to the same terminal block. Let $\mathfrak{b}_0\mathbf{s}_1\mathfrak{b} \sim \mathfrak{b}_0\mathbf{s}_2\mathfrak{b}$ mean that if the bug started at a block of type $\mathfrak{b}_0$ (in standard position) and followed the walk instructions $\mathbf{s}_1$, it would wind up in the same position and orientation, *and hence the same block* as if it followed the walk instructions $\mathbf{s}_2$. We define a structure to be *rigid with respect to* $\sim$ if for any three blocks $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ of the structure, where $\text{type}(\mathbf{b}_0) = \mathfrak{b}_0$ and $\text{type}(\mathbf{b}_1) = \text{type}(\mathbf{b}_2) = \mathfrak{b}$, the following is true: $\mathbf{b}_1 = \mathbf{b}_2$ if and only if for any framed walk $\mathfrak{b}_0\mathbf{s}_1\mathfrak{b}$ that guides the bug from $\mathbf{b}_0$ to $\mathbf{b}_1$ and for any framed walk $\mathfrak{b}_0\mathbf{s}_2\mathfrak{b}$ that guides the bug from $\mathbf{b}_0$ to $\mathbf{b}_2$, $\mathfrak{b}_0\mathbf{s}_1\mathfrak{b} \sim \mathfrak{b}_0\mathbf{s}_2\mathfrak{b}$. Notice that $\sim$ depends on the shapes of the blocks, and on the geometric space in which the blocks are placed. This relation $\sim$ is defined through cycles in the walkspace, i.e., by identifying those framed walks that in the geometry lead the bug back to the block on which it started (see Figure 4). For a set of walks W denote with $\hat{W}$ the equivalence classes of walks in W.

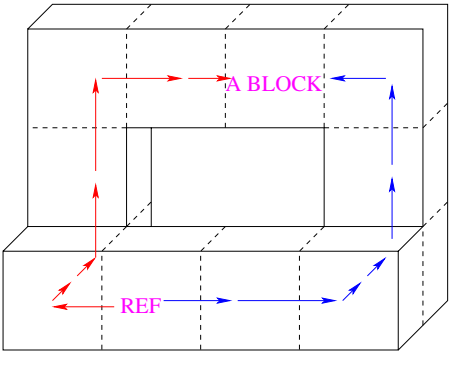


Fig. 4. An example of a structure built using the blocks (and transitions) of the automaton in Figure 3. Consider paths from block REF to block BLOCK. As is the sole block type, we have $RRBUUL \sim LBUURR$. This indicates that when following one of the framed walks, and then the other *but in reverse*, one returns to the original block.

This notion of rigidity of the structure has the following consequence.

Theorem 1. *Suppose that two structures* $\mathfrak{C}_1$ *and* $\mathfrak{C}_2$ *which consist of* n *blocks each are generated from the same block types according to transitions of the same*

automaton. Then they are (geometrically) congruent if and only if for some block $\mathbf{b}_1$ *of* $\mathfrak{C}_1$ *and some block* $\mathbf{b}_2$ *of* $\mathfrak{C}_2$, $\hat{W}_n(\mathfrak{C}_1, \mathbf{b}_1) = \hat{W}_n(\mathfrak{C}_2, \mathbf{b}_2)$.

Hence we can treat a structure as a set of framed walks, i.e., as a language, thus removing the geometry from the computation. Using this fact, and *assuming that* $\sim$ *(and associated apparatus) is PTIME computable*, we can prove that congruence of rigid structurees is PTIME computable, which contrasts from the popular suspicion that the Graph Isomorphism problem is *not* PTIME computable.

4 Summary

We also need to make sure that two blocks do not try to occupy the same space. We can use a forbidden set of words (walks) in the automaton to prevent blocks from intersecting (see Figure 5). By creating a set of *excluded* walks the bug is forbidden from making. Call this set of excluded walks I. It can be shown that I is a semigroup ideal (the walkspace itself is a semigroup under the concatenation operator). Note that in the example of Figure 3, $I = \varnothing$. But given more possible blocks and orientations, as in the example shown in Figure 5, we can have I to be nonempty, indeed infinite, but sometimes (as in the case of the example in Figure 5), readily computable.

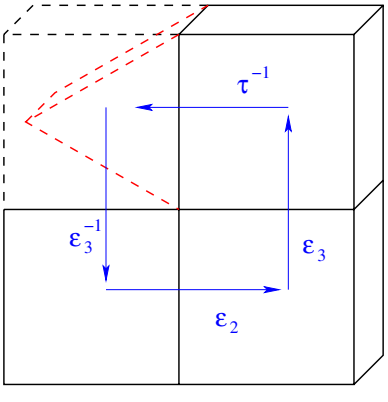


Fig. 5. We cannot insert a cube in the given slot. Using " " to represent the cube type and "△" to represent the pyramid type, and setting D to be Down, R to be Right, etc., we see that the framed walk $DRUL\triangle$ is excluded.

By building a computable algebraic model of a structure, block by block, not adding any blocks forbidden by the exclusion set, connecting blocks to complete cycles mandated by $\sim$, a computer and programmer may design a blueprint or an analysis of a complex structure. (Computer-aided analysis (e.g., the database at [2]) and design (e.g., the engine described in [4]) is already done for crystals.) Characterizing classes of building block types that have computable algebraic structures ($\sim$ and I) is a challenge for the theoreticians.

References

1. Abelson, H., deSessa, A.A.: Turtle Geometry: The Computer as a Medium for Exploring Mathematics. MIT Press, Cambridge (1980)
2. Cambridge Structural Database, Cambridge Crystallographic Data Centre, http://www.ccdc.cam.ac.uk/
3. Chen, H-I., Goel, A.: Error-free assembly using error-prone tiles. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA Computing. LNCS, vol. 3384, pp. 62–75. Springer, Heidelberg (2005)
4. Foster, M.D., Treacy, M.M.J., Higgins, J.B., Rivin, I., Balkovsky, E., Randall, K.H.: A systematic topological search for the framework of ZSM-10. J. Appl. Crystallography 38, 1028–1030 (2005), http://www.hypotheticalzeolites.net/
5. Jonoska, N., McColm, G.L.: Flexible versus Rigid Tile Assembly. In: Calude, C.S., Dinneen, M.J., Păun, G., Rozenberg, G., Stepney, S. (eds.) UC 2006. LNCS, vol. 4135, pp. 139–151. Springer, Heidelberg (2006)
6. Kuipers, J.B.: Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality. Princeton U. Press, Princeton (1999)
7. Twarock, R.: A tiling approach to virus capsid assembly explaining a structural puzzle in virology. J. Theor. Biol. 226, 477–482 (2004)
8. Winfree, E.: Algorithmic Self-assembly of DNA, PhD Thesis, CalTech (1998)

On the Undecidability of the Tiling Problem

Jarkko Kari*

Department of Mathematics, FIN-20014 University of Turku, Finland
jkari@utu.fi

Abstract. The tiling problem is the decision problem to determine if a given finite collection of Wang tiles admits a valid tiling of the plane. In this work we give a new proof of this fact based on tiling simulations of certain piecewise affine transformations. Similar proof is also shown to work in the hyperbolic plane, thus answering an open problem posed by R.M.Robinson 1971 [9].

1 Introduction

A *Wang tile* is a unit square tile with colored edges. Tiles are placed on the plane edge-to-edge, under the matching constraint that abutting edges must have the same color. Tiles are used in the given orientation, without rotating. If T is a finite set of Wang tiles, a tiling of the plane is a covering $t : \mathbb{Z}^2 \longrightarrow T$ of the plane by copies of the tiles in such a way that the color constraint is satisfied everywhere.

The *tiling problem* (also known as the domino problem) is the decision problem that asks whether a given finite tile set T admits at least one valid tiling $t : \mathbb{Z}^2 \longrightarrow T$. This problem was proved undecidable by R.Berger in 1966 [1], see also R.M.Robinson [9] for another proof. Both proofs rely on an explicit construction of an *aperiodic* tile set. Set T is called aperiodic if it admits some valid tiling of the plane, but it does not admit a valid periodic tiling, i.e. a tiling that is invariant under some translation. Note that existence of such aperiodic sets is not obvious, and in fact it was conjectured prior to Berger's work that they do not exist. If aperiodic sets did not exist, then the tiling problem would be decidable as one can simply try tilings of larger and larger rectangles until either (1) a rectangle is found that can no longer be tiled, or (2) a tiling of a rectangle is found that can be repeated periodically. Only aperiodic tile sets fail to reach either (1) or (2).

Note that Wang tiles are an abstraction of geometric tiles. Indeed, by using suitable "bumps" and "dents" on the sides to represent different colors, one can effectively replace any set of Wang tiles by a set of geometric tiles (all polygons with rational coordinates) such that the geometric tiles admit a tiling (a non-overlapping covering of the plane) if and only if the Wang tiles admit a tiling. Hence undecidability of the tiling problem by geometric tiles follows from Berger's result.

* Research supported by the Academy of Finland grant 211967.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 74–82, 2008.

In this work we present a new proof for the undecidability of the tiling problem. The proof uses plane tilings to simulate dynamical systems that are based on piecewise affine transformations. The undecidability of the tiling problem will then be concluded from the undecidability of the mortality problem of such dynamical systems.

A particularly nice feature of our proof is the fact that it is purely combinatorial. As a result of this the method generalizes easily to tilings in other lattices as well. In particular, we show that the tiling problem is undecidable in the hyperbolic plane. This resolves an open question asked already by Robinson in 1971 [9], and discussed by him in more details in 1978 [10]. In particular, Robinson proved the undecidability of the origin constrained tiling problem in the hyperbolic plane. This is the easier question where one asks the existence of a valid tiling that contains a copy of a fixed seed tile. We mention that there is a concurrent, independent and unpublished approach by M.Margenstern to the tiling problem in the hyperbolic plane [8]. We have reported our approach previously in [5].

We first discuss piecewise affine transformations and their mortality problem. We then outline the construction of corresponding Wang tiles. We then conclude by providing the analogous construction in the hyperbolic plane.

2 Mortality Problems of Turing Machines and Piecewise Affine Maps

Our proof is based on a reduction from the *mortality problem* of Turing machines. In this question we are given a deterministic Turing machine with a halting state, and the problem is to determine if there exists a non-halting configuration, that is, a configuration of the Turing machine that never evolves into the halting state. Such configuration is called immortal. Note that the Turing machine operates on an infinite tape, and configurations may contain infinitely many non-blank symbols.

Mortality problem of Turing machines. Does a given Turing machine have an immortal configuration ?

The mortality problem was proved undecidable by P.K.Hooper in 1966 [4], the same year that Berger proved his result. The two results have similar flavor, but proofs are independent in the sense that they do not rely on each other in either direction. Note an analogy to aperiodic tile sets: Hooper's result means that there must exist aperiodic Turing machines, that is, Turing machines that have immortal configurations but no immortal configuration repeats itself periodically. Our present proof establishes another connection between Hooper's and Berger's results since we reduce the mortality problem to the tiling problem.

We first consider dynamical systems determined by piecewise affine transformations of the plane. There exists a well known technique to simulate Turing machines by such transformations, see e.g. [2,7]. The idea is to encode Turing machine configurations as two real numbers $(l, r) \in \mathbb{R}^2$, representing the

left and the right halves of the infinite tape, respectively. The integer parts of l and r uniquely determine the next rule of the Turing machine to be used. More precisely, suppose the tape and state alphabets of the Turing machine are $A = \{0, 1, \ldots, a\}$ and $Q = \{0, 1, \ldots, b\}$. Let M be an even integer such that $M > a + 1$ and $M > b$. Then we let

$$l = \sum_{i=-1}^{-\infty} M^i t_i, \text{ and}$$

$$r = Mq + \sum_{i=0}^{\infty} M^{-i} t_i,$$

where $q \in Q$ is the current state of the Turing machine and $(t_i)_{i\in\mathbb{Z}}$ is the content of the infinite tape. We use a moving tape model of Turing machines: The Turing machine always reads cell 0 while the tape shifts left or right according to the rules of the Turing machine. Note that $\lfloor r \rfloor = Mq + t_0$ determines the next move of the machine, and that the encoding is one-to-one.

For each transition rule of the Turing machine one can effectively associate a rational affine transformation of $\mathbb{R}^2$ that simulates that transition. The matrix of the transformation is

$$\begin{pmatrix} M & 0 \\ 0 & \frac{1}{M} \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} \frac{1}{M} & 0 \\ 0 & M \end{pmatrix}$$

depending on the direction of the movement associated with the transition. The translation part of the transformation takes care of the changes in the tape symbol in cell 0 as well as the change in the state of the Turing machine.

In this fashion any deterministic Turing machine is converted into a system of finitely many rational affine transformations $f_1, f_2, \ldots, f_n$ of $\mathbb{R}^2$ and corresponding disjoint unit squares $U_1, U_2, \ldots, U_n$ with integer corners. Squares U_i serve as domains for the affine maps: the affine transformation f_i is applied when (l, r) is in the unit square U_i. Together the transformations define a partial function $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ whose domain is $U = U_1 \cup U_2 \cup \ldots \cup U_n$, and whose operation is

$$\vec{x} \mapsto f_i(\vec{x}) \text{ for } \vec{x} \in U_i.$$

Point $\vec{x} \in \mathbb{R}^2$ is called immortal if for every $i = 0, 1, 2, \ldots$ the value $f^i(\vec{x})$ is in the domain U. In other words, we can continuously apply the given affine transformations and the point we obtain always belongs to one of the given unit squares U_i.

The reduction from Turing machines to piecewise affine transformations preserves immortality: the Turing machine has an immortal configuration if and only if the corresponding system of affine maps has an immortal starting point. Hence we conclude from Hooper's result that the following immortality question is undecidable:

Mortality problem of piecewise affine maps: Does a given system of rational affine transformations $f_1, f_2, \ldots, f_n$ of the plane and disjoint unit squares $U_1, U_2, \ldots, U_n$ with integer corners have an immortal starting point ?

3 Reduction into the Euclidean Tiling Problem

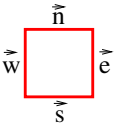
Next the mortality question of piecewise affine maps is reduced into the tiling problem of Wang tiles. The idea is very similar to a construction of an aperiodic Wang tile set presented in [6]. In [6] a tile set was given such that every valid tiling is forced to simulate an infinite orbit according to the one-dimensional piecewise linear function $f : [\frac{1}{2}, 2] \longrightarrow [\frac{1}{2}, 2]$ where

$$f(x) = \begin{cases} 2x, & \text{if } x \leq 1, \text{ and} \\ \frac{2}{3}x, & \text{if } x > 1. \end{cases}$$

Function f has no periodic orbits so the corresponding tile set is aperiodic.

The construction needs to be generalized in two ways: (1) instead of linear maps we need to allow more general affine maps, and (2) instead of $\mathbb{R}$ the maps are now over $\mathbb{R}^2$. Fortunately both generalizations are very natural and work without any complications.

The colors of our Wang tiles are elements of $\mathbb{R}^2$. Let $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ be an affine function. We say that tile

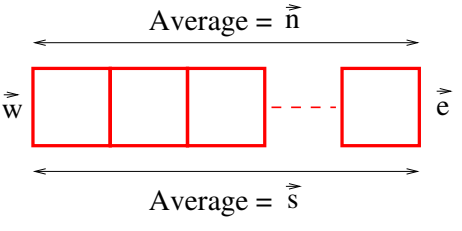


computes function f if

$$f(\vec{n}) + \vec{w} = \vec{s} + \vec{e}.$$

(The "input" $\vec{n}$ comes from north, and $f(\vec{n})$ is computed. A "carry in" $\vec{w}$ from the west is added, and the result is split between the "output" $\vec{s}$ to the south and the "carry out" $\vec{e}$ to the east.)

Suppose we have a correctly tiled horizontal segment of length n where all tiles compute the same f.



It easily follows that

$$f(\vec{n}) + \frac{1}{n}\vec{w} = \vec{s} + \frac{1}{n}\vec{e},$$

where $\vec{n}$ and $\vec{s}$ are the averages of the top and the bottom labels. As the segment is made longer, the effect of the carry in and out labels $\vec{w}$ and $\vec{e}$ vanish. Loosely speaking then, in the limit if we have an infinite row of tiles, the average of the input labels is mapped by f to the average of the output labels.

Consider now a given system of affine maps f_i and unit squares U_i. For each i we construct a set T_i of Wang tiles that compute function f_i and whose top edge labels $\vec{n}$ are in U_i. An additional label i on the vertical edges makes sure that tiles of different sets T_i and T_j cannot be mixed on any horizontal row of tiles. Let

$$T = \bigcup_i T_i.$$

If T admits a valid tiling then the system of affine maps has an immortal point. Namely, consider any horizontal row in a valid tiling. The top labels belong to a compact and convex set U_i. Hence there is $\vec{x} \in U_i$ that is the limit of the top label averages over a sequence of segments of increasing length. Then $f_i(\vec{x})$ is the limit of the bottom label averages over the same sequence of segments. But the bottom labels of a row are the same as the top labels of the next row below, so $f_i(\vec{x})$ is the limit of top label averages of the next row. The reasoning is repeated for the next row, and for all rows below. We see that $\vec{x}$ starts an infinite orbit of the affine maps, so it is an immortal point.

We still have to detail how to choose the tiles so that any immortal orbit of the affine maps corresponds to a valid tiling. Consider a unit square

$$U = [n, n+1] \times [m, m+1]$$

where $n, m \in \mathbb{Z}$. Elements of

$$\mathrm{Cor}(U) = \{(n, m), (n, m+1), (n+1, m), (n+1, m+1)\}$$

are the *corners* of U. For any $\vec{x} \in \mathbb{R}^2$ and $k \in \mathbb{Z}$ denote

$$A_k(\vec{x}) = \lfloor k\vec{x} \rfloor$$

where the floor is taken for each coordinate separately:

$$\lfloor (x, y) \rfloor = (\lfloor x \rfloor, \lfloor y \rfloor).$$

Denote

$$B_k(\vec{x}) = A_k(\vec{x}) - A_{k-1}(\vec{x}) = \lfloor k\vec{x} \rfloor - \lfloor (k-1)\vec{x} \rfloor.$$

It easily follows that if $\vec{x} \in U$ then

$$B_k(\vec{x}) \in \mathrm{Cor}(U).$$

Vector $\vec{x}$ will be represented as the two-way infinite sequence

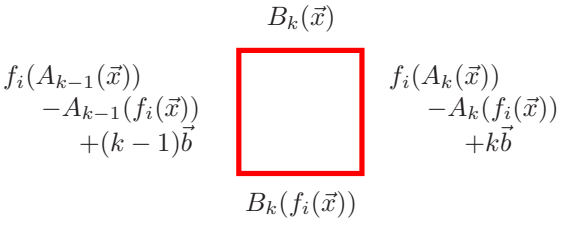
$$\ldots B_{-2}(\vec{x}), B_{-1}(\vec{x}), B_0(\vec{x}), B_1(\vec{x}), B_2(\vec{x}), \ldots$$

of corners. It is the *balanced representation* of $\vec{x}$, or the sturmian representation of $\vec{x}$. Note that both coordinate sequences are sturmian.

The tile set corresponding to a rational affine map

$$f_i(\vec{x}) = M\vec{x} + \vec{b}$$

and its domain square U_i consists of all tiles



where $k \in \mathbb{Z}$ and $\vec{x} \in U_i$. Observe the following facts:

(1) For fixed $\vec{x} \in U_i$ the tiles for consecutive $k \in \mathbb{Z}$ match in the vertical edges so that a horizontal row can be formed whose top and bottom labels read the balanced representations of $\vec{x}$ and $f_i(\vec{x})$, respectively.
(2) A direct calculation shows that the tile above computes function f_i, that is,

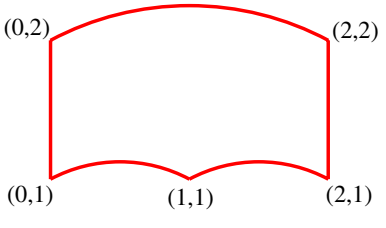
$$f_i(\vec{n}) + \vec{w} = \vec{s} + \vec{e}.$$

(3) Because f_i is rational, there are only finitely many tiles constructed, even though there are infinitely many $k \in \mathbb{Z}$ and $\vec{x} \in U_i$. Moreover, the tiles can be effectively constructed.

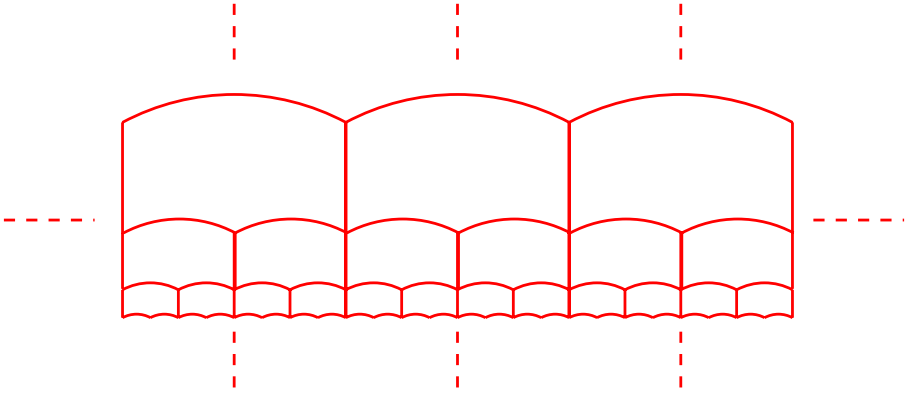
Now it is clear that if the given system of affine maps has an immortal point $\vec{x}$ then a valid tiling exists where the labels of consecutive horizontal rows read the balanced representations of the consecutive points of the orbit for $\vec{x}$. We conclude that the tile set we constructed admits a tiling of the plane if and only if the given system of affine maps is immortal. Undecidability of the tiling problem follows from the undecidability of the immortality problem that we established in Section 2

4 Reduction into the Tiling Problem on the Hyperbolic Plane

The method of the previous section works just as well in the hyperbolic plane. Instead of Wang tiles we use hyperbolic pentagons that in the half-plane model of hyperbolic geometry are copies of



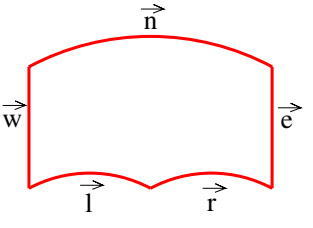
Note that all five edges are straight line segments. These tiles admit valid tilings of the hyperbolic plane in uncountably many different ways



In these tilings the tiles form infinite "horizontal rows" in such a way that each tile has two adjacent tiles in the next row "below".

In the following these hyperbolic pentagons are used instead of the Euclidean square shaped Wang tiles. The five edges will be colored, and in a valid tiling abutting edges of adjacent tiles must match. This is an abstraction – analogous to Wang tiles in the Euclidean plane – that can be transformed into hyperbolic geometric shapes using bumps and dents.

Exactly as in the Euclidean case we color the edges by vectors $\vec{x} \in \mathbb{R}^2$. We say that pentagon

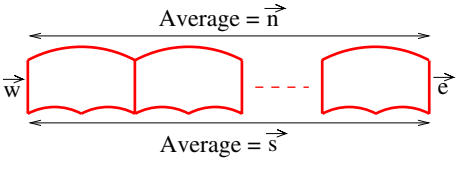


computes the affine transformation $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ if

$$f(\vec{n}) + \vec{w} = \frac{\vec{l} + \vec{r}}{2} + \vec{e}.$$

Note that the difference to Euclidean Wang tiles is the fact that the "output" is now divided between $\vec{l}$ and $\vec{r}$.

Consider a correctly tiled horizontal segment of length n where all tiles compute the same f.



Clearly we have

$$f(\vec{n}) + \frac{1}{n}\vec{w} = \vec{s} + \frac{1}{n}\vec{e},$$

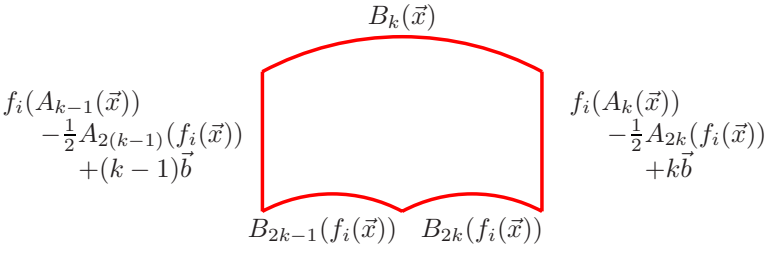
where $\vec{n}$ and $\vec{s}$ are the averages of the top and the bottom labels on the segment.

Analogously to the Euclidean case, given a system of affine maps f_i and unit squares U_i, we construct for each i a set T_i of pentagons that compute function f_i and whose top edge labels $\vec{n}$ are in U_i. It follows, exactly as in the Euclidean case, that if a valid tiling of the hyperbolic plane with such pentagons exists then from the labels of horizontal rows one obtains an infinite orbit in the system of affine maps.

We still have to detail how to choose the tiles so that the converse is also true: if an immortal point exists then its orbit provides a valid tiling. The tile set corresponding to a rational affine map

$$f_i(\vec{x}) = M\vec{x} + \vec{b}$$

and its domain square U_i consists of all tiles



where $k \in \mathbb{Z}$ and $\vec{x} \in U_i$. Now we can reason exactly as in the Euclidean case:

(1) For fixed $\vec{x} \in U_i$ the tiles for consecutive $k \in \mathbb{Z}$ match so that a horizontal row can be formed whose top and bottom labels read the balanced representations of $\vec{x}$ and $f_i(\vec{x})$, respectively.
(2) A direct calculation shows that the tile computes function f_i:

$$f_i(\vec{n}) + \vec{w} = \frac{\vec{l} + \vec{r}}{2} + \vec{e}.$$

(3) There are only finitely many pentagons constructed (because f_i is rational), and they can be formed effectively.

The tiles constructed admit a valid tiling of the hyperbolic plane if and only if the corresponding system of affine maps has an immortal point. So we have proved

Theorem. The tiling problem is undecidable in the hyperbolic plane.

References

1. Berger, R.: Undecidability of the Domino Problem. Memoirs of the American Mathematical Society 66, 72 (1966)
2. Blondel, V., Bournez, O., Koiran, P., Papadimitriou, C., Tsitsiklis, J.: Deciding stability and mortality of piecewise affine dynamical systems. Theoretical Computer Science 255, 687–696 (2001)

3. Goodman-Strauss, C.: A strongly aperiodic set of tiles in the hyperbolic plane. Inventiones Mathematicae 159, 119–132 (2005)
4. Hooper, P.K.: The undecidability of the Turing machine immortality problem. The Journal of Symbolic Logic 31, 219–234 (1966)
5. Kari, J.: The Tiling Problem Revisited (extended abstract). In: Durand-Lose, J., Margenstern, M. (eds.) MCU 2007. LNCS, vol. 4664, pp. 72–79. Springer, Heidelberg (2007)
6. Kari, J.: A small aperiodic set of Wang tiles. Discrete Mathematics 160, 259–264 (1996)
7. Koiran, P., Cosnard, M., Garzon, M.: Computability with low-dimensional dynamical systems. Theoretical Computer Science 132, 113–128 (1994)
8. Margenstern, M.: About the domino problem in the hyperbolic plane, a new solution. Manuscript , 109 (2007), http://www.lita.univ-metz.fr/~margens/ and also see arXiv:cs/0701096, same title
9. Robinson, R.M.: Undecidability and nonperiodicity for tilings of the plane. Inventiones Mathematicae 12, 177–209 (1971)
10. Robinson, R.M.: Undecidable tiling problems in the hyperbolic plane. Inventiones Mathematicae 44, 259–264 (1978)

Remote Entrusting by Run-Time Software Authentication

Mariano Ceccato[2], Yoram Ofek[1], and Paolo Tonella[2]

[1] University of Trento, Italy
ofek@dit.unitn.it
[2] Fondazione Bruno Kessler—IRST, Trento, Italy
{ceccato, tonella}@fbk.eu

Abstract. The problem of software integrity is traditionally addressed as the static verification of the code before the execution, often by checking the code signature. However, there are no well-defined solutions to the run-time verification of code integrity when the code is executed remotely, which is refer to as run-time remote entrusting. In this paper we present the research challenges involved in run-time remote entrusting and how we intend to solve this problem. Specifically, we address the problem of ensuring that a given piece of code executes on an remote untrusted machine and that its functionalities have not been tampered with both before execution and during run-time.

1 Introduction

When the software industry discusses software integrity, the main focus is on the protection of static software modules (e.g., by verifying the signature of their originator). On the other hand, dynamic software authentication in real-time during execution is a known problem without a satisfactory solution. Specifically, how to ensure that trusted code base (i.e., the software as was specified and coded) is running on an untrusted machine at all times and that the original code functionality was not modified prior to or during execution, is an open research challenge. This issue of entrusting software components is crucial since software, computers and networks are invading all aspects of modern life.

The issue of executing software in a trusted computing (TC) environment has gained a great deal of attention recently, in particular, the TCG (Trusted Computing Group) [22], Microsoft NGSCB (Next Generation Secure Computing Base) [23] and TrustZone developed by ARM [24] (see the next related work subsection for more details). These activities are somewhat complementary and orthogonal to the work presented in this paper. The previous approaches are hardware-based, and consequently, will not be available on all existing machines. Our research hypothesis is that a solution can be designed at any layer as a software component enhancing the layer itself in a cost-effective fashion; in contrast, TC is invasive, since it requires special hardware on the "motherboard". The proposed novel paradigm for remote entrusting of software will be available as a general, platform-independent solution (i.e., it is non-monopolistic, thus more

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 83–97, 2008.

competitive). The solution adds another line of defense to complement the current hardware solutions; while Trusted Computing (TC) can help manage keys and verify the system integrity during startup, it offers little protection against an attacker that already has access to the machine.

The key research question in remote entrusting is: "How can the execution of a software component be continuously entrusted by a remote machine, albeit the software component is running inside an untrusted environment?" (This is refer to as the "remote entrusting problem").

The solution to the above research problem should be able to employ external hardware, such as, smart cards, but not as a mandatory component. Furthermore, this work investigates a novel methodology for solving this problem by employing a software-based trusted logic component on a remote untrusted machine that in turn authenticates its operation continuously during run-time (i.e., execution). The method should assure the entrusting component that if the authentication is successful, then the original (i.e., unchanged) software functionality is being executed.

The long-term objective of the proposed approach is to entrust selected functionalities that are executed on untrusted machines and thereby ensure crucial trust/security properties.

Examples of possible applications are:

1. Protecting network resources and servers from users employing untrusted (i.e., unauthorized) software and protocols — specifically in the critical applications, such as, e-commerce, e-government.
2. Ensuring data privacy in Grid computing as well as digital right management (DRM) adherence by assuring proper processing of untrusted (possibly misbehaving) machines.

There are two fundamental differences between remote entrusting and other related approaches. Those fundamental differences are clear manifestation of some of the advancements beyond the state-of-the-art proposed by our approach.

1. *Core of trust location* — the basic working assumption when dealing with trust is that "some system components can be trusted" called at times, "core of trust". In some current approaches, such as trusted computing (TC), the "core of trust" is located locally on the "mother board", while in RE-TRUST the "core of trust" is placed in a remote trusted entity across the network. In other words, our model intend to address the trust problem by using the network under the assumption of continuous network connectivity, which is almost a reality today.
2. *Entrusting/validation method* — our proposed validation method is a significant departure from the-state-of the-art by introducing a novel protocol that provides software trust (or authentication) that is continuous during run-time — in other words, we introduce a proactive (avoidance) method. The main current approach to trust, e.g., TC (trusted computing) is offline or reactive (after the fact); namely, it may be possible to detect trust violations after some damage has been done. The objective of RE-TRUST project is to avoid breach of SW trust damages all together.

In the rests of the paper describes how we intend to address the remote entrusting problem. In Section 2 the basic remote entrusting approach is described and in Section 3 more details are given regarding the general architecture, pure-software and the hardware assisted solutions, then Section 4 introduces some possible attacks and discusses some possible protection mechanisms. The discussion in Section 5 concludes the paper.

1.1 Related Works

The initial work concerning the remote entrusting was developed by some consortium members in the TrustedFlow research activities [1,2]. The initial work introduced the ideas on how to generate a continuous stream of signatures using software only. The remote entrusting methodology is novel and challenging, and presents a major advancement beyond the state of the art. However, some specific aspects of the proposed research activities have been dealt with in different contexts. Therefore, the following state of the art discussion is divided into several subsections corresponding to various research aspects that are related to our approach.

Software dependability state of the art. Software dependability is a mature and well-established research area that seeks solutions to the problem of software errors that can corrupt the integrity of an application. To this aim, several techniques have been developed and the most prominent are control-flow checking and data duplication. Control-flow checking techniques are meant to supplement the original program code with additional controls verifying that the application is transitioning through expected valid "traces" [3,4,5]. In data duplication techniques, program variables are paired with a backup copy [6,7,8]. Write operations in the program are instrumented to update both copies. During each read access, the two copies are compared for consistency. There is one main difference regarding the "attack model", between software dependability and the current project. Software dependability assumes that modifications are accidental (random) errors (say bit flips), while remote entrusting deals with intentional and malicious software modifications.

Software tamper resistance state of the art. Among the several possible attacks, the focus is on the problem of authenticity, i.e., attacks aiming at tampering with application code/data for malicious purposes, like bypassing licensing, or forcing a modified (thus unauthorized) execution. Different solutions have been proposed in the literature to protect software from the above-mentioned rogue behaviors. Such solutions are surveyed in details in [9,10] and briefly described in the following. Obfuscation is used to make application code obscure so that it is complex to understand by a potential attacker who wants to reverse engineer the application. Obfuscation techniques, change source code structure without changing its functional behavior through different kinds of code transformations [11,12]. Theoretical studies about complexity of reverse engineering and de-obfuscation are in early stage. It is well-known that for binaries that

mix code and data disassembly and de-compilation are undecidable in the worst case [13]. On the other hand, some work reported that de-obfuscation (under specific and restrictive conditions) is an NP-easy problem [14]. Further, it was proven that a large number of functions cannot be obfuscated [15].

Replacement background state of the art. Dynamic replacement strategy relies on the assumption that tampering attempts can be made more complex if the attackers have to face newer versions continuously. This approach has some similarities with software aging [16], where new updates of a program are frequently distributed. This limits the spread of software "cracks" and it allows renewal of software protection techniques embedded in the application. Another relevant area of related work is represented by techniques for protection of mobile agents [16,17]. For instance, previous work proposed a scheme to protect mobile code using a ring-homomorphic encryption scheme based on CEF (computation with encrypted functions) with a non-interactive protocol [18,19]. However the existence of such homomorphic encryption function (also known as a privacy homomorphism) is still an open problem. Furthermore, some approaches mix obfuscation and mobility. For instance, in [20] agents are periodically re-obfuscated to ensure that the receiving host cannot access the agent state.

Hardware-based entrusting state of the art. Solution proposed by Trusted Computing initiatives [21,22,23,24] rely both on a trusted hardware component on the motherboard (co-processor) and on a common architecture that enable a trusted server-side management application to attest the integrity of a machine and to establishing its "level of trust". This non run-time approach has been applied to assess integrity of a remote machine enhanced with a trusted coprocessor and a modified Linux kernel [25]. In that work a chain of trust is created. First BIOS and coprocessor measure integrity of the operating system at start-up, then the operating system measure integrity of applications, and so on. Other non run-time approaches rely on additional hardware to allow a remote authority to verify software and hardware originality of a system [26]. Beside Trusted Computing, another interesting approach is presented in [27]. This approach has some similarities to our hardware assisted method, as it is based on commodity hardware tokens (e.g., smart cards) and remote execution of selected software components.

2 Basic Approach

Detection of software changes on a 1st machine by a 2nd machine across the network is difficult since the 2nd machine cannot directly observe the software executed on the 1st machine. As shown in Figure 1, in order to solve the problem, the 2nd machine should receive some "proofs" regarding the authenticity of the software that is running on the 1st machine. Such "proofs" are hard to obtain since the 2nd machine often receives only data from the 1st machine, while what is actually needed by the 2nd machine is to receive signatures (or attestations) continuously from selected parts of the software running on the 1st machine,

i.e., selected applications and protocols that are executed on the 1st machine. The signatures (or attestations) will thereby authenticate the respective selected software parts executed on the 1st machine. In other words, the signatures that are continuously emanated from selected parts of the software on the 1st machine provide the "identity" of the running software and thereby enabling the 2nd machine, after validation, to entrust the software running on the 1st machine. However, today selected applications and protocols that are developed and deployed on such 1st machines are not designed to emanate signatures (or attestations) continuously. In essence this is a paradigm shift and one of the main scientific/technical challenges introduced in the RE-TRUST project [29].

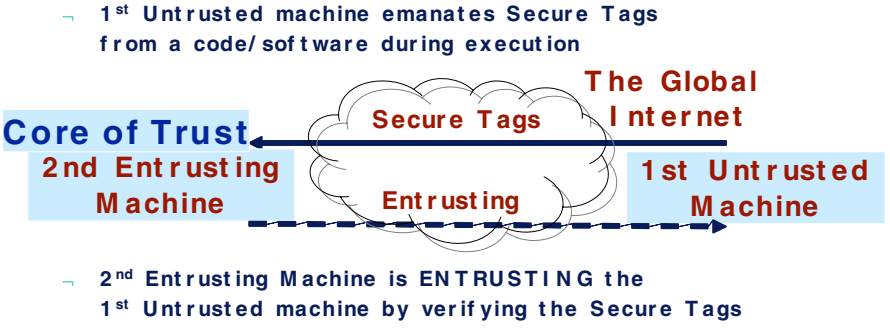


Fig. 1. Entrusting by remote software authentication during execution

As stated before, networking and computing are converging into one system, consequently, various security and trust problems are emerging. The core of the remote entrusting principle (or entrusting, for short), presented in this research project, is: "*To utilize trusted entities in the system/network (firewall, interface, server, protocol client, etc.) in order to entrust selected software components in otherwise untrusted machines across the network, assuring their on-line/run-time functionality*". Namely, entrusting is based on the assumption that there are trusted entities in the converged system of networking and computing (obviously, if nothing can be trusted, building any trust relationship is not feasible). The term "untrusted machine" implies that a malicious user has access to system resources (e.g., memory, disks, etc.) and tools (e.g., debuggers, disassemblers, etc.) on the 1st untrusted machine, and consequently, is capable of tampering/modifying the authentic (i.e., original) code prior to or during execution. In other words, the objective is that a 2nd entrusting machine (e.g., "Core of Trust", see Figure 1) will entrust the 1st untrusted machine by "authenticating its execution" (i.e., in real-time). Indeed, the execution of software (code/protocol) is authentic/trusted if and only if its functionality has not been altered/tampered by an untrusted/unauthorized entity, both prior to execution and, more importantly, during run-time. Finally, note that the basic remote entrusting scheme depicted in Figure 1, can be extended to contemplate:

– Mutual remote entrusting: where the 1st and 2nd machines are entrusting one another.
– Transitive remote entrusting: where a 1st machine is entrusting a 2nd machine and a 2nd machine is entrusting a 3rd machine.

3 General Architecture

The scientific and technical challenges involved in the present approach follows three orthogonal dimensions represented in Figure 2. Two dimensions represent two main software only approaches(code tamper resistance and code replacement), while the third represents the hardware assisted approach (tamper resistance (TR) using combined hardware (e.g., smart cards) and software).

The first software based dimension is the tamper resistance quality, it measures how difficult is to apply malicious modifications to the running program. Several techniques could be applied to increase the protection along this dimension, such as obfuscation, to increase the reverse engineering effort required to apply any attack.

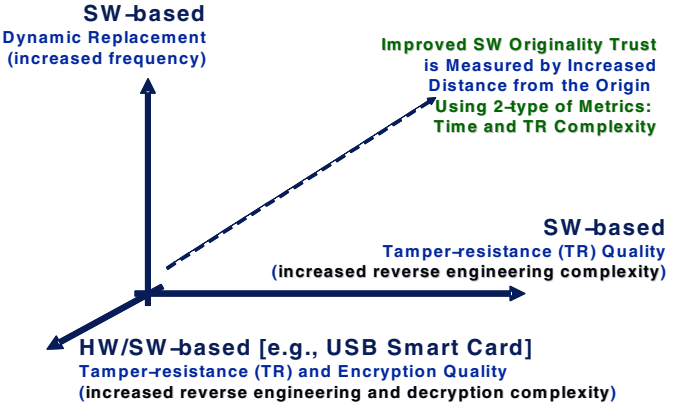


Fig. 2. Quality of remote entrusting

The second software pure-software dimension involves dynamic replacement. A portion of the application is periodically replaced at runtime, in order to give an attacker a limited time to complete an attack. Ideally a solution for remote entrusting should take advantage of both these two dimensions, because the more resilient is a tamper proof technique, the longer an attack would take to break it, the lower replacement frequency is required.

In the third dimension, tamper resistant methods involve co-design of application with software and hardware components, analyzing trade off between hardware and software. Pure software-based techniques may be extended to take advantage of the hardware dimension to increase the level of protection.

3.1 Pure Software Approach

The pure software dimensions investigate software-only methodologies for realizing the above-mentioned principle. In particular two objectives are addressed: (1) the secure software monitor should be combined (interlocked) in a secure way with the original application, and (2) the combined monitor must be robust against tampering (i.e., tamper resistance - TR). The first challenge will be dealt with by means of SW dependability techniques (e.g., for software faults detection). Tampering attacks are similar to random faults with the major difference that they are intentional (not accidental). Consequently, software dependability techniques are applicable to the trust domain as defined in Section 2. Finally, note that software dependability techniques are traditionally applied to a compiled code (e.g., C and C++). An additional challenge in this task will be to extend the above-mentioned techniques to an interpreted code (e.g., C# and Java). The second above objective will be addressed with two complimentary techniques: tamper resistance through software-based techniques, like source and binary obfuscation, and tamper avoidance, by dynamically replacing (parts of) secure software monitor, hence limiting the monitor lifetime (thus, also the tampering duration).

3.2 Hardware Assisted Approach

The hardware assisted dimension investigates tamper resistance methodologies combining hardware and software. With this approach, relatively inexpensive and widely available hardware monitors, such as smart cards or Trusted Platform Modules (TPMs) can be used to strengthen and improve the software-only protection method. A wide spectrum of possible solutions will be investigated ranging from low to high trust protection. This ranges from the hardware performing only some central operations (e.g., public key cryptography) to directly controlling the execution of major parts of the application, where the (untrusted) computer only stores encrypted code and data.

To investigate the combination of hardware- and software based software protection. The idea is twofold. On one hand, it is desired to utilize cheap and available hardware that alone may not be able to provide enough functionality. For example, trusted platform monitors, to strengthen the software protection. At the other extreme, the hardware itself may control most of the program flow, delivering maximum security at the price of a performance penalty. Finally, it is desired to investigate solutions in between the two extremes, to allow developers to freely choose the trade off between hardware cost, performance and security. Along the above lines, two major issues need to be investigated:

1. Regarding the low trust protection mode, execution of the code must be split between hard- and software, in a way that maximizes protection and minimizes the performance penalty.
2. Regarding the full trust protection, methods must be developed that allow an attacker to observe the entire communication between the computing

engine (the secure hardware) and the memory (in the PC), without learning any useful information.

Novel methods should be developed to scale the protection level, i.e., to discreetly adapt the trust and security level of specific scenarios.

3.3 Monitor

The secure software monitor and the original application must be correlated in such a way that any attempt to corrupt the authenticity of the application will be detected by the monitor, and that any attempt to harm the integrity of the monitor itself will stop the generation of valid signatures. This constitutes a major and open challenge in a pure software methodology. The initial approach will investigate techniques borrowed from the software dependability discipline, as in the area of software fault tolerance.

Innovative methods will be investigated to exploit the "time dimension" to increase overall tamper resistance of the secure software monitor. Namely, to bound the time available for attackers by means of dynamic software updates, where (parts of) the secure software monitor can be replaced at any instant during run-time. This approach improves tamper avoidance by making the life-time of each secure software monitor to a short defined time interval. To achieve this goal two major issues must be investigated, i.e., replacement strategies for interpreted (e.g., C# and Java) and compiled (e.g., C and C++) code the automated and non-predictable generation of secure software.

In pure software methods, the secure software monitor has to be protected from tampering. In particular, this requires solutions to two different problems: (1) the monitor behavior must be hidden to avoid trivial reverse engineering, and (2) secret data inside the monitor (e.g., encryption keys) must be hidden in order to be not easily spotted. It is envisioned that the obfuscation and white-box cryptography are the means to address the above-mentioned problems, respectively. This problem is articulated in the following ones: source-to-source obfuscation, obfuscation of (Java) byte code and protection of embedded keys with white-box cryptography techniques.

4 Attacks and Analysis

This sections introduces some possible attacks and then discusses some possible protection mechanisms.

4.1 Possible Attacks

A number of attacks may be applied to the remote entrusting software/hardware protection schemes. The attacker objective is to prevent the trusted machine from detecting tampering, and consequently, (remote) entrusting an untrusted machine. The rest of the section uses P to denote the program running on the untrusted machine that must be protected. The secure monitor will be called M.

Reverse engineering attacks. Reverse engineering attacks aim at locating important functionalities and data both in P and M. Once located, functionalities and data are altered maliciously. Key functionalities and data that can be the target of a reverse engineering attack in the remote entrusting scheme are:

- Secure tag (sequence) generator.
- Authenticity checking functions.
- Secret keys (e.g., used for secure tag generation).
- Input data (e.g., passed to checking functions).
- Output data (e.g., produced by checking functions).

The attacker may attempt to locate the function of M that generates the authenticity secure tag sequence and any secret key used for it. Once located, this functionality could be tampered with in order to produce correct authenticity tags even when P should not be trusted.

The attacker may attempt to locate and tamper with the functions that are devoted to checking the authenticity of P and of the underlying software and hardware. They could be modified so as to return a positive test result even when the actual result is negative. Moreover, they could be analyzed to understand which parts of the whole system are subjected to frequent checks and which ones are verified more rarely, in order to discover weak points where to apply a malicious modification that would be revealed with low probability.

The attacker may attempt to locate and change the input and output values involved in function calls, so as to change the behavior of the called function. This could be used to alter the result of a check performed by the monitor M or to tamper with the expected behavior of the program P. One way to change inputs or outputs is by directly modifying the code. Another possibility, which requires a combined execution environment attack (see next section), consists of intercepting the function call and changing input or output values dynamically. This second approach is stealthier than the former, because it cannot be detected by a code analysis on P. In fact, the running code is the original one.

Execution environment attacks. The attacker can tamper with the underlying execution environment, thus altering the behavior of P without modifying its code. Instead of deploying P on the actual processor, the attacker could run it on a simulated processor, which implements the same functionalities of the actual processor in software. It provides registers, interrupts and I/O devices. It can interpret and execute binary code. The simulated processor can be stopped when specific events occur. The current context (*i.e.*, memory, call stack, parameters) can be analyzed and modified. Then, execution is resumed. The attacker can take advantage of this infrastructure to intercept calls to libraries, to operating system and I/O facilities in order to dynamically modify parameters and memory locations and, thus, maliciously change the behavior of the program.

A similar attack consists of executing program P inside a debugger, which traces all the executed instructions, memory accesses and memory content. The debugger can interrupt the execution when selected instructions or conditions are met and the user can perform dynamic modifications.

Another attack to the execution environment may be directed toward the dynamic libraries. By altering them, it could be possible to trace and modify any operation that the program delegates to them, such as I/O, memory management, file system storage and network communication.

Cloning attack. In a cloning attack two copies of P are installed. The first copy is the original, unadulterated one. The monitor M is correctly installed with it and it is periodically updated. The execution environment, operating system and hardware are genuine. Such a program sends the server the expected authenticity secure tag sequence and, thus, is entrusted. The attacker maliciously modifies the second copy of P. The tampered copy runs in parallel with the first one, but not necessarily on the same client. The output of its monitor M is simply discarded.

All the network traffic coming from the server is sent both to the original and to the tampered applications (*e.g.* using a modified network device), so they can be executed in parallel on the same input values. The original application provides the authenticity tags required to be trusted by the server, whereas the tampered copy provides the modified behavior required by the malicious user.

The effectiveness of this attack depends on the possibility to decouple the generation of the tag sequence from the communication occurring between client and server. In fact, if the secure tag sequence originated from P includes data used by the server to carry out the computation required by the client (as prescribed by some variants of the remote entrusting scheme), a consistent computation must be performed on original and tampered copy, so as to keep unchanged the communication with the server. There are several classes of network applications for which the preservation of the communication between client and server entails that no malicious tampering is actually taking place (e.g., no unfair behavior can take place, no incorrect billing can be originated, etc.). One important class of applications for which preservation of the communication is not enough to ensure that no malicious tampering is taking place is Digital Right Management (DRM). In fact, DRM applications should prevent the client from creating illegal copies. However, creating an illegal copy does not involve any communication between client and server.

Differential analysis attack. Differential analysis consists of gathering information about the monitors by comparing the sequence of monitors produced and delivered by the monitor factory over time. Previously released monitors may be successfully broken when new ones are delivered, since the attacker has more time to reverse engineer them. If their analysis reveals, to some extent, the strategy implemented by the monitor factory, the attacker could take advantage of this knowledge to reduce the time necessary to break the current monitor, eventually compromising it before its expiration time.

Dependencies among attacks. The attacks described above are not independent of each other. In particular, the execution environment, cloning and differential analysis attacks all depend on the reverse engineering attack and cannot be performed without it.

With regards to the execution environment attack, deciding when to intercept the execution and how to alter it depends on the goal of the attacker. In turn, this requires some level of understanding of the program being tampered with. Potentially, a huge amount of information can be gathered and modified at run time. Focusing on the relevant functions, data and events requires a deep level of knowledge about the running code, hence the dominant problem becomes reverse engineering. Thus, modification of the execution environment is just a way to implement the reverse engineering attack.

A similar argument holds for cloning and differential analysis attacks. In the cloning attack, substantial reverse engineering effort must be devoted to locating the functions to be tampered in the program and to ensure the correct authentication secure tag sequence is sent to the server. Comparison between successive versions of the monitor aims at simplifying its reverse engineering, which is at the core of the possibility to modify it maliciously.

Overall, the attack model consists of a specific technique to gather and alter information (either adulteration of execution environment, cloning or differential analysis) combined with the understanding of the program and the monitor, to be achieved through reverse engineering.

4.2 Analysis of Attack Resistance

The trust model, and its variants, currently defined in the remote entrusting scheme address in various ways the attacks described in the previous section. In this section each source of trust in the trust model is related to the attacks it provides some defense against. The strength of such a defense is also briefly discussed.

Table 1 relates sources of trust to attacks. The meaning of a cell marked `X` is that the corresponding source of trust contributes to some extent to defend against the attack in this cell column. This does not mean that the source of trust provides full or proved protection against an attack. The given defense makes the attack harder, by addressing the vulnerabilities exploited by the attack, but it may still be only a partial defense.

For example, code obfuscation provides a limited defense against reverse engineering. In fact, an obfuscated code is expected to be harder to understand and analyze than a clear text. However, given enough time, a determined attacker can always reverse engineer a program, despite its code obscurity. Combined with monitor replacement, code obfuscation becomes a stronger defense, since only a limited time is available to the attacker to de-obfuscate the code.

Verification of the integrity of P (first row of Table 1) is a protection against malicious modifications that do not involve the monitor M. It must be combined with the verification of the verifier (M) itself to become effective against attacks that involve modifications of the monitor (columns 2, 3, 4, 6). Verification of the libraries (row 3) is an important defense line against execution environment attacks. However, more checks are needed to verify that HW/OS are genuine and that execution is not in debug mode. Direct implementation of this defense might be problematic and hard to achieve, since the monitor has limited

Table 1. Sources of trust related to specific attacks

	Attacks											
	Reverse engineering attacks							Execution environment attacks				
Sources of trust	P is tampered with (1)	Replace checking function (2)	Replace tag sequence generator (3)	Modify input before call on M/P (4)	Modify input before call on env. (5)	Modify output before return on M/P (6)	Modify output before return on env. (7)	Replace HW/OS (8)	Replace dynamic libraries (9)	Tampered execution (debug mode) (10)	Cloning attack (11)	Differential analysis (12)
(1) M checks P text and data segment	X											
(2) M self checks itself before checking P		X	X	X		X						
(3) M checks libraries used by P									X			
(4) M checks execution environment					X		X			X		
(5) M checks the OS and the HW								X				
(6) M checks results of computation	X			X	X	X	X	X	X	X		
(7) Secret key used to generate the tag sequence			X									
(8) Monitor replacement		X	X	X	X	X	X					X
(9) Rev-eng resistance (code obfuscation)	X	X	X	X	X	X	X	X	X	X	X	X
(10) Network of trust (self-checking implementation)		X	X	X		X						
(11) Tags include (portion of) output		X				X	X				X	
(12) Bi-directional communication (challenge from the server)		X	X	X	X	X	X					

capabilities of testing the HW/OS and the execution mode when running on the client. However, further lines of defense can be put in place. Firstly, the verification of the output of computations that depend on HW/OS and execution mode can be exploited for this purpose (see row 6). Moreover, since modification of the execution environment must be necessarily combined with a reverse engineering attack, reverse engineering resistance mechanisms, such as code obfuscation, provide a defense against execution environment attacks as well, in an indirect way.

The verification of the results of selected computations is also effective against modifications of P, M, as well as modifications of data passed to or returned from genuine functions of P and M. Hiding a secret key into the monitor is essential to protect the secure tag sequence generator (row 7). Monitor replacement (row 8) is an important countermeasure against tampering with the monitor (columns 2-7). Combined with the capability of producing new monitors that are independent from the previous ones, monitor replacement gives also some protection against the differential analysis attack.

Reverse engineering resistance, of which code obfuscation is one possible instance, is potentially effective against any attack, either directly or indirectly, because any attack must be necessarily combined with reverse engineering and program understanding, in order to alter the program behavior in a meaningful way, according to the attacker's goals.

The network of trust (row 10) increases the protection of the monitor's code (columns 2, 3, 4, 6). Inclusion of output data into the secure tags makes replacement of the checking function harder, as well as tampering with the output data produced by functions. Making the communication with the server bi-directional makes any change to the monitor harder (columns 2-7), since such changes might affect the verification triggered by the challenge, invalidating the result. If the verification is coupled with the ongoing computation, because the secure tags include a portion of the output (row 11), the cloning attack can be effective only if the communication with the server is preserved in the clone, which means that for many classes of applications no malicious tampering can actually take place at all.

5 Discussion

In this paper we presented the problem of remote entrusting as a novel paradigm, which give rise to multiple interdisciplinary problems encompassing many aspects of computing and networking. The central issue is how to entrust a piece of software executing on an untrusted and remote machine. We proposed a general architectural framework for remote entrusting with three problems: (*i*) how to combine two programs (the original code with the secure tags generation code) into one combined program, (*ii*) how to make it hard to separate using reverse engineering techniques, and (*iii*) how to dynamically replace parts of the combined program during run-time in order to limit the time available for an attacker to reverse engineer the combined program. The previous three problems

are investigated and solved along three main research dimensions, which require comprehensive solution. Specifically, dynamic replacement, tamper resistance with and without hardware assistance. A number of attacks have been identified on the proposed architectural framework, they have been analyzed and discussed.

Effective solutions of the remote entrusting problem will impact many application areas. Two main categories of applications have been identified,depending on the direction of the flow of data. The first category contains all the applications where the untrusted client sends data to the trusted machine (e.g., server) and the latter reacts by delivering a certain service (e.g., e-commerce, e-government). For these applications, a viable solution is based in hiding secure or authenticity tags in the outgoing data, in such a way that it would be difficult to tamper with them without affecting the data. Applications in the second category are those ones that receive private or protected data from the trusted party (e.g., grid computing, digital right management). In this case the solution is more challenging because, once delivered, protected data can not be longer protected, even if a tampering is detected.

Acknowledgments. This work was supported by funds from the European Commission (contract N^o 021186-2 for the RE-TRUST project, see [29]).

References

1. Baldi, M., Ofek, Y., Young, M.: Idiosyncratic Signatures for Authenticated Execution of Management Code. In: Brunner, M., Keller, A. (eds.) DSOM 2003. LNCS, vol. 2867, Springer, Heidelberg (2003)
2. Baldi, M., Ofek, Y., Young, M.: The TrustedFlow(TM) Protocol - Idiosyncratic Signatures for Authenticated Execution. In: 4th Annual IEEE Information Assurance Workshop, West Point, NY, USA (June 2003)
3. Oh, N., Shirvani, P.P., McCluskey, E.J.: Control-flow checking by software signatures. IEEE Transactions on Reliability 51(1) (March 2002)
4. Ohlsson, J., Rimen, M.: Implicit signature checking. In: Proceedings of 25th International Symposium on Fault-Tolerant Computing (June 1995)
5. Benso, A., Di Carlo, S., Di Natale, G., Prinetto, P., Tagliaferri, L.: Control-flow checking via regular expressions. In: Proceedings of 10th Asian Test Symposium (November 2001)
6. Oh, N., Mitra, S., McCluskey, E.J.: ED4 I: error detection by diverse data and duplicated instructions. IEEE Transactions on Computers 51(2) (February 2002)
7. Oh, N., Shirvani, P.P., McCluskey, E.J.: Error detection by duplicated instructions in super-scalar processors. IEEE Transactions on Reliability 51(1) (March 2002)
8. Benso, A., Chiusano, S., Prinetto, P., Tagliaferri, L.: A C/C++ source-to-source compiler for dependable applications. In: DSN. Proceedings of International Conference on Dependable Systems and Networks (June 2000)
9. Collberg, C., Thomborson, C., Low, D.: Watermarking: Tamper-Proofing, and Obfuscation - Tools for Software Protection. IEEE Transactions on Software Engineering 28 (2002)
10. Naumovich, G., Memon, N.: Preventing piracy, reverse engineering, and tampering. IEEE Computer 36(7), 64–71 (2003)

11. Wang, C., Davidson, J., Hill, J., Knight, J.: Protection of software-based survivability mechanisms. In: DSN. Proceeding of International Conference on Dependable Systems and Networks, Goteborg, Sweden (July 2001)
12. Valdez, E., Yung, M.: Software DisEngineering: Program Hiding Architecture and Experiments. Information Hiding (1999)
13. Linn, C., Debray, S.: Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In: CCS. Proceedings of the 10th ACM Conference on Computer and Communications Security (October 2003)
14. Appel, A.W.: Deobfuscation is in NP, www.cs.princeton.edu/~appel/papers/deobfus.pdf
15. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (Im)possibility of Obfuscating Programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, Springer, Heidelberg (2001)
16. McGraw, G., Felten, E.W.: Mobile Code and Security. IEEE Internet computing 2(6) (1998)
17. Esparza, O., Soriano, M., Munoz, J.L., Forne, J.: Detecting and Proving Manipulation Attacks in Mobile Agent Systems. In: Karmouch, A., Korba, L., Madeira, E.R.M. (eds.) MATA 2004. LNCS, vol. 3284, pp. 224–233. Springer, Heidelberg (2004)
18. Sander, T., Tschudin, C.F.: Towards Mobile Cryptography. IEEE Symposium on Security and Privacy (May 1998)
19. Sander, T., Tschudin, C.F.: Protecting mobile agents against malicious hosts. LNCS (1998)
20. Badger, L., et al.: Self-protecting mobile agents obfuscation techniques evaluation report. NAI Labs Report (November 2001), www.isso.sparta.com/research/documents/spma.pdf
21. Pearson, S.: Trusted computing platforms, the next security solution. Technical Report HPL-2002-221, HP Laboratories (2002)
22. The Trusted Computing Group, https://www.trustedcomputinggroup.org
23. Next Generation Secure Computing Base, http://www.microsoft.com/resources/ngscb
24. York, R.: A New Foundation for CPU Systems Security. ARM Limited, http://www.arm.com
25. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and Implementation of a TCG-based Integrity Measurement Architecture. In: Proceedings of the 13th USENIX Security Symposium San Diego, CA, USA (August 2004)
26. Kennell, R., Jamieson, L.H.: Establishing the Genuinity of Remote Computer Systems. In: Proceedings of the 12th USENIX Security Symposium (2003)
27. Mana, A., Lopez, J., Ortega, J., Pimentel, E., Troya, J.M.: A Framework for Secure Execution of Software. International Journal of Information Security 3(2) (2004)
28. Seshadri, A., Luk, M., Shi, E., Perrig, A., van Doorn, L., Khosla, P.K.: Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems. In: SOSP. Proceedings of the 20th ACM Symposium on Operating Systems Principles, Brighton, UK, pp. 1–16 (October 23-26, 2005)
29. http://re-trust.org/

Trusted Computing — Special Aspects and Challenges

Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
sadeghi@crypto.rub.de

Abstract. The advent of e-commerce, e-government, and the rapid expansion of world-wide connectivity demands end-user systems that adhere to well-defined security policies. In this context Trusted Computing (TC) aims at providing a framework and effective mechanisms that allow computing platforms and processes in a distributed IT system to gain assurance about each other's integrity/trustworthiness. An industrial attempt towards realization of TC is the initiative of the Trusted Computing Group (TCG), an alliance of a large number of IT enterprises. The TCG has published a set of specifications for extending conventional computer architectures with a variety of security-related features and cryptographic mechanisms. The TCG approach has not only been subject of research but also public debates and concerns. Currently, several prominent academic and industrial research projects are investigating trustworthy IT systems based on TC, virtualization technology, and secure operating system design.

We highlight special aspects of Trusted Computing and present some current research and challenges. We believe that TC technology is indeed capable of enhancing the security of computer systems, and is another helpful means towards establishing trusted infrastructures. However, we also believe that it is not a universal remedy for all of the security problems we are currently facing in information societies.

1 Introduction

The increasing global connectivity, distributed applications and digital services over the Internet, both for business and private use, require IT systems that guarantee confidentiality, authenticity, integrity, privacy, as well as availability. Examples include online banking, e-commerce and e-government services, content delivery, etc. Modern cryptography and information security provide a variety of very useful technical security measures such as encryption and strong authentication mechanisms to achieve these security targets. However, these measures provide only partial solutions as long as the underlying computing platforms still suffer from various security problems in hardware and software: Beside architectural security problems and the inherent vulnerabilities resulting from high complexity, common computing platforms require careful and attentive system administration skills, and are still unable to effectively protect against execution of malicious code and tampering.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 98–117, 2008.

In this context some fundamental and challenging issues are how to define and to determine/verify the integrity/trustworthiness[1] of a computing platform or in general of an IT system, and how could common computing platforms support such functionalities. Note that even a perfectly secure operating system cannot verify its own integrity. Today, the authorized access to online services or data over the Internet is controlled by means of secure channel protocols where the standard approaches are security protocols like Transport Layer Security (TLS) [14] or Internet Protocol Security (IPSec) [25]. However, these protocols do not provide any guarantees regarding the integrity/trustworthiness of the communication endpoints. Security breaches on the Internet today rarely involve compromising the secure channel because endpoints are much easier to compromise since they are frequently subject to attacks by malware (e.g., viruses and Trojan horses). Using a secure channel to an endpoint of unknown integrity is ultimately futile.[2]

Hence, reliable mechanisms are desired that allow to verifiably reason about the "trustworthiness" of a peer endpoint (local or remote), i.e., whether its state (the hardware and software configuration) conforms to a defined security policy.

A recent industrial initiative towards this goal is put forward by the *Trusted Computing Group* (TCG), a consortium of a large number of IT enterprises, which proposes a new generation of computing platforms that employs both, supplemental hardware and software. The claimed goal of this architecture is to improve the security and the trustworthiness of computing platforms (see, e.g., [35,42]). The TCG has published several specifications on various concepts of trusted infrastructures [54].[3] The core component the TCG specifies is the Trusted Platform Module (TPM). The current widespread implementation of the TPM is a small tamper-evident cryptographic chip [57,53]. Many vendors already ship their platforms with TPMs (mainly laptop PCs and servers). To this end, the conventional PC architecture is extended by new mechanisms to (i) protect cryptographic keys, (ii) authenticate the configuration of a platform (attestation), and (iii) cryptographically bind (sensitive) data to a certain system configuration (sealing), i.e., the data can only be accessed (unsealed) if the corresponding system can provide the specific configuration for which the data has been sealed.

Trusted Computing technology was subject of public debates due to its claimed capabilities, in particular, in conjunction with Digital Rights Management (DRM) (see, e.g., [3,16,36]). Concerns were aroused that TC technology may give

[1] "Trust" is a complicated notion that has been studied and debated in different areas (social science and humanities as well as computer science). A possible definition of the notion "trustworthy" is the degree to which the (security) behavior of the component is demonstrably compliant with its stated functionality (e.g., see [7]).

[2] In the words of Gene Spafford, *"Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit card information from someone living in a cardboard box to someone living on a park bench."* [50].

[3] Claimed role is to develop, define and promote open, vendor-neutral industry specifications for trusted computing including hardware building blocks and software interface specifications across multiple platforms and operating environments.

vendors and content providers new abilities to get control over personal systems and users' private information. Hence they may allow commercial censorship, political censorship, and product discrimination or undermine the General Public License (GPL [18]).

Meanwhile Trusted Computing has attracted many researchers and developers, and the capabilities as well as the shortcomings of the TCG proposals are better understood. Currently, several prominent research and industrial projects are investigating open trustworthy computing platforms (PC, servers, embedded) and applications based on Trusted Computing, virtualization technology and secure operating system design[4]. Since the body of related literature in this area has grown rapidly, and we have limited space, we only highlight some special aspects of the TC technology, and present some current research challenges and proposals made so far to tackle some shortcomings of the TCG specifications. Hence, we do not claim completeness and apologize for not referring to some of the works done in this area.

2 Main Aspects of the TCG Specification

2.1 Core Components and Functionalities

The main components of the TCG approach are a hardware component called *Trusted Platform Module* (TPM), a kind of (protected) pre-BIOS (Basic I/O System) called the *Core Root of Trust for Measurement* (CRTM), and a support software called *Trusted Software Stack* (TSS), which has various functions like providing a standard interface for TPMs of different manufacturers communicating with the rest of the platform or with remote platforms.

The TPM is the main component of the specification and provides a secure random number generator, non-volatile tamper-resistant storage, key generation algorithms, cryptographic functions like RSA encryption/decryption, and the hash function SHA-1 (see Figure 1). The TPM protects a variety of keys. Two of its main keys are the endorsement key (EK), an encryption key that uniquely identifies each TPM, and the Storage Root Key (SRK) or Root of Trust for Storage (RTS), uniquely created inside the TPM. The private SRK never leaves the TPM, and it is used to encrypt all other keys created by the TPM. The TPM state contains further security-critical data shielded by the TPM. Amongst them is a set of registers called *Platform Configuration Registers* (PCR) that can be used to store hash values. The hardware ensures that the value of a PCR register can only be modified as follows: $PCR_{i+1} := \text{SHA1}(PCR_i|I)$, with the old register value PCR_i, the new register value PCR_{i+1}, and the input I (e.g. a SHA-1 hash value). This process is called *extending* a PCR. Hash values computed during this process are called *measurements* in TCG terminology.

The TCG has published two specifications versions 1.1b [57] and 1.2 [53] for the TPM where the latter has more and also improved functionalities. In particular TPM 1.2 provides at least 4 concurrent monotonic counters and privacy

[4] See, e.g., EMSCB project (www.emscb.org) and OpenTC (www.opentc.net)

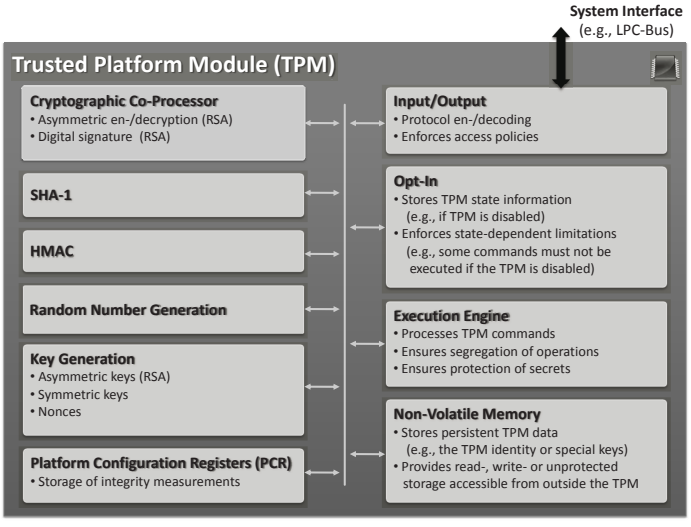


Fig. 1. TPM Architecture

enhanced protocols such as *Direct Anonymous Attestation* (DAA) [9]. Based on TPM functionalities, the TCG specification defines four mechanisms called *integrity measurement*, *attestation*, *sealing* and *maintenance* which are explained briefly in the following:

Integrity Measurement & Platform Configuration. Integrity measurement is done during the boot process by computing the hash value of the initial platform state. For this purpose the CRTM computes a hash of (*measures*) the code and parameters of the BIOS and extends the first PCR register by this result. A chain of trust is established if an enhanced BIOS and bootloader also measure the code they are transferring control to, e.g., the operating system. The security of the chain relies strongly on explicit security assumptions about the CRTM. The state of the PCRs is also called the platform's *configuration*.

Attestation. The TCG attestation protocol is used to give assurance about the platform configuration to a remote party. To guarantee integrity and freshness, PCR values and a fresh nonce provided by the remote party are digitally signed with an asymmetric key called *Attestation Identity Key* (AIK), which is under the sole control of the TPM. A trusted third party called *Privacy Certification Authority* (Privacy-CA) is used to guarantee the pseudonymity of the AIKs. In order to overcome the problem that the Privacy-CA can link transactions to a certain platform (e.g., by means of the same AIK), version 1.2 of the TCG specification defines a cryptographic protocol called *Direct Anonymous Attestation* (DAA) [9], a cryptographic credential scheme that aims at providing anonymity of the platform and unlinkability of remote authentication transactions of the TPM. DAA can be used to sign an AIK, or PCR values.

Sealing/Binding. Data can be cryptographically bound/encrypted to a certain platform configuration by the sealing operation of the TPM. The unseal operation releases the decrypted data only if the current configuration (software and/or hardware) equals the configuration, which has been defined when the data was sealed. Binding is like conventional asymmetric encryption where encrypted data can only be recovered by the TPM that knows the corresponding secret key (no platform configuration check is required).

Secure and Authenticated Boot. The former means that a system terminates the boot process in case the integrity check (e.g., comparing the system measurements with a securely stored reference value) fails, whereas the latter aims at proving the platform integrity to a (remote) verifier.

Maintenance. Maintenance functions can be used to migrate the SRK to another TPM: The TPM owner can create a maintenance archive by encrypting the SRK under a public key of the TPM vendor and an own secret key. In case of a hardware error the TPM vendor can load the SRK from the maintenance archive into another TPM. Currently the maintenance function is still optional and, to our knowledge, not implemented in the currently available TPMs. Moreover, the maintenance function works only for TPMs of the same vendor.

2.2 Trust Model and Assumptions

The TCG defines a component or a system as "trusted" *if it always behaves in the expected manner for the intended purpose*. Note that this definition differs subtly from the mostly used definition that a system or component is "trusted" *if its failure breaks the security policy of the system* (see, e.g., [2]). This definition requires the number of trusted components in a system, also called Trusted Computing Base (TCB), to be as small as possible.

The correctness and soundness of the TCG proposed functionalities are based on some assumptions, which we briefly discuss below. Building systems that can reasonably satisfy these assumptions in practice is a technical challenge and subject of ongoing research, which we will shortly consider in the subsequent sections.

- The platform configuration cannot be forged after the corresponding hash values are computed and stored in the TPM's PCR registers. Otherwise no reliable integrity reporting is possible. Note that currently available operating systems such as Windows or Linux can easily be modified, e.g., by exploiting security bugs or by changing memory which has been swapped to a hard disk. Hence, one requires an appropriate secure operating system design methodology.
- A verifier can determine the trustworthiness of the code from digests (hash) values. Note that today's operating systems and application software are extremely complex, making it very difficult, if not impossible, to determine their trustworthiness. Hence, in practice one requires more effective, reliable, and realistic methodology for this purpose than relying on hash values of binary codes.

– Secure channels can be established between hardware components (TPM and CPU), e.g., when both components are on the same chipset. Note that currently TPM chips are connected to the I/O system with an unprotected interface that can be eavesdropped and manipulated easily [28]. Secure channels to remote parties can be established based on cryptographic mechanisms and a Public Key Infrastructure (PKI).

The main hardware-based components, CRTM and TPM, are assumed to be trusted by all involved parties. Currently the CRTM is not contained in a tamper-resistant module. It should be noted that the TCG specification requires protection of these components only *against software attacks*. Nevertheless, certain hardware attacks may defeat the security of the TCG approach. Some TPM manufacturers have already started a third party certification of their implementation with respect to security standards (Common Criteria [13]) to assure a certain level of tamper-resistance. Although an integration of the TPM functionality into chipsets makes the manipulation of the communication link between TPM and the CPU significantly more difficult and costly, it also introduces new challenges since an external validation and certification of the TPM functionalities (e.g., required by some governments) will be much more difficult.

Even though the TCG approach explicitly allows an application to distinguish between different TPM implementations, the trade-off between costs and tamper-resistance, which is certainly application dependent, will finally determine the level of tamper-resistance.

2.3 Trusted Network Connect (TNC)

The specification for Trusted Network Connect (TNC) [52] has been published by the TNC working group of the TCG. It should be a vendor independent network standard that enhances network security by combining network access control with Trusted Computing. The goal is to integrate the concepts of Trusted Computing with existing network access control mechanisms.

The overall goal of TNC is to prevent compromise of the hosts that connect to a network or other network resources and thus the network itself. The specification suggests platform authentication through a proof of identity in combination with the integrity status of the platform that wants to connect to a network. Therefore network access control is based on extended attributes like platform authentication (e.g., by using an AIK), endpoint compliance, or software state information, which are collected and attested to a verifier. Based on this information the verifying instance is able to decide whether it is secure to extend the network to that platform.

2.4 Mobile Trusted Module (MTM)

The Mobile Trusted Module is a technology that can serve as an integrity control mechanism for protecting non-discretionary services in embedded devices. It adds features to the baseline TPMs in the domain of secure booting. In case

of mobile or embedded platforms, one uses the notion *Mobile Trusted Module* (MTM) [55,56]. Although an MTM has similar features as a common TPM, there are some differences. The Mobile Trusted Module specification defines two types of MTMs: the Mobile Remote-Owner Trusted Module (MRTM) and the Mobile Local-Owner Trusted Module (MLTM). The difference between them is that the MRTM must support mobile-specific commands defined in the MTM specification as well as a subset of the TPM 1.2 commands. Typically, phone manufacturers and network service providers use an MRTM. These parties only have remote access to the MTM whereas the MLTM is used by the user who has physical access to the device and his applications. The different parties, called stakeholders, have different requirements on the integrity, device authentication, SIM Lock/device personalization, secure software download, mobile ticketing and payment, user data protection, privacy issues and more. How these different types of MTMs are implemented is not defined since the specification published by the TCG is quite vague.

3 Property-Based Attestation and Sealing

Integrity verification of applications and their underlying Trusted Computing Base (TCB) is especially important in the context of security policies enforced in a distributed system. Here, remote integrity verification mechanisms should enable a remote party to verify whether an application *behaves* according to certain security policies.

The TCG solution for remote integrity verification are mechanisms called remote *binary attestation*, remote *binary binding*, and *binary sealing*. Although binding and sealing are two slightly different concepts, in the following only the term binding is used for simplicity. Nevertheless, the concepts of property-based binding can also be applied to the sealing functionality.

Loosely speaking, binary attestation and binary binding are based on (i) a measurement of the chain of executed code using a cryptographic digest and (ii) some trust assumption (see Section 2.2). However, TCG proposals have some shortcomings: (i) they reveal the information about the platform's hardware and software configuration and thus make fingerprinting attacks on the platform much easier (*security and privacy*), (ii) they allow remote parties to exclude certain system configurations (*discrimination*), e.g., a content provider can collaborate with an operating system vendor to allow only the software of that vendor to download certain content, (iii) data bound to a certain configuration is inaccessible after any update in firmware or software, or after hardware migrations (*data availability*), and (iv) the verifier is required to know all possible trusted configurations of all platforms (*scalability*).

A more general and flexible extension to the binary attestation is *property-based attestation* [41,38,27]: on higher system levels, attestation should only determine whether a platform configuration or an application has a desired property. Property-based attestation/binding should determine whether the target

machine to be attested fulfills certain requirements (e.g., provides certain access control methods). This avoids revealing the concrete configuration of software and hardware components. For example, it would not matter whether Web browser A or B is used, as long as both have the same properties. For most practical applications, the verifier is not really interested in the specific system or application configuration. This is even disadvantageous for the verifier since he has to manage a multitude of possible configurations. Basically, properties change rarely compared to binaries on program updates.

Some proposals in the literature consider the protection and prove the integrity of computing platforms in the context of secure and authenticated (or trusted) boot (see, e.g., [4], [15], [45], [49], [58]). A high-level protocol for property-based attestation is presented in [38]. The solution is based on property certificates that are used by a verification proxy to translate binary attestations into property attestations. In [41] the authors propose and discusses several protocols and mechanisms that differ in their trust models, efficiency, and the functionalities offered by the trusted components. In particular, [41] discusses how the TSS, the TPM library proposed by the TCG, can provide a property-based attestation protocol based on the existing TC hardware without a need to change the underlying trust model. Another refinement of this idea is proposed in [27]. Moreover, based on ideas of [41], [12] proposes a cryptographic zero-knowledge protocol for anonymous property-based attestation.

In [24] the authors propose *semantic remote attestation* using language-based trusted virtual machines (VM) to remotely attest high-level program properties. The general idea is to use a trusted virtual machine (TrustedVM) that verifies the security policy of the machine that runs within the VM.

In [31], [34] and [33] the authors propose a software architecture based on Linux providing attestation and binding. The architecture binds short-lifetime data (e.g., application data) to long-lifetime data (e.g., the Linux kernel) and allows access to that data only if the system is compatible to a security policy certified by a security administrator.

However, many challenges remain to be solved and are subject of ongoing research: how to define useful properties applicable for practice, how to build efficient mechanisms to determine properties of complex and composed systems, and how to formally capture this notion.

4 Secure Data Management

The integrity measurement mechanism securely stores the platform's initial configuration into the registers (PCRs) of the TPM. Any change to the measured software components results in changed PCR values, making sealed data inaccessible under the changed platform configuration. While this is desired in the case of an untrustworthy software suite or malicious changes to the system's software, it may become a major obstacle for applying patches or software updates. Such updates do generally not change the mandatory security policy enforced by an

operating system (in fact, patches *should* close an existing security weakness not included in the system specification). Nevertheless, the altered PCR values of the operating system make the sealed information unavailable under the new configuration. As mentioned in Section 3 the semantic of the sealing operation is too restrictive to efficiently support sealed information through the software life-cycle including updates/patches. The main problem is the lack of a mapping between the security properties provided by a platform configuration and its measurements. This difficulty is also pointed out in [42]. A further problem with the TCG's proposal is how to handle hardware replacements in a computing platform. Such replacements are necessary due to outdated or faulty hardware. In corporate contexts, hardware is typically replaced every few years. Any sealed data bound to a given TPM cannot directly be transferred to another TPM, because it is encrypted with a key protected by the SRK, which in turn is stored within the TPM. Further, the maintenance function does not allow platform owners to migrate to a TPM of a different vendor.

In [26] the authors address these problems and propose possible solutions for the secure migration, maintenance, and a more flexible sealing procedure. They also use the ideas on property-based attestation [41,38] (see also Section 3) to construct property-based sealing. However, some of these solutions require changes to the TPM specification and consequently to the TPM firmware. The future versions of the TPM specification may integrate some of these ideas. The challenge is, however, to reduce the TPM complexity but still have appropriate solutions for the mentioned problems above.

In [45] the authors present an integrity measurement architecture (IMA) for Linux (see also Section 8). However, platform updates and migration are not addressed, and, as the PCR values are employed to protect the current list of measurements, working with sealed data seems to be difficult.

5 Trusted Channels — Beyond Secure Channels

The standard approach for creating secure channels over the Internet is to use security protocols such as Transport Layer Security (TLS) [14] or Internet Protocol Security (IPSec) [25], which aim at assuring confidentiality, integrity, and freshness of the transmitted data as well as authenticity of the involved endpoints. However, as mentioned before, secure channels do not provide any guarantees about the integrity of the communication endpoints, which can be compromised by viruses and Trojans. Based on security architectures that deploy Trusted Computing functionalities (see also Section 10), one can extend these protocols with integrity reporting mechanisms (either binary or property-based attestation), as proposed in [21] for the case of TLS (see also related work in [21]).

In this context, an interesting issue would be to analyze how to extend other cryptographic protocols and mechanisms (e.g., group based cryptography like group key exchange) with integrity reporting and binding/sealing mechanisms

and all this under the weakest possible assumptions. Besides, the underlying security architecture should be capable of handling configuration changes and its validation in run time environment and be able to manage the corresponding access control, e.g., to generate new session keys. However, efficient and effective run time attestation remains an open problem.

6 Compliance and Conformance

The Trusted Platform Module (TPM) acts as the root of trust and is the basis for all other specifications of the TCG. Vendors already deploy computer systems, e.g., laptop computers, that are equipped with a TPM chip. The TPM is a basic but nevertheless very complex security component. Its specifications are continuously growing in size and complexity (120 commands and up to 19 command parameters in TPM v1.2), and there is still no published analysis on the minimal TPM functionalities that are practically needed. In addition to this, TPM users have to completely trust implementations of TPM manufacturers regarding the compliance but also conformance (security) to the TCG specification. This also requires the user to trust the TPM implementation that no malicious functionalities have been integrated (trapdoors or Trojan horses). Finally, the TCG adversary model considers software attacks only (see also 2.2).

Due to the complexity of the TPM specifications, one expects that not all TPM chips operate exactly as specified. In practice, different vendors may implement the TPM differently. They may exploit the flexibility of the specification or they may deviate from it by inappropriate design and implementation[5].

In [40], the authors introduce a prototype test suite developed for TPM compliance tests. Based on the test results, they point out the non-compliance and bugs of many TPM implementations currently available at the market. They present a testing strategy, some sample test results, and an analysis for different TPM implementations of different manufacturers, and discuss how one can construct attacks by exploiting non-compliance and deficiencies of protection mechanisms. However, their test suite does not cover the entire TPM specifications.

In the recent years many efforts have been invested into thorough analysis of cryptographic algorithms and security protocols resulting in a variety of methods and automatic tools. Security models and manual security proofs up to formal and automatic verification have been developed for this purpose, leading to a deeper understanding of proposed algorithms and protocols. However, not much is publicly known whether the same has been done for the TPM specification. While [40] focuses mainly on TPM compliance issues, [23] presents the results of an automated verification of some generic scenarios based on TPM functionalities

[5] One may argue that vendors can deviate from the TPM specification because the TCG does not enforce its brand name strongly enough. However, the TCG specification itself has a certain degree of flexibility which may be exploited. This is also the case with many other standards. Pushing a brand name or logo “TCG inside” may not be sufficient in general given the complexity of the TPM.

and identifies some security problems and inconsistencies which have been adopted into the recent version of the specification.

7 Virtual TPM

Virtualization [17,22] is a very useful technology that allows the cost-effective use of hardware and resource sharing. In the recent years virtualization technology is enjoying its rediscovery. Virtualization allows to run several virtual machines (VM) (e.g., several operating systems) on top of the same hardware platform, move the VMs among different platforms etc. *Virtual Machine Monitors* (VMM), or *hypervisors*, acting as a control instance between virtual machines and physical resources, provide isolation of processes. However, for security-critical applications where confidentiality and integrity of data are required, one needs mechanisms that assure the integrity of the underlying software (VMs and the virtualization layer), or that these components conform to the defined security policy. Combining VMM with Trusted Computing functionalities based on a hardware-based root of trust (e.g., TPM) can provide such mechanisms under specific assumptions (see also Sections 2.2 and 10).

In this context TPM virtualization makes TPM's capabilities available to multiple virtual machines running on the platform. However, a virtual (software) TPM (vTPM) underlies a different trust model than a hardware TPM and hence, there should be a secure link between each virtual TPM and the unique hardware TPM. This is a challenging task in particular with regard to migration of vTPM instances among different platforms that may have a different level of trust and security requirements. Moreover, the state of a virtual TPM shall not be subject to manipulation or cloning.

In [8] the authors propose an architecture where all vTPM instances are executed in one special VM. This VM also provides a management service to create vTPM instances and to multiplex the requests. Optionally, the vTPM instances may be realized in a secure co-processor card. In their approach, migration of vTPM instances is realized through an extension of the TPM command set. When migration of a vTPM is requested, the state of the vTPM is encrypted with a symmetric key that itself is encrypted by a migratable key of the real TPM. On the destination site, the symmetric key and then the state of the vTPM are decrypted and the vTPM instance resumes.

GVTPM [39] is a virtual TPM framework that supports various TPM models and even different security profiles for each VM under the Xen hypervisor. In contrast to providing fully virtualized TPM functionality, [46] virtualizes only one functionality of a TPM 1.2, namely monotonic counters.

However, one still needs more flexible architectures for secure management of vTPMs that among others (i) do not require extensions, and consequently more complexity for TPMs, (ii) are less dependent on binary hash-based measurements to properly manage updates and resealing, and (iii) can migrate VMs and their associated vTPMs to platforms with different security policies and different binary implementations of the underlying virtual machine monitor (VMM).

8 Integrity Measurement

AEGIS [4] performs an integrity check during the boot process of the whole operating system. It protects the integrity reference values by building a chain of trust and protecting the root reference value by special hardware. Enforcer [31] is a security module for the Linux kernel, which works as an integrity checker for file systems. It uses a TPM to verify the integrity of the boot process and to protect the secret key of an encrypted file system. A certain configuration of a system can be proved by comparing the values inside the TPM register to previously computed reference values. If data or applications are sealed with these values, the integrity of the underlying platform is indirectly assured. [45] introduces IMA, an integrity measurement architecture for Linux. It extends the Linux kernel and inserts measurement hooks in functions relevant for loading executable code. In this way, they extended the measurement chain from the BIOS and bootloader to the application level. However, frequent changes in application files on a running system, e.g., due to updates and patches, steadily increase the measurement list and may become impractical. Remote parties can first verify the integrity of the table of measurements using remote attestation, and can then decide if the current platform configuration is trustworthy.

9 New Processor Generation

Hardware vendors have improved both, the security features provided by the CPU and the appropriate chipsets by the following features.[6] First, a new CPU mechanism to protect security-critical code and data from untrusted code, e.g., an existing operating system. This allows a security kernel to be executed in parallel to an existing legacy operating system. Second, a CPU secure startup-technique based on TPM functionality that allows to load security critical code dynamically, i.e., after untrusted code has already been loaded. This functionality, called `SKINIT` on AMD Pacifica and `SENTRY` on Trusted Execution Technology, is an alternative realization of the CRTM that does neither require a tamper-resistant BIOS extension nor modifications of the bootloader. Third, an extension of the mainboard chipset allows to prevent DMA-enabled devices from accessing security-critical data [47,11].

10 Security Architectures — Possible Approach

The basic desired primitives required for a trustworthy IT system are (i) means for integrity verification that allow a computing platform to export verifiable information about its properties (comes from the requirement of assuring the executing image and environment of an application located on a remote computing platform), (ii) secure storage that allows applications to persist data

[6] Intel's Trusted Execution Technology formerly known as LaGrande (see `http://www.intel.com/technology/security`) and AMD's Virtualization Technology formerly code-named Pacifica (see `http://www.amd.com/virtualization`)

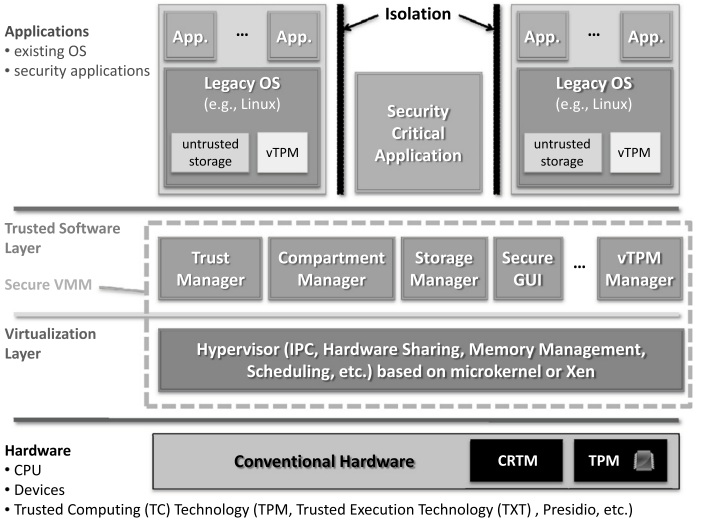


Fig. 2. Possible Security Architecture

securely between executions using traditional untrusted storage, (iii) strong process isolation, and assuring (memory space) separation between processes, (iv) secure I/O to assure that users securely interact with the intended application, and (v) interoperability and ability to use legacy software.

A possible security architecture that aims to provide the above mentioned properties is shown in Figure 2. Its realization deploys various technologies such as virtualization and Trusted Computing (TC). The Virtual Machine Monitor (VMM) consists of a security kernel that is located as a control instance between the hardware and the application layer. It implements elementary security properties like trusted channels and strong isolation between processes. Virtualization technology enables reutilization of legacy operating systems and existing applications whereas TC technology serves as the root of trust.

Hardware Layer. The hardware layer consists of commercial off-the-shelf PC hardware enhanced with trusted computing technology as defined by the Trusted Computing Group (TCG) [51] (see also Section 2).

Virtualization Layer. The main task of the virtualization layer, also called *hypervisor*, is to provide an abstraction of the underlying hardware, e.g., CPU, interrupts and devices, and to offer an appropriate management interface as well as inter-process communication (IPC). Device drivers and other essential operating system services, such as process and memory management, run in isolated user-mode processes. Moreover, this layer enforces an access control policy based on these resources. Providing a virtualized environment is one approach

to secure computing systems that process potentially malicious code. This technique is widely used for providing V-Servers, i.e., servers that feature several virtual machines. While users have full control over the virtual environment, they should not be able to cause damage outside that environment. Although virtualization offers abstraction from physical hardware and some control over process interaction, there still are problems to be solved. For example, in the x86 architecture, direct memory access (DMA) devices can access arbitrary physical memory locations. However, new hardware generations (see Section 9) aim to address these problems and could eventually lead to secure isolation among virtual machines. Virtualization technology can be leveraged for building an appropriate environment for a variety of applications, especially because several works, such as [43,44], have already begun to consider architectures that feature policy enforcement in the virtualization framework.[7] Possible implementations can be based on microkernels [29] or Xen hypervisor [6].

Trusted Software Layer. The trusted software layer provides various security related services and uses the functionality offered by the virtualization layer to provide security functionalities on a more abstract level. A possible design of these services is illustrated in Figure 2, which is being currently implemented within different projects.[8] The main services and their main tasks are as follows: A *Compartment Manager* for managing compartments, i.e., logically isolated components, e.g., virtual machines (VMs); a *Storage Manager* for compartments' persistent storage providing integrity, confidentiality, authenticity and freshness; a *Trust Manager* for providing trusted computing services, e.g., attestation, sealing, trusted channels to other (remote) compartments; a *User Manager* for managing user accounts and credentials; and a *Secure UI* for establishing a trusted path between the user and compartments, e.g., to assure to which application a user is communicating and to prevent keyloggers and spyware attacks.

Application Layer. On top of the security kernel, several instances of legacy operating systems (e.g., Linux) as well as security-critical applications (e.g., grid jobs, online banking, hard disk encryption, e-voting, VPN, DRM, etc.) can be executed in strongly isolated compartments. The proposed architecture offers migration of existing legacy operating systems. The legacy operating system provides all operating system services that are not security-critical and offers users a common environment and a large set of existing applications. If a mandatory security policy requires isolation between applications of the legacy OS, they can be executed by parallel instances of the legacy operating system.

Verifiable Initialization. For verifiable bootstrapping of the TCB, the CRTM measures the Master Boot Record (MBR) before passing control to it. A secure

[7] sHype is a security extension for Xen developed by IBM. sHype provides a MAC-based security architecture by adding static type enforcement (static coloring) on communication channels and allows to enforce a *Chinese Wall* restriction on concurrently running domains.

[8] EMSCB (www.emscb.org) and OpenTC (www.opentc.net)

chain of measurements is then established: before a program code is executed, it is measured by a previously (measured and executed) component.[9] The measurement results are securely stored in PCRs of the TPM. All further compartments, applications, and legacy OS instances are then subsequently loaded, measured, and executed by the Compartment Manager.

Examples for a security architecture as above is PERSEUS [37]. Other related work include Microsoft's *Next-Generation Secure Computing Base* (NGSCB) and *Terra* [20]. NGSCB was originally planed as a security architecture based on the TCG specifications [35]. However, the status of this project is currently unclear.[10] Terra is a trusted virtual machine monitor (VMM) that partitions a hardware platform into multiple, strictly isolated virtual machines (VM).

11 Applications

Trusted Computing framework and the related security architectures (see, e.g., Section 10) enable policy enforcement, and in particular multilateral security, which is crucial for applications with sophisticated security needs such as commercial grid, e-government, e-health, online banking to name some. In the following we consider some of the recent work on application of TC functionalities.

In [32,30] the authors propose to utilize TC technology for grid applications, with [48] more closely examining which scenarios require TC techniques. The common protection mechanisms in grid usually do not concern themselves with protecting the grid user (the person or entity wishing to utilize resources). The user is forced to trust the provider, often without the possibility to verify whether that trust is justified. However, in most of the current literature on grid security the user is usually not regarded as trustworthy. This trust asymmetry could potentially lead to a situation in which the grid provider causes large damage to the user with little risk of detection or penalty.

Trusted monotonic counters enable different types of distributed services that are vulnerable to replay attacks, including offline payment, e-wallets, virtual trusted storage, and management of stateful licenses in various digital rights management (DRM) applications. Stateful licenses are required in scenarios for trading and using digital goods where the enforcement of the underlying policies requires to securely maintain the state information about the past usage or environmental factors. In [46] the authors propose a trusted monotonic counter solely based on multiplexing the TPM counters without requiring a trusted operating system whereas [5] proposes a security architecture for the secure management (e.g., usage, and transfer) of stateful licenses and content on a virtualized open platform. The proposed architecture makes use of the monotonic counters of the

[9] For this purpose, one can use a modified GRUB bootloader c(`www.prosec.rub.de/tgrub.html`).

[10] Microsoft has often changed the name of its security platforms: Palladium (see, e.g., [47]), NGSCB, Longhorn and recently Windows Vista which uses TPM v1.2 for hard disk encryption (system partition) to protect data against theft.

TPM combined with a software security service to provide trusted storage with freshness.

In [19], the authors propose the design and implementation of a security architecture that aims at preventing both classical and malware phishing attacks. The security architecture deploys ideas of compartmentalization for isolating applications of different trust level, and a trusted wallet for storing credentials and authenticating sensitive services. Other work on preventing phishing attacks based on TC can be found in [1].

In [10] the authors describe a secure network virtualization framework for establishing *Trusted Virtual Domains* (TVDs) and its implementation based on Xen hypervisor [6]. The proposed framework aims at connecting groups of related virtual machines running on separate physical machines as if there were on their own separate network while enforcing the domain's security policy (in their case isolation, confidentiality, and information flow control). Trusted computing functionalities can assure that core components for establishing TVDs are trustworthy, i.e., conform to a the well-defined security policy.

Other applications include deployment of TC for biometric systems security, video broadcasting, medical privacy, Web security, protection of massively multiplayer online games (MMOGs), RFID applications with privacy need, single sign-on, secure information sharing, protecting digital signatures, TC-based security architecture for mobile platforms, VPN and hard disk encryption applications.[11]

12 Conclusion and Future Work

Trusted Computing (TC) is an emerging technology that can enhance the security of computing platforms in many ways. Nevertheless, TC technology is not a universal solution to all of the IT security problems, but rather an additional tool to pave the way towards trusted infrastructures. The technological realization of TC brings new technical but also legal and economical challenges. It is difficult to predict whether or not the security benefits of TC outweighs the efforts needed to face these challenges. Though we strongly believe that research on this technology results in a deeper understanding of complex IT systems and how to maintain their integrity, since trusted computing concerns security aspects at many system abstraction layers (hardware, operating systems and application software). Many challenges still remain as we discussed previously: designing effective and efficient remote proof of trustworthiness of codes, or of a platform (e.g., property-based attestation), incorporating the underlying hardware in the chain of trust, designing a minimal and less complex but effective TPM, simplifying complex cryptographic protocols by using TC functionalities, developing secure and efficient VMMs, establishing appropriate formal security models and analysis for the corresponding mechanisms and architectures.

[11] See `www.isg.rhul.ac.uk/~pnai178/tcgresources.htm`, and `www.emscb.org` for a collection of papers on these topics.

In particular, trusted mobile platforms seem to be an important future application and market segment since embedded devices are increasingly used for security critical applications and have challenging requirements due to their constrains but also the diversity of the parties involved as well as their requirements and interests.

Even though the security level strongly depends on the details of design and implementation, Trusted Computing as an abstract functionality is inevitable for realizing applications and IT infrastructures with sophisticated security requirements.

Acknowledgment. We are very grateful to Frederik Armknecht, Christian Stüble and in particular to Christian Wachsmann for the fruitful discussions and their valuable comments on the early draft of this survey.

References

1. Alsaid, A., Mitchell, C.J.: Preventing Phishing Attacks using Trusted Computing Technology. In: INC (July 2006)
2. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems, 1st edn. John Wiley & Sons, New York, USA (2001)
3. Anderson, R.J.: The TCPA/Palladium FAQ (2002), http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html
4. Arbaugh, W.A., Farber, D.J., Smith, J.M.: A secure and reliable bootstrap architecture. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society, Technical Committee on Security and Privacy, pp. 65–71. IEEE Computer Society Press, Los Alamitos (1997)
5. Asokan, N., Ekberg, J.-E., Sadeghi, A.-R., Stüble, C., Wolf, M.: Enabling fairer digital rights management with trusted computing. In: ISC (2007)
6. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP (2007)
7. Benzel, T.V., Irvine, C.E., Levin, T.E., Bhaskara, G., Nguyen, T.D., Clark, P.C.: Design principles for security. Technical Report NPS-CS-05-010, Naval Postgraduate School (September 2005)
8. Berger, S., Caceres, R., Goldman, K.A., Perez, R., Sailer, R., van Doorn, L.: vTPM: Virtualizing the Trusted Platform Module. In: Proceedings of the 15th USENIX Security Symposium, pp. 305–320. USENIX (August 2006)
9. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: ACM-CCS (October 2004)
10. Cabuk, S., Chris, H.R., Dalton, I., Schunter, M.: Towards automated provisioning of secure virtualized networks. In: ACM-CCS (2007)
11. Carroll, A., Juarez, M., Polk, J., Leininger, T.: Microsoft "Palladium": A business overview. Technical report, Microsoft Content Security Business Unit (August 2002)
12. Chen, L., Landfermann, R., Löhr, H., Rohe, M., Sadeghi, A.-R., Stüble, C.: A protocol for property-based attestation. In: ACM-STC, ACM Press, New York (2006)

13. Common Criteria Project Sponsoring Organisations. Common criteria for information technology security evaluation. Norm Version 2.1, CCIMB-99-031 – 33, Common Criteria Project Sponsoring Organisations (August 1999), http://csrc.nist.gov/cc/CC-v2.1.html
14. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1, RFC4346 (April 2006), http://www.ietf.org/rfc/rfc4346.txt
15. Dyer, J., Lindemann, M., Perez, R., Sailer, R., van Doorn, L., Smith, S.W., Weingart, S.: Building the IBM 4758 Secure Coprocessor. IEEEC 34(10), 57–66 (2001)
16. Felten, E.W.: Understanding Trusted Computing — Will Its Benefits Outweigh Its Drawbacks? IEEE Security and Privacy, 60–62 (May/June 2003)
17. Figueiredo, R., Dinda, P.A., Fortes, J.: Resource virtualization renaissance. IEEE Computer 38, 28–31 (2005)
18. Foundation, F.S.: GNU General Public License, Version 3, http://gplv3.fsf.org/
19. Gajek, S., Sadeghi, A.-R., Stüble, C., Winandy, M.: Compartmented security for browsers — or how to thwart a phisher with trusted computing. In: ARES (2007)
20. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. In: SOSP, pp. 193–206. ACM, New York (2003)
21. Gasmi, Y., Sadeghi, A.-R., Stewin, P., Unger, M., Asokan, N.: Beyond secure channels. In: ACM-STC (2007)
22. Goldberg, R.P.: Architectural Principles for Virtual Computer Systems. PhD thesis, Harvard University (1972)
23. Gürgens, S., Rudolph, C., Scheuermann, D., Atts, M., Plaga, R.: Security evaluation of scenarios based on the TCG TPM specification. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 438–453. Springer, Heidelberg (2007)
24. Haldar, V., Chandra, D., Franz, M.: Semantic remote attestation: A virtual machine directed approach to trusted computing. In: USENIX Virtual Machine Research and Technology Symposium May 2004, Also Technical Report No. 03-20, School of Information and Computer Science, University of California, Irvine (October 2003)
25. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol, RFC2401 (November 1998), www.ietf.org/rfc/rfc2401.txt
26. Kühn, U., Kursawe, K., Lucks, S., Sadeghi, A.-R., Stüble, C.: Secure data management in trusted computing. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, Springer, Heidelberg (2005)
27. Kühn, U., Selhorst, M., Stüble, C.: Property-Based Attestation and Sealing with Commonly Available Hard- and Software. In: ACM-STC (2007)
28. Kursawe, K., Schellekens, D., Preneel, B.: Analyzing Trusted Platform Communication. In: ECRYPT-CRASH (2005)
29. Liedtke, J.: Towards real micro-kernels. Commun. ACM 39(9) (1996)
30. Löhr, H., Ramasamy, H.G.V., Schulz, S., Schunter, M., Stüble, C.: Enhancing Grid Security Using Trusted Virtualization. In: ATC (2007)
31. MacDonald, R., Smith, S., Marchesini, J., Wild, O.: Bear: An open-source virtual secure coprocessor based on TCPA. Technical Report TR2003-471, Department of Computer Science, Dartmouth College (2003)
32. Mao, W., Jin, H., Martin, A.: Innovations for Grid Security from Trusted Computing, http://www.hpl.hp.com/personal/Wenbo_Mao/research/tcgridsec.pdf

33. Marchesini, J., Smith, S., Wild, O., Barsamian, A., Stabiner, J.: Open-source applications of TCPA hardware. In: ACSAC, ACM, New York (2004)
34. Marchesini, J., Smith, S.W., Wild, O., MacDonald, R.: Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear. Technical Report TR2003-476, Department of Computer Science, Dartmouth College (2003)
35. Mundie, C., de Vries, P., Haynes, P., Corwine, M.: Microsoft whitepaper on trustworthy computing. Technical report, Microsoft Corporation (October 2002)
36. Oppliger, R., Rytz, R.: Does trusted computing remedy computer security problems? IEEE Security & Privacy 3(2), 16–19 (2005)
37. Pfitzmann, B., Riordan, J., Stüble, C., Waidner, M., Weber, A.: The PERSEUS system architecture. Technical Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory (April 2001)
38. Poritz, J., Schunter, M., Van Herreweghen, E., Waidner, M.: Property attestation—scalable and privacy-friendly security assessment of peer computers. Technical Report RZ 3548, IBM Research (May 2004)
39. Rozas, C.: Intel's Security Vision for Xen (April 2005), http://www.xensource.com/files/XenSecurity_Intel_CRozas.pdf
40. Sadeghi, A.-R., Selhorst, M., Christian Stüble, C., Wachsmann, Winandy, M.: TCG Inside? — A Note on TPM Specification Compliance. In: ACM-STC (2006)
41. Sadeghi, A.-R., Stüble, C.: Property-based attestation for computing platforms: Caring about properties, not mechanisms. In: ACM SIGSAC, The 2004 New Security Paradigms Workshop, ACM Press, New York (2004)
42. Safford, D.: The need for TCPA. IBM Research (October 2002)
43. Sailer, R., Jaeger, T., Valdez, E., Caceres, R., Perez, R., Berger, S., Griffin, J.L., van Doorn, L.: Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor (2005)
44. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: Secure hypervisor approach to trusted virtualized systems. Technical Report RC23511, IBM Research Division (February 2005)
45. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. Research Report RC23064, IBM Research (January 2004)
46. Sarmenta, L.F.G., van Dijk, M., O'Donnell, C.W., Rhodes, J., Devadas, S.: Virtual monotonic counters and count-limited objects using a tpm without a trusted os. In: ACM-STC, pp. 27–42 (2006)
47. Schoen, S.: Palladium details (2002), http://www.activewin.com/articles/2002/pd.shtml
48. Smith, M., Friese, T., Engel, M., Freisleben, B.: Countering Security Threats in Service-Oriented On-Demand Grid Computing Using Sandboxing and Trusted Computing Techniques. Journal of Parallel and Distributed Computing (2006)
49. Smith, S.W.: Outbound authentication for programmable secure coprocessors. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 72–89. Springer, Heidelberg (2002)
50. Spafford, G.: Risks Digest 19.37 (September 1997), http://catless.ncl.ac.uk/Risks/19.37.html
51. Trusted Computing Group, www.trustedcomputinggroup.org

52. Trusted Computing Group. TCG Architecture Overview (April 2004)
53. Trusted Computing Group. TPM main specification. Main Specification Version 1.2 rev. 85, Trusted Computing Group (February 2005)
54. Trusted Computing Group (TCG). About the TCG, http://www.trustedcomputinggroup.org/about/
55. Trusted Computing Group (TCG). TCG Mobile Reference Architecture, Specification version 1.0, Revision 1 (June 12, 2007)
56. Trusted Computing Group (TCG). TCG Mobile Trusted Module Specification, version 1.0, Revision 1 (June 12, 2007)
57. Trusted Computing Platform Alliance (TCPA). Main specification, Version 1.1b (February 2002)
58. Yee, B.S.: Using Secure Coprocessors. PhD thesis, School of Computer Science, Carnegie Mellon University, CMU-CS-94-149 (May 1994)

Optimizing Winning Strategies in Regular Infinite Games

Wolfgang Thomas

RWTH Aachen, Lehrstuhl Informatik 7, 52056 Aachen, Germany
thomas@informatik.rwth-aachen.de

Abstract. We consider infinite two-player games played on finite graphs where the winning condition (say for the first player) is given by a regular omega-language. We address issues of optimization in the construction of winning strategies in such games. Two criteria for optimization are discussed: memory size of finite automata that execute winning strategies, and – for games with liveness requests as winning conditions – "waiting times" for the satisfaction of requests. (For the first aspect we report on work of Holtmann and Löding, for the second on work of Horn, Wallmeier, and the author.)

1 Introduction

A central result in the algorithmic theory of infinite two-player games is the Büchi-Landweber Theorem [2]. It shows how to "solve" finite-state games where the winning condition is given by a regular ω-language (over the state space of the underlying finite game arena). The solution of a game involves two algorithms, the first one to decide for each state who of the two players wins the plays starting from this state, and the second one to synthesize a winning strategy for the respective winner. As it turns out, finite-state winning strategies suffice, i.e. strategies that are realized by a finite-state machine with output (for example in the format of a Mealy automaton). For some further background see [10,13].

The Büchi-Landweber Theorem provides an approach for the automatic synthesis of finite-state controllers, assuming that the synthesis problem can be modeled in the framework of finite-state games (with a regular winning condition). Two directions of research have been opened by this fundamental result. First, various options of extending the framework have been investigated, such as infinite-state games, concurrent games, distributed (or multiplayer) games, and stochastic games, and in each case the scope of models that still allow an algorithmic solution has been explored. The other direction aims at a refined analysis of finite-state games, with the objective to single out cases that are interesting in applications and allow an efficient treatment. We discuss the second problem in this paper, starting with the observation that from a practical viewpoint the Büchi-Landweber Theorem is insufficient in two respects: It involves procedures of high computational complexity, and – if a controller exists – the controller's behavior is not optimized in any way. Progress on both questions seems possible

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 118–123, 2008.

since the full power of regular winning conditions is rarely needed in applications; many interesting cases can be studied in a more restricted framework. As important conditions of this kind we mention (and pursue below) the "weak" winning conditions and the "request-response conditions". In this context we address the aspect of optimization for two criteria:

1. memory size of the controller
2. waiting times when liveness conditions are to be satisfied.

Regarding these criteria we report (in Sections 2 and 3, respectively) on recent and initial results obtained in the Aachen research group. The results on memory reduction are due to Michael Holtmann and Christof Löding [7,6], the results on minimizing waiting times in request-response games to Florian Horn, Nico Wallmeier, and the author [11,8].

2 Reducing Memory in Winning Strategies

It is well-known that regular winning conditions are reducible to a format that is called "Muller condition". A game with a Muller winning condition is given by a pair $(G, \mathcal{F})$ where G is a finite game arena (or: game graph), say, over the set Q, and $\mathcal{F}$ is a family of subsets of Q. A play $\varrho \in Q^\omega$ is declared as won by player 0 (playing against player 1) according to the Muller condition if the set $\mathrm{Inf}(\varrho)$ of states visited infinitely often in ϱ belongs to $\mathcal{F}$. As shown in [3], a lower bound for the memory size of finite-state winning strategies in Muller games is given by the factorial function (in the number of vertices of the game graph). An exponential lower bound is known for the so-called weak Muller games, which are also presented as pairs $(G, \mathcal{F})$ and in which a play is won by player 0 if the "occurrence set" $\mathrm{Occ}(\varrho)$ of states visited in ϱ belongs to $\mathcal{F}$.

The known solutions of (Muller and weak Muller) games yield finite-state strategies as Mealy automata that can then be minimized by classical techniques. A serious disadvantage of this method is the fact that the initial Mealy automaton is not constructed in a way that supports memory reduction. There are examples where a given finite-state winning strategy is minimal (i.e., cannot be reduced w.r.t. number of states), but where another winning strategy exists with exponentially less states. The main (and still open) problem is to have an overview of all possible winning strategies and a method to single out the "small" ones.

In their recent work [7,6] Holtmann and Löding developed a technique which achieves partial progress in the search for memory-optimal strategies. The main idea is to do some preprocessing of the game arena – reducing it in size – so that the subsequent application of the standard algorithms leads to winning strategies with smaller memory.

An efficient preprocessing turns out to be possible for the case of weak Muller games. We describe the technique for this example; other cases such as Muller games and Streett games are treated in ongoing work (including experimental studies).

The first step in the algorithm is to reduce a weak Muller game to a weak parity game. (The weak parity condition refers to a coloring of states by natural numbers, and it requires that the highest used color in a play is even.) This reduction involves an expansion of the state space of the given arena G to a new arena G': From the state set Q of G we proceed to $2^Q \times Q$. Intuitively, each play ϱ over Q is mapped to a new play ϱ' over $2^Q \times Q$ where, at each moment t, $\varrho'(t)$ is the pair consisting of (1) the set of previously visited states in ϱ and (2) the current state $\varrho(t)$. The first component is thus also called the "appearance record" (of visited states). An important property of the appearance record over Q is the fact that for each weak Muller condition over Q one can build a finite-state winning strategy with appearence records as memory states (if a winning strategy exists at all).

The second (and essential) step is a minimization of the resulting game graph G' over $2^Q \times Q$ with the weak parity winning condition. The game graph G' is converted into a weak parity automaton $\mathcal{A}$ (with state space $2^Q \times Q$ and input alphabet Q) accepting precisely those plays ϱ over G that satisfy the given weak Muller condition. A variant of the Löding's efficient minimization of weak automata [9] yields a reduced automaton $\mathcal{A}'$ which accepts the same plays as $\mathcal{A}$; this automaton is then converted back into a weak parity game (over a quotient graph G'' of G'). The specific structure of game graphs (with vertices attributed to the two players, which deviates from the format of acceptors) requires some technical work in order to be able to switch from games to automata and back. In particular, the form of the state equivalence that leads to a state space reduction has to be designed appropriately.

The approach of optimizing winning strategies via a reduction of game graphs can result in an exponential improvement of the size of finite-state strategies; a family of examples is given in [6].

The main point for the practical applicability of this approach is the availability of efficient minimization (or reduction) procedures for ω-automata. In the case of deterministic weak parity automata, as needed above, the minimization algorithm of [9] can be used. For more general winning conditions, such as the Büchi condition and the (strong) parity condition, other algorithms have to be used ([4],[5]; see [6]).

Many questions remain open in this field. Already in [2], Büchi and Landweber raise the question whether the space of all (winning) strategies for a given Muller game can be parametrized by data that are derived from the loop structure of a Muller game graph. We do not have as yet any transparent scheme that captures all winning strategies of an infinite game. Another problem is motivated by the fact that many game specifications are given as logical formulas (e.g., formulas of linear-time temporal logic LTL). In this case, the first step is to construct a Muller game which is then treated as discussed above. The interplay between these two steps (introducing states for the Muller game, and introducing further states for its solution in terms of finite-state strategies) is not well understood; it seems reasonable to expect gains in efficiency by integrating these two levels of state space reduction.

3 Minimizing Waiting Times in Liveness Conditions

Many specifications (winning conditions) for infinite games consist of a combination of a safety condition ("all states visited in play should share a certain "good" state property") with liveness conditions of the following form:

($*$) Whenever a P-state is visited, later also an R-state is visited

where P and R are subsets of the vertex set of the game graph (state properties). This corresponds to a situation where a visit to P signals a certain "request" and a visit to R the corresponding "response" (usually meaning that satisfaction of the request is granted). The safety conditions can be captured by a restriction of the game graph. We thus concentrate on the second type of winning condition ($*$). A *request-response condition* is a finite conjunction of conditions of the form ($*$). In linear-time temporal logic this amounts to a conjunction $\bigwedge_{i=0}^{k} \mathrm{G}(p_i \rightarrow \mathrm{XF}\ r_i)$ with formulas p_i, r_i expressing state properties. A request-response game (in short, RR-game) is a finite-state game with a request-response winning condition.

It is not difficult to show that RR-games can be reduced to Büchi games, in which the winning condition just requires to visit states in a certain designated set F infinitely often [12]. Via this reduction (and the solution of Büchi games), the RR-games can be solved, and a finite-state winning strategy can be constructed for the respective winning player. In most practical scenarios, however, the construction of a finite-state strategy which just ensures winning an RR-game is unsatisfactory. As in scheduling problems, one would like to have a solution that minimizes the time lags between visits to a set P and the succeeding visits to R afterwards (referring to a condition ($*$)). An approach that is familiar from "parametrized model-checking" [1] is to bound the time lags by a constant and to check whether this bound is respected for all RR-conditions involved. A corresponding optimization problem asks to compute the minimal possible bound. We pursue here a more ambitious goal, namely to realize an optimization of the waiting times for the individual RR-conditions. Furthermore, we use real numbers as values for optimization; they reflect average waiting times in infinite plays. We obtain associated valuations for winning strategies, and declare a winning strategy as optimal if it realizes the optimal value (if it exists) among all winning strategies.

We discuss here two natural valuations (a more general framework is treated in [8]). Let us first consider just a single RR-condition with sets P, R as above, and assume that a play ϱ satisfies the condition. We count the time lags between "first visits" to P, which we call "activations of waiting" and the successive R-visits. (A visit is "first" if all previous visits to P are already matched by an R-visit.) For each finite play prefix $\varrho[0 \ldots n]$, the average waiting time $w_{\varrho,0}(n)$ is the sum of the waiting times in the prefix $\varrho[0 \ldots n]$ divided by the number of activations.

A weakness of this approach of "linear penalty for waiting" is that it does not distinguish between the following two kinds of plays, with two RR-conditions to

be observed: In the first play, the conditions are met with waiting times 9 and 1, respectively, in the second both conditions are satisfied with waiting time 5 each (and the repetition is assumed to be the same in both cases). The intuitive preference for the second solution (where the maximum waiting time is smaller) is met by introducing the penalty m (rather than 1) for the m-th successive time instance of waiting. This results in a quadratic increase of the penalty during a waiting phase; and we define $w_{\varrho,1}(n)$ as the sum of these penalties for the play prefix $\varrho[0 \dots n]$, again divided by the number of activations.

Let us write $w_\varrho(n)$ for any of these two values $w_{\varrho,i}(n)$. (We suppress the index $i \in \{0,1\}$ for better readability.) The value of a play ϱ is then $v(\varrho) := \limsup_{n\to\infty} w_\varrho(n)$ (which is set to be ∞ if this limit does not exist). If there are k RR-conditions with respective values $w^j_\varrho(n)$ $(j = 1, \dots, k)$, then the value of ϱ is defined as

$$v(\varrho) := \limsup_{n\to\infty} \frac{1}{k} \sum_{j=1}^{k} w^j_\varrho(n)$$

We write $v(\sigma, \tau)$ for the value of the play that is determined by the strategies σ, τ of players 0 and 1, respectively. The value of strategy σ is $v(\sigma) := \sup_\tau v(\sigma, \tau)$. A winning strategy for player 0 is called optimal if there is no other winning strategy for player 0 with smaller value.

A simple example shows that the value derived from linear penalty for waiting does not allow, in general, to construct optimal winning strategies that are finite-state. However, in the case of quadratic penalty for waiting, an optimal finite-state winning strategy exists, which moreover can be effectively computed ([11,8]). The key fact to establish this result is a lemma that ensures a uniform bound on the waiting times for an optimal winning strategy. An optimal finite-state strategy can then be computed via a reduction to the solution of mean-payoff games [14].

This study addresses games where the payoff of a play is different from 0 or 1. This view is standard, for example, in mean-payoff games and stochastic games. Here we apply it to discrete infinite games, extract continuous parameters, and referring to them compute again discrete optimal objects (finite-state strategies).

4 Conclusion

We discussed two problems of optimization in solving infinite games with regular winning conditions. While the two problems are quite different in nature, they point to a shift in the analysis of games. The origins of the theory of infinite games are descriptive set theory and questions of mathematical logic, in which the existence of winning strategies is the main issue. The refined studies in theoretical computer science (and their applications in system design) naturally lead to modifications of the objectives. The present paper illustrates this by studies on quantitative classifications of winning strategies.

References

1. Alur, R., Etessami, K., La Torre, S., Peled, D.: Parametric temporal logic for "model measuring". ACM Trans. Comput. Logic 2, 388–407 (2001)
2. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Trans. Amer. Math. Soc. 138, 367–378 (1969)
3. Dziembowski, S., Jurdziński, M., Walukiewicz, I.: How much memory is needed to win infinite games? In: Proc. 12th IEEE Symp. on Logic in Computer Science, pp. 99–110. IEEE Computer Society Press, Los Alamitos (1997)
4. Etessami, K., Wilke, Th., Schuller, R.A.: Fair bismulation relations, parity games, and state space reduction for Büchi automata. SIAM J. Comput. 34, 1159–1175 (2005)
5. Fritz, C., Wilke, Th.: Simulation relations for alternating parity automata and parity games. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 59–70. Springer, Heidelberg (2006)
6. Holtmann, M., Löding, C.: Memory reduction for strategies in infinite games. In: Holub, J., Žd'árek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 253–264. Springer, Heidelberg (2007)
7. Holtmann, M.: Memory Reduction for Strategies in Infinite Games, Diploma thesis, RWTH Aachen 2007, http://www.automata.rwth-aachen.de/~holtmann/
8. Horn, F., Thomas, W., Wallmeier, N.: Optimal strategy synthesis for request-response games (submitted)
9. Löding, C.: Efficient minimization of deterministic weak ω-automata. Inf. Proc. Lett. 79, 105–109 (2001)
10. Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 1–13. Springer, Heidelberg (1995)
11. Wallmeier, N.: Strategien in unendlichen Spielen mit Liveness-Gewinnbedingungen: Syntheseverfahren, Optimierung und Implementierung, Dissertation, RWTH Aachen (2007)
12. Wallmeier, N., Hütten, P., Thomas, W.: Symbolic synthesis of finite-state controllers for request-response specifications. In: Ibarra, O.H., Dang, Z. (eds.) CIAA 2003. LNCS, vol. 2759, pp. 11–22. Springer, Heidelberg (2003)
13. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci. 200, 135–183 (1998)
14. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comput. Sci. 158, 259–343 (1996)

Recursive Domain Equations of Filter Models

Fabio Alessi and Paula Severi

Università degli Studi di Udine
Dipartimento di Matematica ed Informatica
via delle Scienze 208, 33100 Udine, Italy

Abstract. Filter models and (solutions of) recursive domain equations are two different ways of constructing lambda models. Many partial results have been shown about the equivalence between these two constructions (in some specific cases). This paper deepens the connection by showing that the equivalence can be shown in a general framework. We will introduce the class of disciplined intersection type theories and its four subclasses: *natural split*, *lazy split*, *natural equated* and *lazy equated*. We will prove that each class corresponds to a different recursive domain equation. For this result, we are extracting the essence of the specific proofs for the particular cases of intersection type theories and making one general construction that encompasses all of them. This general approach puts together all these results which may appear scattered and sometimes with incomplete proofs in the literature.

1 Introduction

This paper is concerned with two different ways of constructing models of the untyped lambda calculus which are strongly related.

1. *Scott's D_∞-models.* They were introduced in the 70's as solutions of the recursive domain equation $D = [D \to D]$ [19]. Their construction depends on an initial domain D_0 and an initial embedding i_0. Variations of D_0 and i_0 define different D_∞-models.
2. *Filter models.* They were introduced in the 80's using the notion of type as the elementary brick for their construction [5]. We start from an extension of the simply typed lambda calculus with intersection types and subtyping. Then, the interpretation of a λ-term is defined as the set of its types and it has the property of being a filter. The so-called intersection type theory $\mathcal{T}$ is a set of subtyping statements $A \leq B$. Variations on the intersection type theory $\mathcal{T}$ induce different filter lambda models denoted by $\mathcal{F}^{\mathcal{T}}$.

Many partial results have been shown about the correspondence between both constructions (in some specific cases). Historically, some instances of D_0 and i_0 have given rise to some specific filter models such as Scott and Park lambda models [19,16]. This result has been generalized [5,8,9,3] by showing that in the category of ω-algebraic complete lattices, any D_∞-model can be described as a filter model by coding the compact elements of D_0 as types and defining an

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 124–135, 2008.

intersection type theory $\mathcal{T}$ that contains both the order of D_0 and i_0[1]. The converse is obviously not true. Filter models are in a sense weaker structures than D_∞-models. Not all of them satisfy the recursive domain equation $D = [D \to D]$. If we restrict our attention to the extensional filter models, then some of them have been described as D_∞-models by choosing an appropriate D_0 and i_0 [7,8,2,10]. In spite of the fact that the non-extensional filter models do not satisfy $D = [D \to D]$, in some cases it is possible to find other recursive domain equations for them [2]. For instance, the non-extensional filter model for the lazy λ-calculus [1] satisfies the equation $D = [D \to D]_\perp$, while the filter model of [5] satisfies the equation $D = E \times [D \to D]$ for a suitable E. In some cases [8], it is not clear if a filter model satisfy any domain equation at all.

In this paper we show that the connection between filter models and (solutions of) domain equations can be shown in a general framework. We characterize the classes of filter models that correspond to each recursive domain equation as shown in Fig. 1. Many filter models in the literature belong to one of these four classes. Our classification will be done by analyzing patterns of axioms in the definition of the subtyping relation inducing the filter model. In order to give a proper formalization for the study of these patterns of axioms, we will define the subtyping parametric on a set Σ of axioms. The judgements are of the form $\Sigma \vdash A \leq B$. This defines the *intersection type theory* $\mathcal{T}^\Sigma$*, or just* $\mathcal{T}$*, generated by* Σ [8]. Depending on the shape of the axioms Σ, we will introduce four classes of type theories: *natural split*, *lazy split*, *natural equated* and *lazy equated* (Section 3). Each class corresponds to a different recursive domain equation in the following sense: the intersection type theory $\mathcal{T}$ belongs to the class iff the filter model $\mathcal{F}^\mathcal{T}$ satisfies the corresponding recursive domain equation. This result will be a consequence of a stronger and more refined theorem which states the correspondence between intersection type theories and triples $\langle F, D_0, i_0 \rangle$ where F is a functor, D_0 an object and i_0 an embedding in the category of complete ω-algebraic lattices. We will specify in each of the four cases how to construct $\mathcal{F}^\mathcal{T}$ as a colimit of ω-chains starting from certain $\langle F, D_0, i_0 \rangle$ (Section 4).

The contribution of this paper can be summarized as follows.

1. We introduce the class of disciplined type theories and its four subclasses: natural split, lazy split, natural equated and lazy equated.
2. In Theorem 27, we prove that filter models over any disciplined intersection type theory can be constructed as the colimit of certain ω-chain. In Theorem 30, we prove the converse.
3. As corollaries of those two theorems, we deduce the correspondence between intersection type theories and recursive domain equations shown in Fig. 1.
4. In the proofs of Theorem 27 and Theorem 30, we are extracting the essence of the specific proofs for Scott, Park, CDZ, DHM and HR and making one general construction that encompasses all these particular cases. This general approach puts together all these results which may appear scattered and sometimes with incomplete proofs in the literature.

[1] In the categories of Scott domains or stable sets, D_∞-models cannot be captured in their full generality.

Class	Equation
Natural split	$D = E \times [D \to D]$
Lazy split	$D = E \times [D \to D]_\perp$
Natural equated	$D = [D \to D]$
Lazy equated	$D = [D \to D]_\perp$

Fig. 1. Classification of recursive domain equations

2 Domain-Theoretic Preliminaries

This auxiliary section is devoted to recall some definition and facts from lattice theory of some importance in the following sections. For fundamental notions and results on lattices we refer to [13].

Definition 1. *1. Let $D = \langle D, \sqsubseteq \rangle$ be a complete lattice. A subset $Z \subseteq D$ is* directed *if it is non-empty and for any $z, z' \in Z$ there exists $z'' \in Z$ such that $z, z' \sqsubseteq z''$.*

2. A monotone function $f : D \to E$ is continuous *if for any directed $Z \subseteq D$, we have that $f(\bigsqcup Z) = \bigsqcup f(Z)$. The space of continuous functions from D to E, ordered with the pointwise ordering, is denoted by $[D \to E]$.*

3. An element $d \in D$ is compact *if for any directed $Z \subseteq D$, $d \sqsubseteq \bigsqcup Z$ implies that there exists $z \in Z$ such that $d \sqsubseteq z$. The set of compact elements of D is denoted by $\mathcal{K}(D)$.*

*4. D is an ω-*algebraic *lattice if $\mathcal{K}(D)$ is countable and moreover, for any $x \in D$, $x = \bigsqcup\{d \in \mathcal{K}(D) \mid d \sqsubseteq x\}$. **ALG** is the category of complete ω-algebraic lattices and continuous functions (see [17]).*

5. Given two ω-algebraic lattices D and E, and two compact elements $d \in \mathcal{K}(D)$, $e \in \mathcal{K}(E)$, we define the step function

$$(d \Rightarrow e)(x) = \begin{cases} e & \text{if } d \sqsubseteq x \\ \perp & \text{otherwise} \end{cases}$$

We have that $(d \Rightarrow e) \sqsubseteq f$ if and only if $e \sqsubseteq f(d)$. Hence $d \Rightarrow e \sqsubseteq d' \Rightarrow e'$ if and only if $d' \sqsubseteq d$ and $e \sqsubseteq e'$. Finite sups of step functions are the compact elements in $[D \to E]$.

Lemma 2. *Let $D, E \in$ **ALG**. Then,*

1. $\mathcal{K}([D \to E]) = \{\bigsqcup_{i=1}^n (d_i \Rightarrow e_i) \mid n \in N \ \&\ d_i, e_i \text{ compact } \ \&\ 1 \leq i \leq n\}$.
2. $[D \to E]$ *is ω-algebraic.*
3. $\bigsqcup_{j=1}^n (c_j \Rightarrow d_j) \sqsubseteq \bigsqcup_{i=1}^n (a_i \Rightarrow b_i) \quad \Leftrightarrow$
   $\forall j \in \{1 \ldots n\}. d_j \sqsubseteq \bigsqcup\{b_i \mid 1 \leq i \leq n \ \&\ a_i \sqsubseteq c_j\}$.

Morever **ALG** *is a CCC (cartesian closed category) with "enough points".*

Definition 3. *Let $i : D \to E$, $j : E \to D$ be continuous functions. We say that $\iota = \langle i, j\rangle : D \to E$ is an* embedding-projection pair *(ep for short) if $j \circ i = Id_D$ and $i \circ j \sqsubseteq Id_E$.*
If $\langle i, j\rangle : D \to E$ and $\langle h, k\rangle : E \to E'$, then $\langle i, j\rangle \circ \langle h, k\rangle = \langle h \circ i, j \circ k\rangle$.
$\mathbf{ALG}^E$ *is the category of ω-algebraic lattices and ep's.*

Next lemma on ep's is very useful. Its proof can be recovered by using the results of Section 0-3 of [13] on basic properties of Galois connections.

Lemma 4. *Let $D, E \in \mathbf{ALG}$ and $\iota = \langle i, j\rangle : D \to E$ be an ep.*

1. *$\forall x \in D, y \in E.\ i(x) \sqsubseteq y \ \Leftrightarrow x \sqsubseteq j(y)$.*
   j is the right adjoint of i and it is often denoted by i^R.
2. *ι is completely determined by the embedding i, since j is forced to satisfy the following equality:*
$$(\dagger)\ j(y) = \bigsqcup\{x \mid i(x) \sqsubseteq y\}$$
3. *i is additive, injective and preserves compact elements.*

Thanks to Lemma 4(2), we identify an ep $\iota = \langle i, i^R\rangle$ with its embedding i.

Definition 5. *Let $F : \mathbf{ALG}^E \to \mathbf{ALG}^E$ be a locally continuous endofunctor*[2]*, $D_0 \in \mathbf{ALG}$ and $i_0 : D_0 \to F(D_0)$ be an embedding in* **ALG**.

1. *The triple $\rho = \langle F, D_0, i_0\rangle$ is called a specification of a colimit.*
2. *Every specification $\rho = \langle F, D_0, i_0\rangle$, induces an ω-chain*
$$D_0 \overset{i_0}{\to} F(D_0) \overset{F(i_0)}{\to} F^2(D_0) \overset{F^2(i_0)}{\to} F^3(D_0) \ldots$$
   that we call the ω-chain of ρ.
3. *Then, D_0 is called the* initial domain *and i_0 the* initial embedding.
4. *The colimit of ρ, denoted by colim(ρ), is the colimit of the ω-chain of ρ.*

3 Intersection Type Theories and Filter Models

In this section we recall the notion of intersection type theory and its induced filter model. We will consider intersection type theories generated by a set Σ of axioms [8]. Depending on the shape of the axioms Σ, we introduce four classes of type theories: natural split, lazy split, natural equated and lazy equated. We show that many of the examples of intersection type theories that appear in the literature fit in one of these four patterns of axioms. In this section we will also show that any of these four classes behaves well in the sense that they all induce reflexive filter structures and, hence, λ-models.

Definition 6. *Let $\mathbb{A}$ be a countable set of symbols, called* atoms.

[2] See [20]: what matters here is that F locally continuous implies that it commutes with colimits in $\mathbf{ALG}^E$ and the domain equation $X = F(X)$ has solution in **ALG**.

1. *We assume there is a special atom* $\top$ *in* $\mathbb{A}$*, called* top.
2. *The set* $\mathbb{B}(\mathbb{A})$ *(or just* $\mathbb{B}$*) of* basic types *over* $\mathbb{A}$ *is defined by* $\mathbb{B} = \mathbb{A} \mid \mathbb{B} \cap \mathbb{B}$.
3. *The set* $\mathbb{T}(\mathbb{A})$ *(or just* $\mathbb{T}$*) of* types *over* $\mathbb{A}$ *is defined by* $\mathbb{T} = \mathbb{A} \mid \mathbb{T} \to \mathbb{T} \mid \mathbb{T} \cap \mathbb{T}$.

Greek letters $\alpha, \beta, \ldots$ range over $\mathbb{A}$ and $A, B, \ldots$ range over $\mathbb{B}$ or $\mathbb{T}$.
Next definition is standard [5] except for the axiom $(\top_{\text{lazy}})$. The intuition behind this last axiom is that anything that is more defined than a function is still a function. Note that we can deduce that $A \to \top = \top \to \top$.

Definition 7. *An* intersection type theory $\mathcal{T}$ *is a set of statements of the form* $A \leq B$ *(to be read:* A *is a subtype of* B*), with* $A, B \in \mathbb{T}$*, that satisfies the following axioms and rules.*

(refl) $A \leq A$ (incl_L) $A \cap B \leq A$ (incl_R) $A \cap B \leq B$ (trans) $\dfrac{A \leq B \quad B \leq C}{A \leq C}$ (glb) $\dfrac{C \leq A \quad C \leq B}{C \leq A \cap B}$	($\top$) $A \leq \top$ ($\top_{\text{lazy}}$) $A \to \top \leq \top \to \top$
($\to\cap$) $(A \to B) \cap (A \to C) \leq A \to B \cap C$	($\to$) $\dfrac{A' \leq A \quad B \leq B'}{A \to B \leq A' \to B'}$

We write $A = B$ (to be read as "A is equivalent to B") for $A \leq B \in \mathcal{T}$ and $B \leq A \in \mathcal{T}$. We extend $\leq, \cap$ and $\to$ to $\mathbb{T}/=$ in the obvious way. The equivalence class of a type A is denoted as $[A]$. Syntactic identity is denoted as $A \equiv B$.

Intersection is associative and commutative with respect to equivalence of types, so if $n \geq 1$, we may write $\bigcap_{i=1}^{n} A_i$ or $\bigcap\{A_1, \cdots, A_n\}$ for $A_1 \cap \cdots \cap A_n$. The case $\bigcap \emptyset$ denotes $\top$. If $n \geq 1$, we may write $\bigcap_{i=1}^{n} A_i$ or $\bigcap\{A_1, \cdots, A_n\}$ for $A_1 \cap \cdots \cap A_n$. The empty intersection $\bigcap \emptyset$ denotes $\top$. In the literature, the top $\top$ is denoted by different symbols such as ω in [6] or Ω in [11].

We now recall the definition of filter structure [5,8].

Definition 8. *Let* $\mathcal{T}$ *be an intersection type theory.*

1. *A* filter *(resp.* basic filter*) is a set* $X \subseteq \mathbb{T}$ *(resp.* $X \subseteq \mathbb{B}$*) such that:*
   (a) $\top \in X$*;*
   (b) $A \leq B$*(resp.*$A \leq_{\mathcal{B}} B$*) and* $A \in X$ *imply* $B \in X$*;*
   (c) $A \in X$ *and* $B \in X$ *imply* $A \cap B \in X$*.*
2. *Let* $Y \subseteq \mathbb{T}$*. Then,* $\uparrow Y$ *denotes the filter generated by* Y*. If* $Y = \{A\}$*, we write* $\uparrow A$ *instead of* $\uparrow\{A\}$ *and* $\uparrow A$ *is called a* principal *filter. Actually it coincides with the upper closure of* A*, i.e.* $\uparrow A = \{B \mid A \leq B\}$*. If* $Y \subseteq \mathbb{B}$ *the basic filter generated by* Y *is denoted by* $\uparrow^{\mathcal{B}} Y$*. Similarly, we write* $\uparrow^{\mathcal{B}} A$ *instead of* $\uparrow^{\mathcal{B}} \{A\}$ *for the principal filter.*
3. $\mathcal{F}^{\mathcal{T}}$ *(resp.* $\mathcal{F}^{\mathcal{B}}$*) is the set of filters (resp. basic filters) over* $\mathcal{T}$*, ordered by set-theoretic inclusion, and is called the* filter structure *(resp.* basic filter structure*) over* $\mathcal{T}$*.*

It is well-known that $\mathcal{F}^{\mathcal{T}}$ is an ω-algebraic lattice [8]. Given $\mathcal{X} \subseteq \mathcal{F}^{\mathcal{T}}$

$$(\text{fil-sup}) \bigsqcup \mathcal{X} = \uparrow \{\bigcap_{j=1}^{n} A_j \mid n \in I\!N, \forall 1 \leq j \leq n. \exists X \in \mathcal{X}. A_j \in X\},$$

Moreover $X \sqcap Y = X \cap Y$, the bottom filter is $\uparrow\top$, the top filter is $\mathbb{T}$. Compact elements in $\mathcal{F}^{\mathcal{T}}$ are the principal filters and they inherit the order $\leq^{op}$:

$$\uparrow A \subseteq \uparrow B \Leftrightarrow B \leq A$$

In the following definition we show how to interpret the untyped lambda calculus in a filter structure $\mathcal{F}^{\mathcal{T}}$.

Definition 9. *Let $\mathcal{T}$ be an intersection type theory.*

1. *We define $\mathsf{App}^{\mathcal{T}} : \mathcal{F}^{\mathcal{T}} \to [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]$ and $\mathsf{Abs}^{\mathcal{T}} : [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}] \to \mathcal{F}^{\mathcal{T}}$ as follows.*
$$\begin{array}{ll} \mathsf{App}^{\mathcal{T}}(X)(Y) & = \{B \mid \exists A \in Y. A \to B \in X\} \\ \mathsf{Abs}^{\mathcal{T}}(f) & = \uparrow\{A \to B \mid B \in f(\uparrow A)\} \end{array}$$
2. *We define an interpretation on λ-terms as follows [8,3].*
$$\begin{array}{ll} [\![x]\!]_\rho & = \rho(x) \\ [\![MN]\!]_\rho & = (\mathsf{App}^{\mathcal{T}}([\![M]\!]_\rho))([\![N]\!]_\rho) \\ [\![\lambda x.M]\!]_\rho & = \mathsf{Abs}^{\mathcal{T}}(\boldsymbol{\lambda} d \in \mathcal{F}^{\mathcal{T}}. [\![M]\!]_{\rho[x/d]}) \end{array}$$

Not any filter structure $\mathcal{F}^{\mathcal{T}}$ gives rise to a lambda model $\langle \mathcal{F}^{\mathcal{T}}, \mathsf{App}^{\mathcal{T}}, \mathsf{Abs}^{\mathcal{T}}, [\![\_]\!] \rangle$, but if the definition of an intersection type theory satisfy some restrictions, not only the induced filter structure will be a λ-model, but also the connection with suitable colimits emerges rather clearly. Importantly, the restrictions we put in Definition 12 are easily satisfied by the main intersection type theories in the literature. Before that we have to introduce the notion of specification of axioms.

Definition 10. *A* specification of axioms *is a pair $\Sigma = (<_{\mathcal{B}}, \mathsf{def})$ where $<_{\mathcal{B}}$ is a partial order on $\mathbb{B}$ such that $\cap$ is the meet and $\top$ is the top and* def *is a function from $\mathbb{A}$ to $\mathbb{T}$. The function* def *is extended from $\mathbb{A}$ to $\mathbb{T}$ by $\mathsf{def}(A \cap B) = \mathsf{def}(A) \cap \mathsf{def}(B)$ and $\mathsf{def}(A \to B) = A \to B$.*

Definition 11. *Let $\Sigma = (<_{\mathcal{B}}, \mathsf{def})$ be a specification of axioms. The* intersection type theory generated by Σ, *denoted by $\mathcal{T}^{\Sigma}$ or just $\mathcal{T}$, derives judgements of the form $A \leq B \in \mathcal{T}^{\Sigma}$ or $\Sigma \vdash A \leq B$ and it is defined as the smallest intersection type theory that contains the following two axioms.*

$$(\mathcal{B}\text{-ax}) \ \frac{A \leq_{\mathcal{B}} B}{\Sigma \vdash A \leq B} \qquad (\mathsf{def}\text{-ax}) \ \Sigma \vdash \alpha = \mathsf{def}(\alpha)$$

We write $\Sigma \vdash A \leq B$ if $A \leq B \in \mathcal{T}^{\Sigma}$. If there is little danger of confusion, when Σ or $\mathcal{T}$ are clear from the context, then we will just write $(A \leq B)$.

Figure 2 shows how many of the intersection type theories that appear in the literature can be generated by a specification $\Sigma = (\leq_{\mathcal{B}}, \mathsf{def})$. For the $\mathcal{B}$-axioms,

$\mathcal{T}$	$\mathbb{A}$	$\mathcal{B}$-axioms $\leq_{\mathcal{B}}$	def-axioms $\alpha = \mathsf{def}(\alpha)$
Scott [19]	$\{\mathtt{0}, \top\}$	$\mathtt{0} \leq_{\mathcal{B}} \top$	$\mathtt{0} = \top \to \mathtt{0}, \top = \top \to \top$
Park [16]	$\{\mathtt{0}, \top\}$	$\mathtt{0} \leq_{\mathcal{B}} \top$	$\mathtt{0} = \mathtt{0} \to \mathtt{0}, \top = \top \to \top$
CDZ [9]	$\{\mathtt{0}, \mathtt{1}, \top\}$	$\mathtt{0} \leq_{\mathcal{B}} \mathtt{1} \leq_{\mathcal{B}} \top$	$\mathtt{0} = \mathtt{1} \to \mathtt{0}, \mathtt{1} = \mathtt{0} \to \mathtt{1}, \top = \top \to \top$
HR [15]	$\{\mathtt{0}, \mathtt{1}, \top\}$	$\mathtt{0} \leq_{\mathcal{B}} \mathtt{1} \leq_{\mathcal{B}} \top$	$\mathtt{0} = \mathtt{1} \to \mathtt{0}, \mathtt{1} = (\mathtt{0} \to \mathtt{0}) \cap (\mathtt{1} \to \mathtt{1}), \top = \top \to \top$
DHM [12]	$\{\mathtt{0}, \mathtt{1}, \top\}$	$\mathtt{0} \leq_{\mathcal{B}} \mathtt{1} \leq_{\mathcal{B}} \top$	$\mathtt{0} = \top \to \mathtt{0}, \mathtt{1} = \mathtt{0} \to \mathtt{1}, \top = \top \to \top$
BCD [5]	$\mathbb{A}^{\infty}$	$\mathtt{c}_i \leq_{\mathcal{B}} \top$	$\mathtt{c}_i = \mathtt{c}_i, \top = \top \to \top$
AO [1]	$\{\top\}$		$\top = \top$

Fig. 2. Intersection type theories generated by $\Sigma = (\leq_{\mathcal{B}}, \mathsf{def})$

we do not specify the whole set $\leq_{\mathcal{B}}$. Each $\leq_{\mathcal{B}}$ is actually defined as the least partial order that contains the pairs $A \leq_{\mathcal{B}} B$ shown in the table. We define a countable set of constants $\mathbb{A}^{\infty} = \{\mathtt{c}_i \mid i \in N\} \cup \{\top\}$.

Definition 12. *Let $\Sigma = (<_{\mathcal{B}}, \mathsf{def})$ be a specification of axioms.*

1. *We say that Σ (and also $\mathcal{T}^{\Sigma}$) is* lazy *if* $\mathsf{def}(\top) = \top$.
2. *We say that Σ (and also $\mathcal{T}^{\Sigma}$) is* natural *if* $\mathsf{def}(\top) = \top \to \top$.
3. *We say that Σ (and also $\mathcal{T}^{\Sigma}$) is* split *if the following two conditions hold.*
   (a) $\mathbb{A} - \{\top\} \neq \emptyset$.
   (b) $\mathsf{def}(\alpha) = \alpha$, $\forall\ \alpha \in \mathbb{A} - \{\top\}$.
4. *We say that Σ (and also $\mathcal{T}^{\Sigma}$) is* equated *if the following conditions hold.*
   (a) $\forall \alpha \in \mathbb{A} - \{\top\}$, $\exists A_1, \ldots A_n \in \mathbb{B}$, $B_1, \ldots B_n \in \mathbb{B} - \{\top\}$,

$$\mathsf{def}(\alpha) \equiv \bigcap_{i=1}^{n} (A_i \to B_i);$$

   (b) $\forall A, B \in \mathbb{B} - \{\top\}$,

$$A \leq_{\mathcal{B}} B \Leftrightarrow \forall j \in \{1 \ldots n\}. \bigcap \{B_i \mid 1 \leq i \leq m \ \&\ A_i \geq_{\mathcal{B}} C_j\} \leq_{\mathcal{B}} D_j$$

   where $\mathsf{def}(A) \equiv \bigcap_{i=1}^{m} (A_i \to B_i)$ *and* $\mathsf{def}(B) \equiv \bigcap_{j=1}^{n} (C_j \to D_j)$.
5. *We say that Σ (and also $\mathcal{T}^{\Sigma}$) is* disciplined *if it is either one of these four possible combinations: natural split, lazy split, natural equated or lazy equated.*

The examples of intersection type theories given in Figure 2 are classified according to the four classes defined above as follows.

Natural split	BCD
Lazy split	none
Natural equated	Scott, Park, CDZ, HR, DHM
Lazy equated	AO

[3] It coincides with the interpretation defined through the *type assignment system* [8].

Lemma 13. *In any intersection type theory $\mathcal{T}$, the following statements hold:*

1. *If $A \leq C$ and $B \leq D$ then $A \cap B \leq C \cap D$.*
2. *$(A_1 \to B_1) \cap \ldots \cap (A_n \to B_n) \leq (A_1 \cap \ldots \cap A_n) \to (B_1 \cap \ldots \cap B_n)$.*
3. *If $\forall j \in \{1 \ldots n\}. \bigcap\{B_i \mid 1 \leq i \leq m \ \&\ A_i \geq C_j\} \leq D_j$ then*

$$\bigcap_{i=1}^{m}(A_i \to B_i) \leq \bigcap_{j=1}^{n}(C_j \to D_j).$$

Definition 14. *We define two functions to extract the sets of outermost atoms and arrows of a type.*

$$\begin{array}{ll} \mathsf{ats}(\alpha) = \{\alpha\} & \mathsf{ars}(\alpha) = \emptyset \\ \mathsf{ats}(A \to B) = \emptyset & \mathsf{ars}(A \to B) = \{A \to B\} \\ \mathsf{ats}(A \cap B) = \mathsf{ats}(A) \cup \mathsf{ats}(B) & \mathsf{ars}(A \cap B) = \mathsf{ars}(A) \cup \mathsf{ars}(B) \end{array}$$

Theorem 15. (Conservativity of $\leq_{\mathcal{B}}$). *Let $A, B \in \mathbb{B}$ and Σ be disciplined. Then, $A \leq_{\mathcal{B}} B$ iff $\Sigma \vdash A \leq B$.*

Since $\leq_{\mathcal{B}}$ is antisymmetric, we have the following:

Corollary 16. *If $\Sigma \vdash \alpha = \beta$ then $\alpha \equiv \beta$.*

The converse of Lemma 13 part 3 is an important property which is not always true. We will see later that it is a sufficient and necesary condition for having a reflexive filter structure. For this, we will define the following notion.

Definition 17. *We say that $\mathcal{T}$ is β-sound if*

$$\textstyle\bigcap_{i=1}^{m}(A_i \to B_i) \leq (C \to D) \Rightarrow \bigcap\{B_i \mid 1 \leq i \leq m \ \&\ A_i \geq C\} \leq D$$

$\forall A_i, B_i, C, D \in \mathbb{T}$ with $1 \leq i \leq m$.

A particular case of β-soundness is the invertibility of the rule $(\to)$, i.e. if $A \to B \leq C \to D$ then $A \geq C$ and $B \leq D$.

Theorem 18. *Let $\mathcal{T}$ be disciplined. If $E \leq F$ then*

1. *$\bigcap \mathsf{ats}(\mathsf{def}(E)) \leq \bigcap \mathsf{ats}(\mathsf{def}(F))$*
2. *$\forall j \in \{1 \ldots n\}. \bigcap\{B_i \mid 1 \leq i \leq m \ \&\ A_i \geq C_j\} \leq D_j$ where $\mathsf{ars}(\mathsf{def}(E)) \equiv \{(A_i \to B_i) \mid 1 \leq i \leq m\}$ and $\mathsf{ars}(\mathsf{def}(F)) \equiv \{(C_j \to D_j) \mid 1 \leq i \leq n\}$.*

Proof. We prove it by induction on the derivation of $E \leq F$. The interesting case is ($\mathcal{B}$-ax) which follows from Condition 4b ($\Rightarrow$) in Definition 12. □

Corollary 19. *Let $\mathcal{T}$ be disciplined. Then $\mathcal{T}$ is β-sound.*

Theorem 20. *If $\mathcal{T}$ is disciplined then $\langle \mathcal{F}^{\mathcal{T}}, \mathsf{App}^{\mathcal{T}}, \mathsf{Abs}^{\mathcal{T}}, [\![\_]\!] \rangle$ is a λ-model.*

Proof. Since $\mathcal{T}$ is β-sound, it follows from [8,18] that $\langle \mathcal{F}^{\mathcal{T}}, \mathsf{App}^{\mathcal{T}}, \mathsf{Abs}^{\mathcal{T}}, [\![\_]\!] \rangle$ is a reflexive filter structure, i.e. $\mathsf{App}^{\mathcal{T}} \circ \mathsf{Abs}^{\mathcal{T}} = Id_{[\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]}$. Since any reflexive filter structure is a λ-model [14,4], we conclude that $\mathcal{F}^{\mathcal{T}}$ is also a λ-model. □

4 Classification of Recursive Domain Equations

This section shows that filter structures over disciplined type theories correspond to the colimit of certain ω-chains. Definition 24 gives the correspondence between the intersection type theories $\mathcal{T}$ and the induced specification $\rho^{(\mathcal{T})} = \langle F^{(\mathcal{T})}, D_0^{(\mathcal{T})}, i_0^{(\mathcal{T})} \rangle$ of a colimit. In Theorem 27, we prove that $\mathcal{F}^{\mathcal{T}}$ is isomorphic to the colimit of $\rho^{(\mathcal{T})}$. In Theorem 30 we prove the converse: for any triple $\rho = \langle F, D_0, i_0 \rangle$ of certain class $\mathcal{C}$, it is possible to construct a displined $\mathcal{T}_\rho$ such that $\mathcal{F}^{\mathcal{T}_\rho}$ is isomorphic to the colimit of ρ. As a consequence of these two theorems, we can justify the correspondence between classes and recursive domain equations shown in Figure 1.

Definition 21. *We define four different functors in* $\mathbf{ALG}^E$.

$$\begin{aligned} \mathsf{Hom}^Y(X) &= Y \times [X \to X] \\ \mathsf{Hom}^Y_\perp(X) &= Y \times [X \to X]_\perp \\ \mathsf{Hom}(X) &= [X \to X] \\ \mathsf{Hom}_\perp(X) &= [X \to X]_\perp \end{aligned}$$

with the expected actions on morphisms (for instance, $\mathsf{Hom}(i)(f) = i \circ f \circ j$ *for an embedding projection pair* $\langle i, j \rangle$ *and a continuous function* $f : X \to X$*).*

Definition 22. *Let* $D_0 = \{\perp\}$. *The* trivial embedding $\mathsf{trv} : D_0 \to F(D_0)$ *is defined as* $\mathsf{trv}(\perp) = \perp$.

Definition 23. *Let* Σ *be equated. We define the embedding* $\hat{\mathsf{def}} : \mathcal{F}^{\mathcal{B}} \to [\mathcal{F}^{\mathcal{B}} \to \mathcal{F}^{\mathcal{B}}]$ *on compact elements as follows.*

1. *If* $\mathsf{def}(A) \equiv \bigcap_{i=1}^m (A_i \to B_i)$ *then*
$$\hat{\mathsf{def}}(\uparrow^{\mathcal{B}} A) = \begin{cases} \bigsqcup_{i=1}^m (\uparrow^{\mathcal{B}} A_i \Rightarrow \uparrow^{\mathcal{B}} B_i) & \text{if } \Sigma \text{ is natural} \\ \bigsqcup_{i=1}^m ((\uparrow^{\mathcal{B}} A_i \Rightarrow \uparrow^{\mathcal{B}} B_i), 0) & \text{if } \Sigma \text{ is lazy} \end{cases}$$
2. *If* $\mathsf{def}(\top) = \top$ *then* $\hat{\mathsf{def}}(\uparrow^{\mathcal{B}} \top) = \perp$.

Definition 24. *Let* $\mathcal{T}$ *be disciplined. We define a specification of a colimit according to the following four cases.*

$\mathcal{T}$	$F^{(\mathcal{T})}$	$D_0^{(\mathcal{T})}$	$i_0^{(\mathcal{T})}$
Natural split	$\mathsf{Hom}^{\mathcal{F}^{\mathcal{B}}}$	$\{\perp\}$	trv
Lazy split	$\mathsf{Hom}^{\mathcal{F}^{\mathcal{B}}}_\perp$	$\{\perp\}$	trv
Natural equated	Hom	$\mathcal{F}^{\mathcal{B}}$	$\hat{\mathsf{def}}$
Lazy equated	$\mathsf{Hom}_\perp$	$\mathcal{F}^{\mathcal{B}}$	$\hat{\mathsf{def}}$

Finally, for any $n > 0$, *define inductively* $D_n^{(\mathcal{T})} = F^{(\mathcal{T})}(D_{n-1})$.

Definition 25. *Let* $\mathcal{T}$ *be disciplined. We define* $\mu_n^{(\mathcal{T})} : \mathcal{K}(D_n^{(\mathcal{T})}) \to \mathbb{T}/=$ *by induction as follows.*

1. – *if* $\mathcal{T}$ *is split, then* $\mu_0^{(\mathcal{T})}(\bot) = [\top]$
   – *if* $\mathcal{T}$ *is equated, then* $\mu_0^{(\mathcal{T})}(\uparrow^{\mathcal{B}} A) = [A]$
2. *Let* $n > 0$. *Then*
   – *if* $\mathcal{T}$ *is natural split, then*
   $\mu_n^{(\mathcal{T})}(\uparrow^{\mathcal{B}} A, \bigsqcup_{i=1}^m (d_i \Rightarrow e_i)) = [A] \cap (\bigcap_{i=1}^m (\mu_{n-1}^{(\mathcal{T})}(d_i) \to \mu_{n-1}^{(\mathcal{T})}(e_i))$
   – *if* $\mathcal{T}$ *is lazy split, then*
   $\mu_n^{(\mathcal{T})}(\uparrow^{\mathcal{B}} A, \bot) = [A]$
   $\mu_n^{(\mathcal{T})}(\uparrow^{\mathcal{B}} A, (\bigsqcup_{i=1}^m (d_i \Rightarrow e_i), 0)) = [A] \cap (\bigcap_{i=1}^m (\mu_{n-1}^{(\mathcal{T})}(d_i) \to \mu_{n-1}^{(\mathcal{T})}(e_i))$
   – *if* $\mathcal{T}$ *is natural equated, then*
   $\mu_n^{(\mathcal{T})}(\bigsqcup_{i=1}^m (d_i \Rightarrow e_i)) = \bigcap_{i=1}^m (\mu_{n-1}^{(\mathcal{T})}(d_i) \to \mu_{n-1}^{(\mathcal{T})}(e_i))$
   – *if* $\mathcal{T}$ *is lazy equated, then*
   $\mu_n^{(\mathcal{T})}(\bot) = [\top]$
   $\mu_n^{(\mathcal{T})}(\bigsqcup_{i=1}^m (d_i \Rightarrow e_i), 0) = \bigcap_{i=1}^m (\mu_{n-1}^{(\mathcal{T})}(d_i) \to \mu_{n-1}^{(\mathcal{T})}(e_i))$.

From now on we omit the superscript $(\mathcal{T})$ on μ_n.

Proposition 26. *Let* $\mathcal{T}$ *be disciplined. Then for all* n,

1. $\forall e, e' \in \mathcal{K}(D_n^{(\mathcal{T})})$. $e' \sqsubseteq e \;\Leftrightarrow \mu_n(e) \leq \mu_n(e')$.
2. $\forall d \in \mathcal{K}(D_n^{(\mathcal{T})}).\mu_n(d) = \mu_{n+1}(i_n(d))$ *where* $i_n = F^{(\mathcal{T})(n)}(i_0)$.
3. $\forall A \in \mathbb{T}$. $\exists n \geq 0, a \in \mathcal{K}(D_n^{(\mathcal{T})})$. $\mu_n(a) = [A]$.

Theorem 27. *Let* $\mathcal{T}$ *be disciplined. Then*

$$\mathcal{F}^{\mathcal{T}} \simeq colim(\rho^{(\mathcal{T})})$$

Proof. First define $\bar{\mu}_n : \mathcal{K}(D_n^{(\mathcal{T})}) \to \mathcal{F}^{\mathcal{T}}$, for any $n \geq 0$ and $d \in \mathcal{K}(D_n^{(\mathcal{T})})$ by

$$\bar{\mu}_n(d) = \uparrow [\mu_n(d)]$$

$\bar{\mu}_n$ are monotone by Proposition 26(1). Then the extensions $\tilde{\mu}_n : D_n^{(\mathcal{T})} \to \mathcal{F}^{\mathcal{T}}$ of $\bar{\mu}$ defined by

$$\tilde{\mu}_n(x) = \bigsqcup \{\bar{\mu}_n(d) \mid d \in \mathcal{K}(D_n^{(\mathcal{T})}) \;\&\; d \sqsubseteq x\}$$

are continuous. It follows from Proposition 26(2) that for any $n \geq 0$, $x \in D_n^{(\mathcal{T})}$, $\tilde{\mu}_n(x) = \tilde{\mu}_{n+1}(i_n(x))$. Hence, $\mathcal{F}^{\mathcal{T}}$ together with $\tilde{\mu}_n$ is a cocone for the ω-chain of $\langle F^{(\mathcal{T})}, D_0^{(\mathcal{T})}, i_0^{(\mathcal{T})} \rangle$. To prove that it is initial, consider another cocone, a domain E with $\sigma_n : D_n^{(\mathcal{T})} \to E$ such that $\sigma_{n+1} \circ i_n = \sigma_n$. We define $\theta : \mathcal{F}^{\mathcal{T}} \to E$ on compact elements by $\theta(\uparrow A) = \sigma_n(a)$ where a is an element of some $\mathcal{K}(D_n^{(\mathcal{T})})$ such that $\mu_n(a) = [A]$. This element exists by Proposition 26(3). We prove that θ is monotone. Suppose $\uparrow A \subseteq \uparrow B$, i.e. $B \leq A$. By Proposition 26(2) (3) there exist $n \geq 0$, $a \in \mathcal{K}(D_n^{(\mathcal{T})})$ and $b \in \mathcal{K}(D_n^{(\mathcal{T})})$ such that $\mu_n(a) = [A]$ and $\mu_n(b) = [B]$. It follows from Proposition 26(1) that $a \sqsubseteq b$. Hence $\sigma_n(a) \sqsubseteq \sigma_n(b)$, since σ_n are monotone. By the definition of θ, $\theta(\uparrow A) \sqsubseteq \theta(\uparrow B)$. We have defined θ to have $\theta(\bar{\mu}_n(d)) = \sigma_n(d)$. The continuous extension $\tilde{\theta} : \mathcal{F}^{\mathcal{T}} \to E$ is the unique mediating morphism such that $\forall n.\tilde{\theta} \circ \tilde{\mu}_n = \sigma_n$. □

Corollary 28

1. *If* $\mathcal{T}$ *is natural split then* $\mathcal{F}^{\mathcal{T}} \simeq \mathcal{F}^{\mathcal{B}} \times [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]$.
2. *If* $\mathcal{T}$ *is lazy split then* $\mathcal{F}^{\mathcal{T}} \simeq \mathcal{F}^{\mathcal{T}} \times [\mathcal{F}^{\mathcal{B}} \to \mathcal{F}^{\mathcal{T}}]_{\bot}$.

3. *If $\mathcal{T}$ is natural equated then $\mathcal{F}^{\mathcal{T}} \simeq [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]$.*
4. *If $\mathcal{T}$ is lazy equated then $\mathcal{F}^{\mathcal{T}} \simeq [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]_{\perp}$.*

As expected, we can go in the other direction, proving that a certain kind of colimits could be recovered as filter structures of disciplined intersection type theories.

Definition 29. *We define the sets of triples $\mathcal{C}_i$ ($i \in \{1,2,3,4\}$) as follows:*

$$\begin{aligned}
\mathcal{C}_1 &= \{\langle \mathsf{Hom}^E, \{\perp\}, tr\rangle \mid E \in \mathbf{ALG}\}\\
\mathcal{C}_2 &= \{\langle \mathsf{Hom}^E_{\perp}, \{\perp\}, tr\rangle \mid E \in \mathbf{ALG}\}\\
\mathcal{C}_3 &= \{\langle \mathsf{Hom}, E, i_0\rangle \mid E \in \mathbf{ALG} \;\&\; i_0 : E \to [E \to E]\}\\
\mathcal{C}_4 &= \{\langle \mathsf{Hom}_{\perp}, E, i_0\rangle \mid B \in \mathbf{ALG} \;\&\; i_0 : E \to [E \to E]_{\perp}\}
\end{aligned}$$

We define $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \cup \mathcal{C}_4$.

Theorem 30. *Let $\rho = (F, D_0, i_0) \in \mathcal{C}$. Then there exists a disciplined intersection type theory $\mathcal{T}^{\rho}$ such that*

$$colim(\rho) \simeq \mathcal{F}^{\mathcal{T}^{\rho}}$$

Proof. For each $\rho = \langle F, D_0, i_0\rangle \in \mathcal{C}$, we first define a set $\mathbb{A}^{\rho}$ of atoms.

$$\mathbb{A}^{\rho} = \{\underline{d} \mid d \in \mathcal{K}(B) \;\&\; d \neq \perp\} \cup \{\top\}$$

Then, we define a preorder $\leq_{\mathcal{B}}$ on $\mathbb{B}(\mathbb{A}^{\rho})$ as follows.

$$\begin{array}{c}
\underline{d} \leq_{\mathcal{B}} \underline{e} \Leftrightarrow e \sqsubseteq d\\
\underline{d} =_{\mathcal{B}} \bigcap_{i=1}^{m} \underline{e_i} \Leftrightarrow d = \bigsqcup_{i=1}^{m} e_i\\
\underline{d} \leq \top
\end{array}$$

For the cases where the embeddings i_0 are non-trivial, we define def as follows.

$$\mathsf{def}(\underline{d}) = \textstyle\bigcap_{i=1}^{m}(\underline{a_i} \to \underline{b_i}) \Leftrightarrow i_0(d) = \bigsqcup_{i=1}^{m}(a_i \Rightarrow b_i)$$

It is routine to check that the resulting intersection type theory $\mathcal{T}^{\rho}$ is disciplined in all four cases. Since $B \simeq \mathcal{F}^{\mathcal{B}}$, by Theorem 27 we have that $\mathcal{F}^{\mathcal{T}^{\rho}} \simeq colim(\rho^{\mathcal{T}^{\rho}}) \simeq colim(\rho)$. □

We now prove the converse of Corollary 28. These two corollaries together justify the classification shown in Figure 1.

Corollary 31

1. *If $\mathcal{F}^{\mathcal{T}} \simeq E \times [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]$ then $\exists \mathcal{T}'$ natural split such that $\mathcal{T}/{=} \simeq \mathcal{T}'/{=}$.*
2. *If $\mathcal{F}^{\mathcal{T}} \simeq E \times [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]_{\perp}$ then $\exists \mathcal{T}'$ lazy split such that $\mathcal{T}/{=} \simeq \mathcal{T}'/{=}$.*
3. *If $\mathcal{F}^{\mathcal{T}} \simeq [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]$ then $\exists \mathcal{T}'$ natural equated such that $\mathcal{T}/{=} \simeq \mathcal{T}'/{=}$.*
4. *If $\mathcal{F}^{\mathcal{T}} \simeq [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]_{\perp}$ then $\exists \mathcal{T}'$ natural equated such that $\mathcal{T}/{=} \simeq \mathcal{T}'/{=}$.*

Proof. We prove only the third case. The rest is similar. Take $\rho = (\mathsf{Hom}, \mathcal{F}^{\mathcal{T}}, i_0)$ where i_0 is the isomorphism from D to $[D \to D]$. By the previous theorem, we have that $\mathcal{F}^{\mathcal{T}} \simeq colim(\rho) \simeq \mathcal{F}^{\mathcal{T}'}$ for some $\mathcal{T}'$ natural equated. Since $\mathcal{K}(\mathcal{F}^{\mathcal{T}}) \simeq \mathcal{T}/{=}$ and $\mathcal{K}(\mathcal{F}^{\mathcal{T}'}) \simeq \mathcal{T}'/{=}$, we conclude that $\mathcal{T}/{=} \simeq \mathcal{T}'/{=}$. □

Acknowledgments. Authors thank the referees for many useful suggestions.

References

1. Abramsky, S., Ong, L.C.: Full abstraction in the Lazy Lambda Calculus. Information and Computation 105, 159–267 (1993)
2. Alessi, F.: Strutture di tipi, teoria dei domini, e modelli del λ-calcolo. PhD Thesis. University of Turin (1991)
3. Alessi, F., Dezani-Ciancaglini, M., Honsell, F.: Inverse Limit Models as Filter Models. In: Proceedings of HOR 2004, pp. 3–25 (2004)
4. Barendregt, H.P.: The Lambda Calculus: Its syntax and semantics. North-Holland Publishing co., Amsterdam (1984)
5. Barendregt, H., Coppo, M., Dezani-Ciancaglini, M.: A filter lambda model and the completeness of type assignment. J. Symbolic Logic 48(4), 931–940 (1983)
6. Coppo, M., Dezani-Ciancaglini, M.: An extension of the basic functionality theory for the λ-calculus. Notre Dame J. Formal Logic 21(4), 685–693 (1980)
7. Coppo, M., Dezani-Ciancaglini, M., Longo, G.: Applicative information systems. In: Protasi, M., Ausiello, G. (eds.) CAAP 1983. LNCS, vol. 159, pp. 35–64. Springer, Heidelberg (1983)
8. Coppo, M., Dezani-Ciancaglini, M., Honsell, F., Longo, G.: Extended type structures and filter lambda models. In: Logic Colloquium 1982, pp. 241–262. North-Holland, Amsterdam (1984)
9. Coppo, M., Dezani-Ciancaglini, M., Zacchi, M.: Type theories, normal forms and D_∞ lambda models. Information and Compuation 72(2), 85–116 (1987)
10. Dezani-Ciancaglini, M., Ghilezan, S., Likavec, S.: Behavioural inverse limit models. Theoret. Comput. Sci. 316(1–3), 49–74 (2004)
11. Dezani-Ciancaglini, M., Honsell, F., Alessi, F.: A complete characterization of complete intersection-type preorders. ACM TOCL 4(1), 120–146 (2003)
12. Dezani-Ciancaglini, M., Honsell, F., Motohama, Y.: Compositional characterization of λ-terms using intersection types. Theoret. Comput. Sci. 304(3), 459–495 (2005)
13. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, D.J., Mislove, M.W., Scott, D.: Continuous lattices and domains. Cambridge University Press, Cambridge (2003)
14. Hindley, R., Longo, G.: Lambda calculus models and extensionality. Z. Math Logik Grundlag. Math 26(4), 289–310 (1980)
15. Honsell, F., Ronchi Della Rocca, S.: An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus. J. Comput. System Sci. 45(1), 49–75 (1992)
16. Park, D.: The Y-combinator in Scott's λ-calculus models (revised version). Theory of Computation Report 13, Department of Computer Science, University of Warick (1976)
17. Plotikin, G.: Set-Theoretic and Other Elementary Models of the λ-Calculus. Theoretical Computer Science 121, 351–409 (1993)
18. Ronchi della Rocca, S., Paolini, L.: The Parametric Lambda Calculus. A Metamodel for Computation. In: Texts in Theoretical Computer Science. An EACTS Series, Springer, Heidelberg (2004)
19. Scott, D.S.: Domains for denotational semantics. In: Nielsen, M., Schmidt, E.M. (eds.) Automata, Languages, and Programming. LNCS, vol. 140, pp. 577–613. Springer, Heidelberg (1982)
20. Smyth, M.B., Plotkin, G.D.: The category-theoretic solution of recursive domain equations. SIAM Journal on Computing 11(4), 761–783 (1982)

Algorithmic Problems for Metrics on Permutation Groups

V. Arvind and Pushkar S. Joglekar

Institute of Mathematical Sciences
C.I.T Campus, Chennai 600 113, India
{arvind,pushkar}@imsc.res.in

Abstract. Given a permutation group $G \leq S_n$ by a generating set, we explore MWP (the minimum weight problem) and SDP (the subgroup distance problem) for some natural metrics on permutations. These problems are know to be NP-hard. We study both exact and approximation versions of these problems. We summarize our main results:

- For our upper bound results we focus on the Hamming and the l_∞ permutation metrics. For the l_∞ metric, we give a randomized $2^{O(n)}$ time algorithm for finding an optimal solution to MWP. Interestingly, this algorithm adapts ideas from the Ajtai-Kumar-Sivakumar algorithm for the shortest vector problem in lattices [AKS01]. For the Hamming metric, we again give a $2^{O(n)}$ time algorithm for finding an optimal solution to MWP. This algorithm is based on the classical Schrier-Sims algorithm for finding pointwise stabilizer subgroups of permutation groups.
- It is known that SDP is NP-hard([BCW06]) and it easily follows that SDP is hard to approximate within a factor of $\log^{O(1)} n$ unless P = NP. In contrast, we show that SDP for approximation factor more than $n/\log n$ is not NP-hard unless there is an unlikely containment of complexity classes.
- For several permutation metrics, we show that the minimum weight problem is polynomial-time reducible to the subgroup distance problem for *solvable* permutation groups.

1 Introduction

We investigate the computational complexity of two natural problems for metrics on permutation groups given by generating sets. Given a permutation group $G = \langle A \rangle \leq S_n$ by a generating set A of permutations, we are interested in the *minimum weight problem* (denoted MWP) and the *subgroup distance problem* (denoted SDP) for natural permutation metrics. These problems were studied in [BCW06, CW06] by Cameron et al and are shown to be NP-hard for several natural permutation metrics.

These problems are analogous to the shortest vector problem and the closest vector problem for integer lattices, and to the minimum Hamming weight problem and nearest codeword problem for linear codes. The corresponding problems for lattices and codes are also NP-hard, and their approximability is a subject of current intensive study (see e.g. [MG02]). Our primary motivation stems from the fact that lattices and codes are abelian groups, and it is interesting to ask if the upper and lower bound techniques

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 136–147, 2008.

and results for approximability can be extended to arbitrary (nonabelian) permutation groups. Several permutation metrics are in the literature and have been studied from a statistical perspective. Deza [DH98] examines permutation metrics from a coding theory perspective by considering subgroups of S_n as codes.

Our main results: We give $2^{O(n)}$ time algorithms for the minimum weight problem for both the Hamming and the l_∞ permutation metrics. Notice that a naive brute-force search algorithm can take $n!$ steps since $G \leq S_n$ can have up to $n!$ elements. In the case of Hamming metric, it turns out that we can design a deterministic $2^{O(n)}$ time algorithm which is *group theoretic* in nature. The algorithm is based on the classical Schrier-Sims algorithm for finding pointwise stabilizer subgroups of permutation groups (see e.g. [Lu93]). However, the problem for l_∞ metric does not appear amenable to a group-theoretic approach. Our $2^{O(n)}$ time randomized algorithm for the problem is more geometric. Interestingly, for this algorithm we are able to adapt ideas from the Ajtai-Kumar-Sivakumar algorithm for the shortest vector problem in lattices [AKS01].

A function $d : S_n \times S_n \mapsto \mathbb{R}$ is a *metric* on the permutation group S_n if for all $\pi, \tau, \psi \in S_n$ $d(\pi, \tau) = d(\tau, \pi) \geq 0$ and $d(\pi, \tau) = 0$ iff $\pi = \tau$. Furthermore, the triangle inequality holds: $d(\pi, \tau) \leq d(\pi, \psi) + d(\psi, \tau)$.

Let $e \in S_n$ denote the identity permutation. For $\tau \in S_n$, $d(e, \tau)$ is the *norm* of τ for metric d, and is denoted by $\|\tau\|$. A *right-invariant* metric d on S_n satisfies $d(\pi, \tau) = d(\pi\psi, \tau\psi)$ for all $\pi, \tau, \psi \in S_n$. A *left invariant* metric is similarly defined. For a detailed discussion regarding metrics on S_n we refer to [DH98]. We recall the definitions of permutation metrics studied in this paper.

Hamming distance: $d(\tau, \pi) = |\{i | \tau(i) \neq \pi(i)\}|$.
l_p **distance:** for $p \geq 1$, $d(\tau, \pi) = (\Sigma_{i=1}^{n} |\tau(i) - \pi(i)|^p)^{1/p}$.
l_∞ **distance:** $d(\tau, \pi) = max_{1 \leq i \leq n} |\tau(i) - \pi(i)|$.
Cayley Distance: $d(\tau, \pi) =$ minimum number of transpositions taking τ to π.

These metrics are right invariant. Furthermore, the Hamming and Cayley metrics are also left invariant.

For $S \subseteq S_n$ and $\tau \in S_n$ let $d(\tau, S) = min_{\psi \in S} d(\tau, \psi)$. For $\tau \in S_n, r \in \mathbb{R}^+$ let $B_n(\tau, r, d) = \{\pi \in S_n | d(\pi, \tau) \leq r\}$ be the ball of radius r centred at τ for a metric d. Analogous to the geometric setting, we define the volume Vol(S) of a subset $S \subseteq S_n$ is its cardinality $|S|$. For right invariant metric d we have, for all $\tau \in S_n, r \geq 0$, Vol$(B_n(e, r, d)) = Vol(B_n(\tau, r, d))$. Next we define Subgroup Distance Problem and Minimum Weight Problem with respect to a metric d.

Definition 1. [CW06, BCW06]
Subgroup Distance Problem (SDP): *Input instances are* (G, τ, k)*, where* $G \leq S_n$ *is given by a generating set,* $\tau \in S_n$*, and* $k > 0$*. Is* $d(\tau, G) \leq k$*?*
Minimum Weight Problem (MWP): *Input instances are* (G, k)*,* $G \leq S_n$ *given by a generating set and* $k > 0$*. Is there a* $\tau \in G \setminus \{e\}$ *with* $\|\tau\| \leq k$*?*

We are also interested in approximate solutions to MWP and SDP. For MWP, given $\gamma > 1$ the approximation problem is to find a $\pi \in G$, $\pi \neq e$ such that $\|\pi\|$ is bounded by γ times the optimal value. The approximation version of SDP is likewise defined. It is useful to define promise decision versions of SDP and MWP that capture this notion of approximation:

For any permutation metric d, the promise problem GapSDP$_\gamma$ where γ is a function of n, is defined as follows: inputs are the SDP inputs (G, τ, k). An instance (G, τ, k) is a YES instance if there exist $\psi \in G$ such that $d(\psi, \tau) \leq k$. And (G, τ, k) is a NO instance if for all $\psi \in G$, $d(\psi, \tau) > \gamma k$. The problem GapMWP$_\gamma$ is similarly defined. An algorithm *solves* the promise problem if it decides correctly on the YES and NO instances.

2 A $2^{O(n)}$ Algorithm for MWP over l_∞ Metric

We consider the search version of MWP: given $G \leq S_n$, the goal is to find a permutation $\tau \in G \setminus \{e\}$ with minimum norm with respect to a metric d. We refer to such a $\tau \in G$ as a *shortest* permutation in G w.r.t. the metric d. First we consider the l_∞ metric and give a $2^{O(n)}$ time randomized algorithm for finding a shortest permutation for $G \leq S_n$ given by generating set. The algorithm uses the framework developed in [AKS01] for the shortest vector problem for integer lattices. Regev's notes [Re] contains a nice exposition.

The basic idea is to first pick N elements of G independently and uniformly at random, where N is $2^{c \cdot n}$ (where the constant c will be appropriately chosen). Each of these elements is multiplied by a random permutation of relatively smaller norm to get a new set of N elements. On this set of permutations a sieving procedure is applied in several rounds. The crucial property of the sieving is that after each stage the remaining permutations have the property that the maximum norm is halved and in the process at most $2^{c' \cdot n}$ elements are sieved out for a small constant c'.

Thus, repeated sieving reduces the maximum norm until it becomes a constant multiple of norm of shortest permutation of G. Then we can argue that for some π_1, π_2 from the final set of permutations, $\pi_1 \pi_2^{-1}$ will be a shortest permutation with high probability.

First we prove certain volume bound for l_∞ metric ball, which is crucially used in the algorithm, next we give a procedure to sample permutations from an l_∞ metric ball uniformly.

Lemma 1. *For $1 \leq r \leq n-1$ we have, $r^n/e^{2n} \leq \text{Vol}(B_n(e, r, l_\infty)) \leq (2r+1)^n$. Consequently, for any constant $\alpha < 1$, $\text{Vol}(B_n(e, r, l_\infty))/\text{Vol}(B_n(e, \alpha r, l_\infty)) \leq 2^{c_1 \cdot n}$, where $c_1 = \log_2(3e^2/\alpha)$.*

Proof. Let $\tau \in B_n(e, r, l_\infty)$. So, $|\tau(i) - i| \leq r$ for all i. Thus, for each i there are at most $2r+1$ choices for $\tau(i)$. This implies $\text{Vol}(B_n(e, r, l_\infty)) \leq (2r+1)^n$. Although better bounds can be shown, this simple bound suffices for the lemma. Now we show the claimed lower bound. Let $n = kr + t$, $0 \leq t \leq r-1$. For $jr + 1 \leq i \leq (j+1)r$, $0 \leq j \leq k-1$, $\tau(i)$ can take any value in $\{jr+1, jr+2, \ldots, (j+1)r\}$. Hence we have $\text{Vol}(B_n(e, r, l_\infty)) \geq r!^k t! \geq (r^r/e^r)^k t! \geq r^{n-t} t^t / e^n$. Using some calculus it is easily seen that the function $y = r^{n-t} t^t$ is minimum at $t = r/e$. Hence $r^{n-t} t^t / e^n \geq r^n / e^{n+r/e} \geq r^n / e^{2n}$. This proves the first part of lemma. The second part is immediate. ∎

We now explain an almost uniform random sampling procedure from $B_n(e, r, l_\infty)$. First, we randomly generate a function $\tau \in [n]^{[n]}$ by successively assigning values to

$\tau(i)$ for $i \in [n]$ as follows. For each $i \in [n]$ we have the list $L_i = \{j | 1 \leq j \leq n, i-r \leq j \leq i+r\}$ of candidate values for $\tau(i)$. Thus we have at most $(2r+1)^n$ functions from which we uniformly sample τ. Of course, τ defined this way need not be a permutation, but if it is a permutation then clearly $\tau \in B_n(e, r, l_\infty)$. Our sampling procedure outputs τ if it is a permutation and outputs "fail" otherwise. By Lemma 1 the probability that τ is a permutation is Prob$[\tau \in B_n(e, r, l_\infty)] \geq \frac{r^n}{e^{2n}(2r+1)^n} > \frac{1}{24^n} > 2^{-5n}$. Thus, if we repeat above procedure sufficiently many times (say 2^{10n} times) then the sampling procedure will fail with negligible probability, and when it succeeds it uniformly samples from $B_n(e, r, l_\infty)$. In summary we have the following lemma.

Lemma 2. *There exists a randomized procedure which runs in time $2^{O(n)}$ and produces an almost uniform random sample from $B_n(e, r, l_\infty)$.*

Now we describe the sieving procedure used in the algorithm. Hereafter we denote $B_n(\psi, r, l_\infty)$ by $B_n(\psi, r)$ for simplicity.

Lemma 3. *[Sieving Procedure] Let $r > 0$ and $\{\tau_1, \tau_2, \tau_3, \ldots, \tau_N\} \subseteq B_n(e, r)$ be a subset of permutations. Then in $N^{O(1)}$ time we can find $S \subset [N]$ of size at most $2^{c_1 n}$ for a constant c_1 such that for each $i \in [N]$ there is a $j \in S$ with $l_\infty(\tau_i, \tau_j) \leq r/2$.*

Proof. We construct S using a greedy algorithm. Start with $S = \emptyset$ and run the following step for all elements $\tau_i, 1 \leq i \leq N$. At the i^{th} step we consider τ_i. If $l_\infty(\tau_i, \tau_j) > r/2$ for all $j \in S$ include i in set S and increment i. After completion, for all $i \in [N]$ there is a $j \in S$ such that $l_\infty(\tau_i, \tau_j) \leq r/2$. To argue that $|S| < 2^{c_1 n}$ for constant c_1 we use the volume bound of Lemma 1. The construction of S implies for distinct indices $j, k \in S$ that $l_\infty(\tau_j, \tau_k) > r/2$. Hence the metric balls $B_n(\tau_j, r/4)$ for $j \in S$ are all pairwise disjoint. The right invariance of l_∞ metric implies Vol$(B_n(\tau_j, r/4)) = Vol(B_n(e, r/4))$. As $\tau_j \in B_n(e, r)$, by triangle inequality we have $B_n(\tau_j, r/4) \subseteq B_n(e, r + r/4)$ for $j \in S$. Hence $|S| < \frac{Vol(B_n(e, 5r/4))}{Vol(B_n(e, r/4))} \leq 2^{c_1 n}$ by Lemma 1, which also gives the constant c_1. This completes the proof of the lemma. ∎

Now we describe our algorithm to find a shortest permutation in G using Lemma 3. Let t denote the norm of a shortest permutation in G. The following claim gives an easy $2^{O(n)}$ time algorithm when $t \geq n/10$.

Lemma 4. *If the norm t of a shortest permutation in G is greater than $n/10$ then in time $2^{O(n)}$ we can find a shortest permutation in G.*

Proof. Consider $B_n(\tau, t/2)$ for $\tau \in G$. By triangle inequality, all $B_n(\tau, t/2)$ are disjoint. Also, by Lemma 1 we have Vol$(B_n(\tau, t/2)) \geq (t/2)^n \cdot e^{-2n} \geq n^n \beta^{-n}$ for some constant $\beta > 1$. Since $|G| \leq |S_n|/\text{Vol}(B_n(\tau, t/2))$, it follows that $|G| \leq \beta^n$. As we can do a brute-force enumeration of G in time polynomial in $|G|$, we can find a shortest permutation in $2^{O(n)}$ time. ∎

Now we consider the case when $t < n/10$. We run the algorithm below for $1 \leq t < n/10$ (the possible values of t) and output a shortest permutation in G produced by the algorithm.

1. Let $N = 2^{cn}$. For $1 \leq i \leq N$, pick ρ_i independently and uniformly at random from G, and pick τ_i almost uniformly at random from $B_n(e, 2t)$.
2. Let $\psi_i = \tau_i \rho_i$, $1 \leq i \leq N$. Let $Z = \{(\psi_1, \tau_1), (\psi_2, \tau_2), \ldots, (\psi_N, \tau_N)\}$, and let $R = max_i \|\psi_i\|$.
3. Set $T = [N]$.
4. While $R > 6 * t$ do the following steps:
   (a) Apply the "sieving procedure" of Lemma 3 to $\{\psi_i \mid i \in T\}$. Let set $S \subseteq T$ be the output of sieving procedure.
   (b) for all $i \in S$ remove tuple (ψ_i, τ_i) from Z.
   (c) for all $i \notin S$ replace tuple $(\psi_i, \tau_i) \in Z$ by $(\psi_i \psi_j^{-1} \tau_j, \tau_i)$, where $j \in S$ and $d(\psi_j, \psi_i) \leq R/2$.
   (d) set $R = R/2 + 2t$.
   (e) $T := T \setminus S$.
5. For all $(\varphi_i, \tau_i), (\varphi_j, \tau_j) \in Z$, let $\varphi_{i,j} = (\tau_j^{-1} \varphi_j)(\tau_i^{-1} \varphi_i)^{-1}$ (which is in G). Output a $\varphi_{i,j}$ with smallest nonzero norm.

In Step 1 of the algorithm, an almost uniform random sampling procedure from l_∞ metric ball is given by Lemma 2. For $G \leq S_n$, uniform sampling from G can be done in polynomial time by using a strong generating set for G (see e.g. [Lu93]). A random element is obtained by picking a coset representative at each level from the pointwise stabilizer tower of subgroups and multiplying them out. Thus Step 1 of the algorithm takes $2^{O(n)}$ time. Clearly, the while loop takes $2^{O(n)}$ time.

In order to prove the correctness, we examine the invariant maintained during each iteration of the while loop.

Proposition 1. *Before each iteration of the while loop, the following invariant is maintained. For all $i \in T$ we have $(\varphi_i, \tau_i) \in Z$, $\tau_i^{-1} \varphi_i \in G$ and $\|\varphi_i\| \leq R$.*

Proof. Clearly, the invariant holds before the first iteration. Inductively, suppose that at the beginning of an arbitrary iteration the set Z is of the form $Z = \{(\varphi_i, \tau_i) \mid i \in T\}$ such that $\tau_i^{-1} \varphi_i \in G$ and $\|\varphi_i\| \leq R$. During this iteration, in Z we replace (φ_i, τ_i) by $(\varphi_i \varphi_j^{-1} \tau_j, \tau_i)$, where $j \in S$ and $l_\infty(\varphi_i, \varphi_j) \leq R/2$. By right invariance of the l_∞ metric, we have $l_\infty(\varphi_i, \varphi_j) = \|\varphi_i \varphi_j^{-1}\| \leq R/2$. Triangle inequality implies $\|\varphi_i \varphi_j^{-1} \tau_j\| \leq \|\tau_j^{-1}\| + \|\varphi_i \varphi_j^{-1}\| = \|\tau_j\| + \|\varphi_i \varphi_j^{-1}\| \leq 2t + R/2$ which equals the value of R set in Step 4(d). Hence, $\|\varphi_i\| \leq R$ at the beginning of next iteration. Clearly, $\tau_i^{-1} \varphi_i \varphi_j^{-1} \tau_j$ is in G since $\tau_i^{-1} \varphi_i$ and $\tau_j^{-1} \varphi_j$ are in G. ∎

By Proposition 1 when the algorithm stops (after Step 5) we have $\tau_i^{-1} \varphi_i \in G$ and $\|\tau_i^{-1} \varphi_i\| \leq 8t$ for all $(\varphi_i, \tau_i) \in Z$. We want to argue that one of th $\varphi_{i,j}$ is equal to a shortest permutation in G with high probability. In Step 1 we pick τ_i almost uniformly at random from $B_n(e, 2t)$. Similar to the AKS algorithm's analysis, as explained in Regev's notes [Re], we define a new randomized procedure which also uniformly samples from $B_n(e, 2t)$ and has some properties which enable us to conveniently argue the correctness of the algorithm. In the lattice setting, the euclidean metric makes it easier to define a modified sampling from $B_n(e, 2t)$. However, for the l_∞ metric over S_n, the modified sampling from $B_n(e, 2t)$ is more involved. We now explain this modified sampling.

Let $\tau \in G$ be an element with shortest nonzero norm t. We introduce some notation. Let $C_\tau = B_n(e,2t) \cap B_n(\tau,2t)$, $C_{\tau^{-1}} = B_n(e,2t) \cap B_n(\tau^{-1},2t)$ and $C = C_\tau \cap C_{\tau^{-1}}$. The following claim is obvious.

Lemma 5. *Consider a map $\phi_1 : C_\tau \longrightarrow C_{\tau^{-1}}$ defined as $\phi_1(\sigma) = \sigma\tau^{-1}$. Then ϕ_1 is a bijection from C_τ onto $C_{\tau^{-1}}$.*

Let $\phi_1' : C_{\tau^{-1}} \longrightarrow C_\tau$ denote the inverse of ϕ_1.

We now define a randomized procedure Sample which on input a random permutation $\sigma \in B_n(e,2t)$ returns a new random permutation $\text{Sample}(\sigma) \in B_n(e,2t)$.

(i) If $\sigma \notin C_\tau \cup C_{\tau^{-1}}$ then $\text{Sample}(\sigma) = \sigma$ with probability 1.
(ii) If $\sigma \in C_\tau \setminus C$ then
  (a) if $\phi_1(\sigma) \in C$ then randomly set $\text{Sample}(\sigma)$ to either σ with probability $3/4$ or to $\phi_1(\sigma)$ with probability $1/4$.
  (b) if $\phi_1(\sigma) \notin C$ then randomly set $\text{Sample}(\sigma)$ to σ or $\phi_1(\sigma)$ with probability $1/2$ each.
(iii) If $\sigma \in C_{\tau^{-1}} \setminus C$ then define $\text{Sample}(\sigma)$ analogously as in Step (ii) above, using ϕ_1' instead of ϕ_1.
(iv) If $\sigma \in C$, then randomly set $\text{Sample}(\sigma)$ to either σ with probability $1/2$, or to $\phi_1(\sigma)$ with probability $1/4$, or to $\phi_1'(\sigma)$ with probability $1/4$.

The following lemma essentially states that the random variables $\text{Sample}(\sigma)$ and σ are identically distributed.

Lemma 6. *If σ is uniformly distributed in $B_n(e,2t)$ then* $\text{Sample}(\sigma)$ *is also uniformly distributed in $B_n(e,2t)$.*

Proof. Let $V = \text{Vol}(B_n(e,2t))$. We claim $\text{Sample}(\sigma)$ is uniformly distributed over $B_n(e,2t)$. For each $\pi \in B_n(e,2t)$ we have

$$\text{Prob}[\text{Sample}(\sigma) = \pi] \;=\; \sum_{\delta \in B_n(e,2t)} \text{Prob}[\sigma = \delta] \cdot \text{Prob}[\text{Sample}(\delta) = \pi].$$

We need to show that $\sum_{\delta \in B_n(e,2t)} \text{Prob}[\sigma = \delta] \cdot \text{Prob}[\text{Sample}(\delta) = \pi] = 1/V$. As σ is uniformly distributed, it is equivalent to showing $\sum_{\delta \in B_n(e,2t)} \text{Prob}[\text{Sample}(\delta) = \pi] = 1$. If $\pi \notin C_\tau \cup C_{\tau^{-1}}$ it is true directly from the definition of Sample. Consider $\pi \in C$, since maps ϕ_1 and ϕ_1' are bijective, there are unique $\sigma_1 \neq \sigma_2$ such that $\phi_1(\sigma_1) = \phi_1'(\sigma_2) = \pi$. By definition of Sample we have $\text{Prob}[\text{Sample}(\sigma_1) = \pi] = \text{Prob}[\text{Sample}(\sigma_2) = \pi] = \frac{1}{4}$ and $\text{Prob}[\text{Sample}(\pi) = \pi] = \frac{1}{2}$. Summing up we get $\Sigma_{\delta \in B_n(e,2t)} \text{Prob}[\text{Sample}(\delta) = \pi] = 1$ as desired. Now suppose $\pi \in C_\tau \setminus C$. If $\phi_1(\pi) = \psi \in C$ then clearly $\phi_1'(\psi) = \pi$. The definition of Sample implies that $\sum_{\delta \in B_n(e,2t)} \text{Prob}[\text{Sample}(\delta) = \pi] = \text{Prob}[\text{Sample}(\psi) = \pi] + \text{Prob}[\text{Sample}(\pi) = \pi] = \frac{1}{4} + \frac{3}{4} = 1$. If $\phi_1(\pi) = \psi \notin C$, we have $\sum_{\delta \in B_n(e,2t)} \text{Prob}[\phi(\delta) = \pi] = \frac{1}{2} + \frac{1}{2} = 1$. The case when $\pi \in C_{\tau^{-1}} \setminus C$ is similar. This proves the lemma. ∎

It follows from the definition of Sample that replacing τ_i by $\text{Sample}(\tau_i)$ does not affect the distribution of ψ_i in Step 2. In fact, $\text{Sample}(\tau_i)$ and τ_i are identically distributed by

Lemma 6. In Step 1 we pick each τ_i almost uniformly at random from $B_n(e, 2t)$. Now, in our analysis we replace this by Sample(τ_i). The crucial point of the argument is that it suffices to replace τ_i by Sample(τ_i) after Step 2, as the τ_i is only used to define ψ_i and it will not affect the distribution of ψ_i if we replace τ_i by Sample(τ_i). Note that τ_i is used during sieving step in the while loop only if i lies in the sieved set S. The remaining τ_i are replaced by Sample(τ_i) in Step 5. Clearly, this modification does not change the probability of computing a shortest permutation as the distributions in the two cases are the same. As already mentioned, note that Sample(τ_i) is introduced for analysis. We cannot implement the procedure Sample efficiently as we do not know τ.

In Step 1 of the algorithm we pick each τ_i almost uniformly at random from $B_n(e, 2t)$. The initial set is $\{\tau_i \mid i \in [N]\}$. The while loop iterates for at most $2 \log n$ steps and in each step we remove a set S of size at most $2^{c_1 n}$, where c_1 is given by Lemma 1. Thus, at the end of the while loop we still have $N - 2 \log n \cdot 2^{c_1 n}$ many τ_i in the remaining set T. Thus, as argued earlier for the purpose of analysis we can replace τ_i by Sample(τ_i) for all $i \in T$ and it still doesn't affect the working of the algorithm.

The triangle inequality implies $B_n(e, t) \subseteq C_\tau$. Thus $\text{Vol}(C_\tau) \geq Vol(B_n(e, t)) \geq t^n \cdot e^{-2n}$ by Lemma 1. Also, $\text{Vol}(B_n(e, 2t)) \leq (5t)^n$. Hence, $\frac{Vol(C_\tau \cup C_{\tau^{-1}})}{Vol(B_n(e,2t))} \geq 2^{-c_2 n}$, for some constant c_2 (which depends on c_1). Thus a random $\pi \in B_n(e, 2t)$ lies in $C_\tau \cup C_{\tau^{-1}}$ with probability at least $2^{-c_2 n}$.

Given a constant $c_3 > 0$, we can choose a suitably large $N = 2^{cn}$ for a constant c so that at least $2^{c_3 n}$ many τ_i for $i \in T$ at the end of the while loop will lie in $C_\tau \cup C_{\tau^{-1}}$. Thus, with probability $1 - 2^{-O(n)}$ we can guarantee that at least $2^{c_3 n}$ many τ_i for $i \in T$ are such that $\tau_i \in C_\tau \cup C_{\tau^{-1}}$ and $(\varphi_i, \tau_i) \in Z$ at the beginning of Step 5.

Furthermore, at the beginning of Step 5 each $(\varphi_i, \tau_i) \in Z$ satisfies $\|\tau_i^{-1}\varphi_i\| \leq 8t$ and $\tau_i^{-1}\varphi_i \in G$. Now we argue using the pigeon-hole principle that there is some $\pi \in G$ such that $\pi = \tau_i^{-1}\varphi_i$, $(\varphi_i, \tau_i) \in Z$ for at least 2^n indices $i \in T$.

Claim. $|G \cap B_n(e, 8t)| < 2^{c_4 n}$ for some constant c_4.

Proof. Note that $l_\infty(\pi_1, \pi_2) \geq t$ for distinct $\pi_1, \pi_2 \in G$. Thus, metric balls of radius $t/2$ around each element in $G \cap B_n(e, 8t)$ are all pairwise disjoint. By triangle inequality, all these $t/2$ radius metric balls are contained in $B_n(e, 8t + t/2)$. Hence $|G \cap B_n(e, 8t)| < Vol(B_n(e, 17t/2))/Vol(B_n(e, t/2)) < 2^{c_4 n}$, by Lemma 1. This proves the claim. ■

Let $c_3 = c_4 + 1$. Then with probability $1 - 2^{-O(n)}$ we have $\pi \in G$ such that $\pi = \tau_i^{-1}\varphi_i$, $(\varphi_i, \tau_i) \in Z$ for at least $2^{c_3 n}/2^{c_4 n} = 2^n$ indices $i \in T$. Call this set of indices T_0.

Recall that in our analysis we can replace τ_i by Sample(τ_i) for each $i \in T_0$. By the definition of Sample(τ_i), Prob[Sample$(\tau_i) = \tau_i \; \forall \; i \in T_0] \leq (3/4)^{2^n}$. Similarly, Prob[Sample$(\tau_i) \neq \tau_i \; \forall \; i \in T_0] \leq (1/2)^{2^n}$. Hence with probability $1 - 2^{-O(n)}$ there are indices $i, j \in T_0$ such that $(\varphi_i, \tau_i), (\varphi_j, \tau_j) \in Z$ and Sample$(\tau_i) = \tau_i$ Sample$(\tau_j) \neq \tau_j$. Clearly, Sample$(\tau_j) = \tau_j\tau$ or Sample$(\tau_j) = \tau_j\tau^{-1}$. Without loss of generality, assume Sample$(\tau_j) = \tau_j\tau$. Then, after Step 5 we have with high probability $\varphi_{i,j} = ((\tau_j\tau)^{-1}\varphi_j)(\tau_i^{-1}\varphi_i)^{-1} = \tau^{-1}\pi\pi^{-1} = \tau^{-1}$. In other words, with probability $1 - 2^{-O(n)}$ one of the $2^{O(n)}$ output permutations is a "shortest" permutation in G. We have shown the following theorem.

Theorem 1. *Given a permutation group $G \leq S_n$ as input, we can find a permutation in $G \setminus \{e\}$ with smallest possible norm with respect to l_∞ metric in $2^{O(n)}$ randomized time.*

3 Weight Problems for Hamming Metric

We first give an easy $2^{O(n)}$ time deterministic algorithm for Minimum Weight Problem in the case of Hamming metric. It turns out we can use a well-known algorithm from permutation groups. Suppose $G \leq S_n$ is given by a generator set. The problem is to find a shortest permutation in G for the Hamming metric. For every $S \subseteq [n]$ consider the pointwise stabilizer subgroup $G_S \leq G$ defined as $G_S = \{g \in G | \forall\, i \in S :\ g(i) = i\}$. Using the Schrier-Sims algorithm in polynomial time [Lu93] we can compute a generating set for G_S. Thus, in $2^{O(n)}$ time we can compute G_S for all $S \subseteq [n]$ and find the largest $t < n$ for which there is $S \subseteq [n]$ such that $|S| = t$ and G_S is a nontrivial subgroup. Clearly, any $\tau \neq e \in G_S$ is a shortest permutation with respect to Hamming metric.

Finally, we also consider the problem of finding an element in $G \leq S_n$ of maximum norm w.r.t. Hamming metric. We first consider the problem of deciding if $G \leq S_n$ has a fixpoint free permutation. In general it is known that this problem is NP-complete [CW06]. Using the Inclusion-Exclusion Principle we give a $2^{O(n)}$ time deterministic algorithm for the search version of the problem as follows. As before, let G_S be the subgroup of G that pointwise fixes $S \subseteq [n]$. Let $F \subseteq G$ denote the set of fixpoint free elements. Clearly, $F \cap G_S = \emptyset$ for each nonempty S. Also, $F \cup \bigcup_{S \neq \emptyset} G_S = G$. In $2^{O(n)}$ time we can compute generating sets for all G_S.

For the algorithm, inductively assume that we have already computed a coset H_{k-1} of $G_{[k-1]}$ in G, where for all $\tau \in H_{k-1}$ we have $\tau(i) = \alpha_i$, $\alpha_i \in [n]$ for $1 \leq i \leq k-1$ and H_{k-1} contains a fixpoint free permutation in G.

We now show how to compute a point $\alpha_k \in [n]$ which will fix the coset H_k of $G_{[k]}$ in $2^{O(n)}$ time such that for all $\tau \in H_k$, $\tau(i) = \alpha_i$ for $i = 1$ to k and H_k contains a fixpoint free element in G. By repeating this successively we can find a fixpoint free permutation. First, from the orbit of k we pick a candidate point α_k distinct from $\alpha_1, \cdots, \alpha_{k-1}$ and k. Let $H_k = \{\tau \in G | \tau(i) = \alpha_i, 1 \leq i \leq k\}$.

Let $A_i = H_k \cap G_{\{i\}}$ for $i = k+1$ to n. It is clear that intersection of any subcollection of these A_i 's is of the form $H_k \cap G_S$ for $S \subseteq [n]$. We can compute G_S in polynomial time for any $S \subseteq [n]$ using the Schreier-Sims algorithm [Lu93]. Furthermore, the coset intersection problem $H_k \cap G_S$ can also be solved in $2^{O(n)}$ time using the machinery of Babai and Luks [BL83, Lu93]. Thus, in time $2^{O(n)}$ we can compute $|\bigcap_{i \in S} A_i|$ for all subsets $S \subseteq [n-k]$. In $2^{O(n)}$ further steps, by using the Inclusion-Exclusion formula, we can compute $|A_{k+1} \cup A_{k+2} \cup \ldots \cup A_n| = m$. Clearly, H_k contains a fixpoint free element of G iff $m < |H_k|$. If $m = |H_k|$, we try the next candidate value for α_k from the orbit of k. This procedure clearly succeeds assuming H_{k-1} has a fixpoint free element of G. This gives $2^{O(n)}$ time algorithm to find a fixpoint free permutation. With minor changes to this algorithm, we can compute an element of maximum norm in G with respect to Hamming norm in $2^{O(n)}$ time. We summarize these observations in the following theorem.

Theorem 2. *Given a permutation group $G \leq S_n$ by a generating set, in $2^{O(n)}$ time we can find $\tau \in G \setminus \{e\}$ with smallest possible norm and $\psi \in G$ with largest possible norm with respect to Hamming metric.*

4 MWP Is Reducible to SDP for Solvable Permutation Groups

For integer lattices, SVP (shortest vector problem) is polynomial-time reducible to CVP (closest vector problem) [GMSS99]. A similar result for linear codes is also proved there. We show an analogous result for *solvable* permutation groups. In fact we give a polynomial-time Turing reduction from MWP to SDP, which works for the gap version of the problem for *any* right invariant metric d. We do not know if this reduction can be extended to nonsolvable permutation groups. Finally we make an observation about the hardness of approximation of SDP and MWP.

Let $G \leq S_n$ be input instance for MWP. The idea is to make different queries of the form (H, τ) to SDP, for suitable subgroups $H \leq G$ and $\tau \notin H$.

Let d be a right invariant metric on S_n. We want to find a shortest permutation $\tau \in G$ w.r.t. metric d. It is well-known in algorithmic permutation group theory (e.g. see [Lu93]) that for solvable permutation groups $G \leq S_n$ we can compute in deterministic polynomial time a composition series $G = G_k \rhd G_{k-1} \rhd \ldots \rhd G_1 \rhd G_0 = \{e\}$, $k \leq n$. In other words, G_{i-1} is a normal subgroup of G_i for each i. Furthermore, since G is solvable, each quotient group G_i/G_{i-1} has prime order, say p_i (where the p_i's need not be distinct). Notice that for any $\tau_i \in G_i \setminus G_{i-1}$, the coset $G_{i-1}\tau_i$ generates the cyclic quotient group G_i/G_{i-1}. It is easily seen that these elements τ_i form a generating set for G with the following standard property. We omit the proof due to lack of space.

Proposition 2. *For each $i, 1 \leq i \leq k$, every $\tau \in G_i \setminus G_{i-1}$ can be uniquely expressed as $\tau = \tau_1^{\alpha_1}\tau_2^{\alpha_2}\ldots\tau_i^{\alpha_i}, 0 \leq \alpha_j < p_j, 1 \leq j \leq i$ and $\alpha_i \neq 0$.*

Theorem 3. *For any right invariant metric d on S_n, and for solvable groups,* GapMWP$_\gamma$ *is polynomial time Turing reducible to* GapSDP$_\gamma$.

Proof. Let (G, m) be an input instance of GapMWP$_\gamma$. We compute $\tau_1, \ldots \tau_k$ for the group G as described above. Then we query the oracle of GapSDP$_\gamma$ for instances $(G_{i-1}, \tau_i^{-r}, m)$, for $1 \leq i \leq k, 1 \leq r < p_i$. The reduction outputs "YES" if at least one of the queries answers "YES" otherwise it outputs "NO". Clearly, the reduction makes at most $O(n^2)$ oracle queries and runs in polynomial time. We prove its correctness.

Suppose (G, m) is a "YES" instance of GapMWP$_\gamma$. We show that at least one of the queries $(G_{i-1}, \tau_i^{-r}, m), 1 \leq i \leq k, 1 \leq r < p_i$ will return "YES". Let $\tau \in G = G_k$ such that $\|\tau\| \leq m$. Let i be the smallest such that $\tau \notin G_{i-1}$, $\tau \in G_i$. From Proposition 2 it follows that τ can be uniquely expressed as $\prod_{j=1}^{i} \tau_j^{\alpha_j}$, where $0 \leq \alpha_j < p_j, 1 \leq j \leq i$ and $\alpha_i \neq 0$. As $\prod_{j=1}^{i-1} \tau_j^{\alpha_j} \in G_{i-1}$, we have $d(\tau_i^{-\alpha_i}, G_{i-1}) \leq d(\tau_i^{-\alpha_i}, \prod_{j=1}^{i-1} \tau_j^{\alpha_j})$. The right invariance of d implies $d(\tau_i^{-\alpha_i}, G_{i-1}) \leq d(e, \prod_{j=1}^{i} \tau_j^{\alpha_j}) = \|\tau\| \leq m$. Hence $(G_{i-1}, \tau_i^{-\alpha_i}, m)$ is a "YES" instance of GapSDP$_\gamma$.

Now suppose $(G_{i-1}, \tau_i^{-r}, m), 1 \leq i \leq k, 1 \leq r \leq p_i - 1$ is not a "NO" instance of GapSDP$_\gamma$. Then there is some $\tau \in G_{i-1}$ such that $d(\tau, \tau_i^{-r}) \leq \gamma m$, i.e. $\|\tau\tau_i^r\| \leq \gamma m$. As $\tau_i \in G_i \setminus G_{i-1}$, $\tau_i^t \notin G_{i-1}$ for $1 \leq t \leq p_i - 1$. Thus $\tau_i^r \notin G_{i-1}$ implying $\tau\tau_i^r \neq e$. Hence (G, m) is not a "NO" instance of GapMWP$_\gamma$. This completes the proof. ■

Cameron et al [BCW06, CW06] have shown that SDP and MWP are NP-hard for several permutation metrics. It follows from [ABSS97] that SDP for linear codes is NP-hard to approximate within a factor of $(\log n)^c$, where n is the block length of the input code and c is an arbitrary constant. Furthermore, Dumer et al [DMS99] have shown that constant-factor approximation is NP-hard for MWP restricted to binary linear codes. Given a binary linear code C of block length n, we can easily construct an abelian 2-group $G \leq S_{2n}$ isomorphic to C. An easy consequence of this construction and known hardness results for binary linear codes directly yields the following hardness results for GapSDP$_\gamma$ and GapMWP$_\gamma$ for different metrics.

Theorem 4. *For Hamming, Cayley, and the l_p metrics,* GapSDP$_\gamma$ *is NP-hard for $\gamma = O((\log n)^c)$ and* GapMWP$_\gamma$ *is NP-hard under randomized reduction for any constant γ.*

5 Limits of Hardness

Since GapSDP$_\gamma$ is NP-hard for $\gamma \leq (\log n)^c$, a natural question is to explore its complexity for larger gaps. For the GapCVP problem on lattices, Goldreich and Goldwasser [GG00] have shown a constant round IP protocol for $O(\sqrt{n/\log n})$ gap in the case of l_2 norm. Consequently, for this gap GapCVP is not NP-hard unless polynomial hierarchy collapses. We adapt similar ideas to the permutation group setting. For the Hamming and Cayley metric we give a constant round IP protocol for the complement problem of GapSDP$_\gamma$ for $\gamma \geq n/\log n$, such that the protocol rejects "YES" instances of GapSDP$_\gamma$ with probability at least $n^{-\log n}$, and always accepts the "NO" instances. For designing the IP protocols we require uniform random sampling procedures from metric balls for the Hamming and Cayley metrics.

We first consider the Cayley metric. Recall that the Cayley distance between τ and e is the least number of transpositions required to take τ to e. Let k be the number of cycles in τ. Each transposition multiplied to τ increases or decreases the number of cycles by 1. Since τ is transformed to e with the fewest transpositions if we always multiply by a transposition that increments the number of cycles, we have $d(e, \tau) = n - k$. Thus, a Cayley metric ball of radius r contains $\tau \in S_n$ such that τ has at least $n - r$ cycles. The number $c(n, k)$ of permutation in S_n with exactly k cycles is a Stirling number of the first kind and it satisfies the recurrence relation $c(n, k) = (n-1)c(n-1, k) + c(n-1, k-1)$. We can compute $c(m, l), 0 \leq m \leq n, 0 \leq l \leq k$ using the recurrence for $c(n, k)$.

Proposition 3. *Let $S \subseteq S_n$ be the set of permutations with k cycles. Let $N = |S| = c(n, k)$. Then there exists a polynomial (in n) time computable bijective function $f_{n,k} : [N] \mapsto S$.*

Proof. If $n = k = 1$, clearly such function exists, $f_{1,1}(1)$ is simply defined as identity element of S_1. We use induction on $n + k$. Assume that such functions exist for

$n+k \leq t$. Now consider n, k such that $n+k = t+1$. We define the function $f_{n,k}(i)$, for $1 \leq i \leq N$:

1. If $i > (n-1)c(n-1,k)$, let $\pi = f_{n-1,k-1}(i-(n-1)c(n-1,k))$ and τ be obtained by appending a 1-cycle (n) to π. Define $f_{n,k}(i) = \tau$.
2. If $i \leq (n-1)c(n-1,k)$ then find j such that $(j-1)c(n-1,k) < i \leq jc(n-1,k)$. Let $\pi = f_{n-1,k}(i-(j-1)c(n-1,k))$, write π as product of disjoint cycles. Let $\tau \in S_n$ be obtained by inserting n in the j^{th} position of the cyclic decomposition of π. Define $f_{n,k}(i) = \tau$.

Clearly, $f_{n,k}$ is polynomial time computable. We show $f_{n,k}$ is bijective by induction. Suppose $f_{n-1,k-1}$ and $f_{n-1,k}$ are bijective. Each $\tau \in S_n$ with k cycles can be uniquely obtained either by inserting element n in cyclic decomposition of a $\pi \in S_{n-1}$ with k cycles (which can be done in $n-1$ ways) or by attaching a 1-cycle with element n to some $\pi \in S_{n-1}$ with $k-1$ cycles. It follows that $f_{n,k}$ is bijective. ■

To uniformly sample $\tau \in S_n$ with k cycles, we pick $m \in \{1, 2, \ldots, c(n,k)\}$ uniformly at random and let $\tau = f_{n,k}(m)$.

Lemma 7. *There is a randomized procedure which runs in time $poly(n)$ and samples from $B_n(e,r,d)$ almost uniformly, where d denotes Cayley metric.*

Now consider the Hamming Metric. The Hamming ball of radius r contains all $\tau \in S_n$ such that $\tau(i) \neq i$ for at most r many points i. Hence, $\text{Vol}(B_n(e,r,d)) = \Sigma_{i=0}^{r} \binom{n}{i} D_i$, where D_i denotes the number of derangements on i points. We can easily enumerate all i-element subsets of $[n]$. The number D_i of derangements on i points satisfies the recurrence $D_i = (i-1)(D_{i-1} + D_{i-2})$. With similar ideas as used for sampling for Cayley metric balls we can do almost uniform random sampling from Hamming metric balls in polynomial time.

Lemma 8. *For $r > 0$, there exists a randomized procedure which runs in time $poly(n)$ and samples almost uniformly at random from the Hamming balls of radius r around e ($B_n(e,r,d)$).*

We now describe the simple 2-round IP protocol for the Hamming metric. Let (G, τ, r) be input instance of GapSDP$_\gamma$ for $\gamma \geq n/\log n$, and d is the Hamming metric.

1. **Verifier:** picks $\sigma \in \{0,1\}, \psi \in G, \beta \in B_n(e, \gamma r/2, d)$ almost uniformly at random. The verifier sends to the prover the permutation $\pi = \beta\psi$ if $\sigma = 0$, and $\pi = \beta\tau\psi$ if $\sigma = 1$.
2. **Prover:** The prover sends $b = 0$ if $d(\pi, G) < d(\pi, \tau G)$ and $b = 1$ otherwise.
3. **Verifier:** Accepts iff $b = \sigma$.

For the protocol we need polynomial time random sampling from a permutation group which is quite standard [Lu93]. We also need uniform sampling from Hamming metric balls which is given by Lemma 8. Due to lack of space, we omit the correctness proof of the protocol stated in the next lemma.

Lemma 9. *The verifier always accepts if* (G, τ, r) *is "NO" instance of* GapSDP_γ . *Furthermore, the verifier rejects with probability at least* $n^{-\log n}$ *if* (G, τ, r) *is a "YES" instance of* GapSDP_γ .

This shows correctness of the protocol for Hamming metric. For the Cayley metric too a similar IP protocol can be designed. As an immediate consequence we have the following.

Corollary 1. *For the Hamming and Cayley metrics,* GapSDP_γ *for* $\gamma \geq n/\log n$ *is not NP-hard unless coNP has constant round interactive protocols with constant error probability with the verifier allowed* $n^{O(\log n)}$ *running time.*

Recall that GapMWP_γ is Turing reducible to GapSDP_γ for solvable groups by Theorem 3 and the Turing reduction makes queries with the same gap. Hence, by the above corollary it follows that GapMWP_γ for solvable groups and for $\gamma > n/\log n$ is also unlikely to be NP-hard for Hamming and Cayley metrics.

Acknowledgement. We thank the referees for their valuable comments.

References

[ABSS97] Arora, S., Babai, L., Stern, J., Sweedyk, E.Z.: The hardness of approximate optima in lattices, codes, and system of linear equations. Journal of Computer and System Sciences 54(2), 317–331 (Preliminary version in FOCS 1993)

[AKS01] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector. In: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 266–275 (2001)

[BL83] Babai, L., Luks, E.M.: Canonical labeling of graphs. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, pp. 171–183 (1983)

[BCW06] Buchheim, C., Cameron, P.J., Wu, T.: On the Subgroup Distance Problem ECCC, TR06-146 (2006)

[CW06] Cameron, P.J., Wu, T.: The complexity of the Weight Problem for permutation groups. Electronic Notes in Discrete Mathematics (2006)

[DH98] Deza, M., Huang, T.: Metrics on Permutations, a Survey. J. Combin. Inform. System Sci. 23, 173–185 (1998)

[DMS99] Dumer, I., Micciancio, D., Sudan, M.: Hardness of approximating minimum distance of a linear code. In: 40th Annual Symposium on Foundations of Computer Science, pp. 475–484 (1999)

[GG00] Goldreich, O., Goldwasser, S.: On the limits of nonapproximability of lattice problems. Journal of Computer and System Sciences 60(3), 540–563 (2000)

[GMSS99] Goldreich, O., Micciancio, D., Safra, S., Seifert, J.P.: Approximating shortest lattice vector is not harder than approximating closest lattice vectors. Information Processing Letters 71(2), 55–61 (1999)

[Lu93] Luks, E.M.: Permutation groups and polynomial time computations. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 11, 139–175 (1993)

[MG02] Micciancio, D., Goldwasser, S.: Complexity of Lattice Problems. A Cryptographic Perspective. Kluwer Academic Publishers, Dordrecht (2002)

[Re] O. Regev.: Lecture Notes - Lattices in Computer Science, http://www.cs.tau.ac.il/ odedr/teaching/lattices_fall_2004/index.html

Periodic and Infinite Traces in Matrix Semigroups*

Paul Bell[1] and Igor Potapov[2]

[1] Department of Mathematics, Turku University
paubel@utu.fi
[2] Department of Computer Science, University of Liverpool
potapov@liverpool.ac.uk

Abstract. In this paper we provide several new results concerning word and matrix semigroup problems using counter automaton models. As a main result, we prove a new version of Post's correspondence problem to be undecidable and show its application to matrix semigroup problems, such as *Any Diagonal Matrix Problem* and *Recurrent Matrix Problem*. We also use infinite periodic traces in counter automaton models to show the undecidability of a new variation of the Infinite Post Correspondence Problem and *Vector Ambiguity Problem* for matrix semigroups.

1 Introduction

Over the last few decades, there has been considerable interest in algorithmic problems for matrix semigroups. Matrices are commonly used objects in mathematics, physics, engineering and other areas of science. Many questions about the dynamics of finite or infinite matrix products tightly connect problems with computation theory and theoretical computer science in general.

It is known that semigroups generated by matrices have a number of undecidable problems. Therefore many questions related to processes with iterative matrix multiplication are very hard to analyse and answer in general.

The majority of the investigated questions related to matrix semigroups were about direct reachability: membership problems (can a particular matrix be constructed by multiplication of matrices from a given set of generators?); vector reachability problems (can one particular vector be reached from another one by means of a set of linear transformations from a given set of generators?) and scalar reachability problems and their variants [2,3,8]. The only problem that may stand alone is the freeness problem for matrix semigroups: "Does every matrix in the semigroup have a unique factorization?" [6]. What might be interesting in terms of further understanding of the fundamental properties of matrix semigroups is to analyse the behavioural properties of possible traces and their relations.

The undecidability results for matrix semigroups were mainly shown by a reduction from the undecidability of Post's Correspondence Problem (PCP).

* This work was partially supported by Royal Society IJP 2007/R1 grant.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 148–161, 2008.

In this paper we want to highlight a number of "cross references" and connections between matrix semigroup problems and behavioral properties of counter machines. As a main result, we show that the *Vector Ambiguity Problem* and *Recurrent Matrix Problem* are undecidable for finitely generated matrix semigroups over integer and rational numbers. We also show that the problem of reaching any diagonal matrix is undecidable for 4×4 matrix semigroups over complex numbers. Moreover, using counter machine models, we provide a new undecidable variant of the infinite Post correspondence problem.

2 Preliminaries

In this section we introduce standard definitions and show some technical results about periodic traces in counter automata with their proofs to make the paper self contained.

Let us denote an empty word by ε. For two words $u = u_1 u_2 \cdots u_m$ and $v = v_1 v_2 \cdots v_n$ we denote their concatenation by $u \cdot v = u_1 u_2 \cdots u_m v_1 v_2 \cdots v_n$. For the inverse of a letter 'a' we sometimes write '$\overline{a}$'. Given a word $w = w_1 w_2 \cdots w_k$, we also define $\overline{w} = w^{-1} = \overline{w_k} \cdots \overline{w_2}\,\overline{w_1}$. We denote a word $\underbrace{aa \cdots a}_{k}$ by a^k.

Post's correspondence problem (PCP) is formulated as follows: Given a finite alphabet Σ and a finite (ordered) set of pairs of words in Σ^*: $\{(u_1, v_1), \ldots, (u_k, v_k)\}$. Does there exist a finite sequence of indices $(i_1, i_2, \ldots, i_m)$ with $1 \leq i_j \leq k$ for $1 \leq j \leq m$, such that $u_{i_1} u_{i_2} \cdots u_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}$. It can be shown that this problem is undecidable even with a binary alphabet Σ.

2.1 Two-Counter Minsky Machines

In this subsection we shall describe a well known computational model known as a Minsky machine. Informally speaking, a Minsky machine is a two-counter machine that can increment and decrement counters by one and test them for zero. It is known that Minsky machines are a universal model of computation [15]. Being of very simple structure, Minsky machines are very useful for proving undecidability results (see for example [11] or [12]).

It is convenient to represent a counter machine as a simple imperative program $\mathcal{M}$ consisting of a sequence of instructions labeled by natural numbers from 1 to some L. Any instruction is one of the following forms:

l: ADD 1 to S_k; GOTO l';
l: IF $S_k \neq 0$ THEN SUBTRACT 1 FROM S_k; GOTO l' ELSE GOTO l'';
l: STOP.

where $k \in \{1, 2\}$ and $l, l', l'' \in \{1, \ldots, L\}$.

The machine $\mathcal{M}$ starts executing with initial nonnegative integer values in counters S_1 and S_2 and the control at instruction 1. We assume the semantics of all the above instructions and of the entire program is clear. Without loss of generality, one can suppose that every machine contains exactly one instruction

of the form l: STOP which is the last one ($l = L$). It should be clear that the execution process (run) is deterministic and has no failure. Any such process is either finished by the execution of instruction L: STOP, or lasts forever.

As a consequence of the universality of such a computational model, the halting problem for Minsky machines is undecidable:

Proposition 1 ([15]). *It is undecidable whether a two-counter Minsky machine halts when both counters initially contain* 0.

In this paper, unless otherwise stated, we assume all machines start with 0 in both counters. Minsky machines can be simulated by PCP in a direct way. Given a two-counter machine M, our aim is to produce a set of pairs of words $P = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ such that there exists a finite sequence of indices $S = (i_1, i_2, \ldots, i_k)$ with each $1 \leq i_j \leq n$ where $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$ if and only if M halts when starting from the initial state with 0 in both counters. The sequence S we shall call a solution of the instance and it corresponds to a halting run of M and can be seen as an instance of PCP. Another option is to consider an infinite sequence S that corresponds to an infinite run of a counter machine.

Proposition 2. *A two-counter automata CA can be simulated by Post's correspondence problem (PCP), where PCP has a solution if and only if CA halts.*

Proof. We use the definitions of a two-counter machine from [10]. We require two operations, firstly "from state q, increment counter $\{1, 2\}$ and move to state s". Secondly, we require an operation to "test if counter $\{1, 2\}$ is zero, moving to state r if it is, or state t if it is positive". We shall use the symbol 'z' throughout to denote a zero counter.

We start with an initial pair of words $(u_1, v_1) = (\#, \#za^i q_0 a^j z\#)$ where i denotes the initial value of the counter C_1 and j denotes the initial value of the counter C_2. Let us deal with the first type of operation. To move from state q to s and increment C_1, we add the pair (q, as) to P. To move from state q to s and increment C_2, we add the pair (q, sa) to P. But the counters could be zero (denoted zqC_2 or C_1qz) so we also add pairs (zq, zas) to increment C_1 and (qz, saz) to increment C_2.

For the second operation, we require to move from q to r if C_1 is zero, so we add pair (zq, zr). To move from q to r if C_2 is zero, we add pair (qz, rz). To move from q to t and decrement C_1 if not zero, we add pair (aq, t) and to move from q to t and decrement C_2 if not zero, we add pair (qa, t). Finally, we add pairs (a, a), $(\#, \#)$, (z, z), $(aq_{\text{accept}}, q_{\text{accept}})$, $(q_{\text{accept}}a, q_{\text{accept}})$ and $(\#\triangleright, \triangleright)$ to P where $\triangleright$ is a new symbol.

We can enforce that the first pair used must be (u_1, v_1) by using a word mapping. Let $y = y_1 y_2 \cdots y_n \in \Gamma^*$ be any word and let '$*$' be a new letter not in Γ. Then define the three mappings:

$$\begin{aligned} \star y &= *y_1 * y_2 \cdots * y_n \\ y\star &= y_1 * y_2 * \cdots y_n* \\ \star y\star &= *y_1 * y_2 * \cdots * y_n* \end{aligned}$$

We apply one of the three above $\star$ mappings to each word pair. Let $(u_1, v_1) = (\star u_1, \star v_1 \star)$, $(u_j, v_j) = (\star u_j, v_j \star)$ for each $2 \leq j \leq (n-1)$ and $(u_n, v_n) = (\#*\triangleright, \triangleright)$ also. Clearly if a match occurs in P it must start with this new (u_1, v_1) since only the first two letters in these two words are equal. Examining the mapping allows us to conclude it must then proceed as before using the new pairs (u_i, v_i) with $2 \leq i \leq n$ and finally finish with the pair (u_n, v_n), see [17] for further details since this is similar to how a Turing machine is encoded.

If there exists some sequence $S = (i_1, i_2, \ldots, i_k)$ such that $u_1 u_{i_1} u_{i_2} \cdots u_{i_k} = v_1 v_{i_1} v_{i_2} \cdots v_{i_k}$ then it corresponds to a correct halting computation of a two-counter machine and it is thus undecidable whether such a sequence S exists.

2.2 Periodicity in Counter Machines

It was proven in [7] that given a counter machine M, we can construct a second counter machine, M', such that M' never halts and has a periodic configuration if and only if M halts. Since the halting problem for arbitrary two-counter machines is undecidable, this means that checking the periodicity is also undecidable. We shall now give a simple proof to the above result from [7] for completeness:

Proposition 3. *Let M' be a counter machine that has no halting configuration. The problem of deciding if M' has a periodic configuration is undecidable*[1].

Proof. Given a specific counter machine M. Let q_0 be the initial state of M and $H = \{q_{h_1}, q_{h_2}, \ldots, q_{h_t}\}$ be the set of halting states. Let $R = \{R_1, R_2, \ldots, R_k\}$ be the set of counters of M. See Figure 1 a.

We shall now show how to create a new machine M'. Initially, let M' have the same states Q as M and the same transition function δ. We add a new start state q_I and add the two rules to δ which move from q_I to state q_0 regardless of whether the first counter R_1 equals zero and leaves all counters as they are.

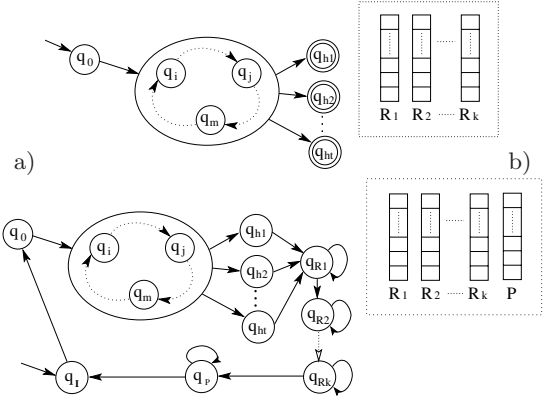


Fig. 1. a) Minsky machine with k counters b) Periodic Minsky machine

[1] The problem is undecidable even in the case of two-counter machines.

We define all states $q \in H$ to be non-halting states and add new states $q_{R_1}, q_{R_2}, \ldots, q_{R_k}$. These new states will be used to zero all counters. We add rules which move us from each $q \in H$ to q_{R_1} regardless of whether R_1 is non-zero and leave all counters at their current values. Then for each state q_{R_i}, $1 \leq i < k$ we add rules which decrease R_i if it is non-zero and remain in the current state. We add a rule to move to state $q_{R_{i+1}}$ if it does equal zero (thus the counter is decremented to zero before going to the next state). Finally, for state q_{R_k} we add a rule to decrease R_k if it is non-zero and stay in state q_{R_k}, or else move to the initial state q_I if it does equal zero (note that once we go to q_I, all counters are equal to 0 and we are back to the original configuration).

Thus, if M reached a halting state with some values in its counters R, then M' will instead decrement all counters to zero and restart the computation. Clearly the only way to get back to q_I is via some state in H of M, thus the only way M' is periodic is if M halts as required.

We may note that there may be some other configuration of M which is periodic, thus M' will contain an ultimately periodic configuration (though not periodic since it still will not go to q_I). We can avoid this situation if required by a simple construction. Add a new counter P such that every transition of machine M increments P and then does what it would do normally (we need to add new states and rules to do this). Then add a new state q_P such that q_{R_k} now goes to q_P instead of q_I. Then add rules to decrement q_P to zero as before and then move to state q_I. The only way to decrement counter P is via a halting state thus now the *only* periodic configurations contain q_I with all counters zero in their period. See Figure 1 b.

Proposition 4. *Let M' be a nondeterministic n-counter machine. The problem of deciding if M' has an infinite number of trajectories leading to a halting state q_{final} with zero in all counters is undecidable for any $n \geq 2$.*

Proof. The problem of determining if a counter machine M can reach a halting state with zero counters is undecidable. Without loss of generality we can also assume that the initial state of M will be visited only once. Let us construct a new nondeterministic counter machine M' based on a deterministic counter machine M as follows. First, we add two extra states q_{final} (which is the only halting state of M') and $q_{continue}$. Then add transitions from all halting states of M leading to both q_{final} and $q_{continue}$ which will only be executed if both counters are zero. Secondly, we create copies of all transitions from the inital state of M and add them to the automaton as outgoing transitions from $q_{continue}$.

As a result, we have that the initial state of M' (which is the same as in M) will be visited only once and state q_{final} with zero counters is reachable in M' if and only if machine M can reach a halting state with zero counters. On the other hand, if q_{final} with zero counters can be reached at least once, we can construct an infinite number of traces that will lead to q_{final} by returning from the halting state of M to $q_{continue}$ and repeating the same looping trace an unbounded number of times before going to state q_{final}.

3 Fixed Element PCP

Problem 1. Fixed Element PCP - Given an alphabet $\Gamma = \{a, b, a^{-1}, b^{-1}, \Delta, \Delta^{-1}, \star\}$ where $\Gamma \setminus \{\star\}$ forms a free group not containing '$\star$', and a finite set of pairs of words over Γ,

$$P = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\} \subset \Gamma^* \times \Gamma^*.$$

Does there exist a finite sequence of indices $s = (s_1, s_2, \ldots, s_k)$ such that $u_{s_1} u_{s_2} \cdots u_{s_k} = v_{s_1} v_{s_2} \cdots v_{s_k} = \star$?

This variant of PCP is interesting since it is similar to the standard PCP however instead of testing for a solution via equality checking for two arbitrary words, the solutions will have a specific form of a fixed letter $\star$.

Theorem 1. *The Fixed Element PCP is undecidable.*

Proof. The instance set of Fixed Element PCP will now be defined. Given an instance of PCP over $\Sigma = \{a, b\}$ where the first and last pairs are used exactly once[2]:

$$P' = \{(u_1, v_1), (u_2, v_2), \ldots, (u_m, v_m)\} \subset \Sigma^* \times \Sigma^*$$

we shall define two sets L, R such that $P = L \cup R$:

$$L = \{(\star u_1, \star\overline{\Delta} bab), (u_i, a^i b)\} \subset \Gamma^* \times \Gamma^* \quad ; 2 \leq i \leq m \tag{1}$$

$$R = \{(\overline{v_i}, \overline{\Delta}\overline{a}^i\,\overline{b}\Delta), (\overline{v_m}, \overline{b}\,\overline{a}^m\,\overline{b}\Delta)\} \subset \Gamma^* \times \Gamma^* \quad ; 1 \leq i \leq m-1 \tag{2}$$

Since we are looking for a product of pairs of words equal to $(\star, \star)$ and $\overline{\star} \notin \Gamma$, then the first pair $L_1 = (\star u_1, \star\overline{\Delta} bab)$ must occur exactly once. Let us denote any such product (if it exists) as:

$$X = (\star, \star) = X_1 X_2 \cdots X_k \in \langle L \cup R \rangle.$$

It can be seen that $X_1 = L_1$, otherwise if $X_j = L_1$ for some $j > 1$, then:

$$\langle (L \cup R) \setminus \{L_1\} \rangle \ni X_1 X_2 \cdots X_{j-1} = (\varepsilon, \varepsilon),$$

but this is impossible since clearly ε is not in the subsemigroup generated by $\{\overline{b}\overline{a}^m\overline{b}\Delta, a^i b, \overline{\Delta}\overline{a}^i\overline{b}\Delta : 1 \leq i \leq m-1\}$, therefore the second word cannot equal ε. This follows since the inverse of '$\overline{b}\Delta$' clearly cannot be found as a prefix of any word in the subsemigroup and thus an element equal to ε would have to begin using elements of the form $a^i b$. But this must eventually concatenate with $\overline{b}\overline{a}^m\overline{b}\Delta$ to reduce the word, and again we will have '$\overline{b}\Delta$' on the right which then cannot be reduced. Therefore we must have:

$$X = L_1 X_2 \cdots X_k = (\star, \star).$$

Let us consider the second words, in order to determine the sequence they must take to give '$\star$'. We have the set of elements:

$$A = \{\star\overline{\Delta} bab, \overline{b}\overline{a}^m\overline{b}\Delta, a^2 b, \ldots, a^m b, \overline{\Delta}\overline{a}\overline{b}\Delta, \ldots, \overline{\Delta}\overline{a}^{m-1}\overline{b}\Delta\}.$$

[2] This is standard in proofs of undecidability of PCP, see the construction in [17].

We know the first element is $(\star\overline{\Delta}bab)$ which is used only once. We now show that the only products equal to '$\star$' are of the form:

$$L_1 L_{i_1} L_{i_2} \cdots L_m R_m \cdots R_{i_2} R_{i_1} R_1, \tag{3}$$

for some $i_1, i_2, \ldots, i_l \in \{2, \ldots, m-1\}$. Since $(\star\overline{\Delta}bab)$ is the first element used, assume that the next element is from R, i.e., of the form $(\overline{\Delta}\overline{a}^i\overline{b}\Delta)$ for some $1 \leq i \leq m-1$ or $(\overline{b}\overline{a}^m\overline{b}\Delta)$. But this gives '$(\star\overline{\Delta}bab)(\overline{\Delta}\overline{a}^i\overline{b}\Delta)$' or '$(\star\overline{\Delta}b\overline{a}^{m-1}\overline{b}\Delta)$' and both cannot be reduced by further right multiplications since clearly from the set A, there is not any product of elements with a '$\overline{\Delta}b$' on the left hand side.

Thus, the only option is for the second element to be of the form $(a^i b)$ for $2 \leq i \leq m$. Let $j+1$ be the first index at which we do not have an element from L, thus the product $X_1 X_2 \cdots X_j$ is of the form: $(\star\overline{\Delta}bab)(a^{i_2}b)(a^{i_3}b)\cdots(a^{i_j}b)$ where $2 \leq i_2, i_3, \ldots, i_j \leq m$. To reduce this product, the next element must be $(\overline{b}\overline{a}^m\overline{b}\Delta)$ since this is the only element with a '$\overline{b}$' on the left. The product of $(a^{i_j}b)(\overline{b}\overline{a}^m\overline{b}\Delta)$ is $\overline{a}^{m-i_j}\overline{b}\Delta$. If $i_j < m$, then this will not reduce to '$\overline{b}\Delta$' and so the left '$\overline{b}$' will not cancel. Similarly to before, we cannot reduce this product any further since the right hand element is '$\overline{b}\Delta$'. Thus $i_j = m$.

The next element to the right cannot be $(\overline{b}\overline{a}^m\overline{b}\Delta)$ since the left hand letter does not cancel with Δ and it will have '$\overline{b}\Delta$' on the right hand side which cannot be canceled. Similarly, the next element cannot be of the form $(a^{k_1}b)$ since this would then give:

$$(\star\overline{\Delta}bab)(a^{i_2}b)(a^{i_3}b)\cdots(a^{i_j})\Delta a^{k_1}b,$$

and again this is only reduced with $(\overline{b}\overline{a}^m\overline{b}\Delta)$, but

$$(a^{i_j})\Delta a^{k_1}b(\overline{b}\overline{a}^m\overline{b}\Delta) = a^{i_j}\Delta\overline{a}^{m-k_1}\overline{b}\Delta,$$

and regardless of whether $k_1 = m$, the product ends with '$\overline{b}\Delta$' which cannot be reduced.

Thus, the next element must be of the form $(\overline{\Delta}\overline{a}^{k_2}\overline{b}\Delta)$ giving:

$$(\star\overline{\Delta}bab)(a^{i_2}b)(a^{i_3}b)\cdots(a^{i_j})\Delta(\overline{\Delta}\overline{a}^{k_2}\overline{b}\Delta)$$

and the product does not end with '$\overline{b}\Delta$' if and only if $k_2 = i_{j-1}$, in which case it ends with $\cdots a^{i_{j-1}}\Delta$. This continues inductively for each pair of elements from the center outwards and we see that we finally reach '$\star$' if and only if the product is of the form shown in Equation (3).

The first word corresponding to this is a correctly encoded PCP sequence which equals '$\star$' if and only if it corresponds to a correct solution word, completing the proof. Note that set P has 2 times the number of elements as the set P'. Since PCP was shown to be undecidable for 7 pairs of words in [13] (even with the first and last pairs used only once), Fixed Element PCP is undecidable for 14 pairs of words.

Let us consider a small example of the second word encoding for the sequence of words $1, i_1, i_2, m$. Following the above proof, we correctly obtain:

$$(\star\overline{\Delta}bab)(a^{i_1}b)(a^{i_2}b)(a^m b)(\overline{b}\overline{a}^m\overline{b}\Delta)(\overline{\Delta}\overline{a}^{i_2}\overline{b}\Delta)(\overline{\Delta}\overline{a}^{i_1}\overline{b}\Delta)(\overline{\Delta}\overline{a}\overline{b}\Delta) = \star.$$

4 Applications of FEPCP and Periodicity of Computations

In this section we show our main results via applications of the Fixed Element PCP and the construction of periodic traces in counter automaton models.

4.1 Any Diagonal Matrix Problem

Problem 2. **Any Diagonal Matrix Problem** - Given a finite set of matrices G generating a semigroup S. Does there exist *any* matrix $D \in S$ such that D is a diagonal matrix?

This problem was considered for integral matrices in [6], but we shall now show that it is undecidable for *rational complex* matrix semigroups by using the Fixed Element PCP. In our proof we shall exhibit a semigroup that has no diagonal matrices if the instance of PCP has no solution and an infinite number of diagonal matrices (i.e. powers of a specific, known diagonal matrix) if the PCP instance does have a solution.

Theorem 2. *Given a finitely generated matrix semigroup $S \subseteq \mathbb{C}(\mathbb{Q})^{4\times 4}$, it is algorithmically undecidable to determine whether there exists any matrix $D \in S$ such that D is a diagonal matrix.*

Proof. We shall utilize the Fixed Element PCP (FEPCP) and the matrix representation ζ of a free group of rational quaternions. Let us define a function $\zeta : \{a, b, \overline{a}, \overline{b}\} \rightarrow \mathbb{C}(\mathbb{Q})^{2\times 2}$ (where $\mathbb{C}(\mathbb{Q})$ denotes the field of complex numbers with rational coefficients):

$$\zeta(a) = \begin{pmatrix} \frac{3}{5} + \frac{4}{5}\mathbf{i} & 0 \\ 0 & \frac{3}{5} - \frac{4}{5}\mathbf{i} \end{pmatrix}, \zeta(b) = \begin{pmatrix} \frac{3}{5} & \frac{4}{5} \\ -\frac{4}{5} & \frac{3}{5} \end{pmatrix},$$
$$\zeta(\overline{a}) = \begin{pmatrix} \frac{3}{5} - \frac{4}{5}\mathbf{i} & 0 \\ 0 & \frac{3}{5} + \frac{4}{5}\mathbf{i} \end{pmatrix}, \zeta(\overline{b}) = \begin{pmatrix} \frac{3}{5} & -\frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{pmatrix}.$$

and ζ forms a free group as proved in [4]. Recall also that in the Fixed Element PCP we have an alphabet of 7 letters, $\Gamma = \{a, \overline{a}, b, \overline{b}, \Delta, \overline{\Delta}, \star\}$. We shall use a homomorphism, γ, to encode these letters using elements of ζ. Specifically, define $\gamma : \Gamma^* \rightarrow \mathbb{C}(\mathbb{Q})^{2\times 2}$ by:

$$\gamma(\star) = \zeta(a), \gamma(a) = \zeta(bab), \gamma(b) = \zeta(b^2a^2b^2), \gamma(\Delta) = \zeta(b^3a^3b^3)$$
$$\gamma(\overline{a}) = \zeta(\overline{b}\overline{a}\overline{b}), \gamma(\overline{b}) = \zeta(\overline{b}^2\overline{a}^2\overline{b}^2), \gamma(\overline{\Delta}) = \zeta(\overline{b}^3\overline{a}^3\overline{b}^3),$$

and then extending to a monoid homomorphism in the usual way. Now, given an instance of FEPCP:

$$P = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\} \subseteq \Gamma^* \times \Gamma^*,$$

we shall use the *mixed product* property of Kronecker products that for any four matrices $A, B, C, D \in \mathbb{C}^{j\times j}$:

$$(AB \otimes CD) = (A \otimes C)(B \otimes D).$$

We create a final set of 4-dimensional rational complex matrices:

$$T = \{\gamma(u_1) \otimes \gamma(v_1), \gamma(u_2) \otimes \gamma(v_2), \ldots, \gamma(u_n) \otimes \gamma(v_n)\} \subseteq \mathbb{C}(\mathbb{Q})^{4\times 4}.$$

From the definition of FEPCP, we know a solution $w = w_1 w_2 \cdots w_k$ gives the equation $u_{w_1} u_{w_2} \cdots u_{w_k} = v_{w_1} v_{w_2} \cdots v_{w_k} = \star$ for the special symbol $\star$. Using our above encoding, we see that $\gamma(\star) = \zeta(a)$ which is clearly diagonal. Thus, given a correct solution to FEPCP, there exists a matrix $D \in \langle T \rangle$ such that:

$$D = \gamma(u_{w_1} u_{w_2} \cdots u_{w_k}) \otimes \gamma(v_{w_1} v_{w_2} \cdots v_{w_k}) = \gamma(\star) \otimes \gamma(\star) = \zeta(a) \otimes \zeta(a),$$

which is diagonal (since the Kronecker product of two diagonal matrices is diagonal). It can be seen that any word which is not a solution will contain at least one matrix $\zeta(b)$ or $\zeta(\overline{b})$ and since these are not diagonal, the tensor product will not be diagonal either.

4.2 The Recurrent Matrix Problem

The next problem that we consider here is the problem of whether an element of a matrix semigroup has an infinite number of factorizations over elements of the generator. This question is trivially undecidable in the case of singular matrices since it can be reduced to the mortality problem (whether a zero matrix belongs to a semigroup). Here we show that this problem is also undecidable for invertible matrix semigroups.

Problem 3. **Recurrent Matrix Problem** - Given a matrix B and a semigroup S generated by a finite set of matrices G. Does B have an infinite number of factorizations over elements of G?

Theorem 3. *The recurrent matrix problem is undecidable for non-singular* 4×4 *integer matrix semigroups.*

Proof. From the construction in Proposition 4 we have that the problem of deciding if a nondeterministic two-counter automaton has an infinite number of trajectories leading to a final state s_{final} with zero counters is undecidable. We can encode and simulate this automaton by PCP, as shown in Proposition 2. We shall now also reduce it to the FEPCP which will be directly applicable to the recurrent matrix problem.

Thus, if the instance of FEPCP has a solution then a nondeterministic two-counter automaton has an infinite number of trajectories leading to a final state s_{final} with zero counters and FEPCP also has an infinite number of different sequences of indices that will lead to the solution $(\star, \star)$. We now show how to encode an instance of FEPCP into a matrix semigroup.

Let $\Sigma = \{a, b\}$ be a binary alphabet and $\overline{\Sigma} = \{\overline{a}, \overline{b}\}$ be the inverse elements of Σ, i.e., $\overline{a} = a^{-1}$ and $\overline{b} = b^{-1}$. We define the homomorphism $\varphi : (\Sigma \cup \overline{\Sigma})^* \to \mathbb{Z}^{2\times 2}$ by:

$$\varphi(a) = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \varphi(b) = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, \varphi(\overline{a}) = \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix}, \varphi(\overline{b}) = \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}$$

It is well known that φ forms a free group of 2×2 matrices from the free group of words generated by $\Sigma \cup \overline{\Sigma}$.

Let $\Gamma = \{a, \overline{a}, b, \overline{b}, \Delta, \overline{\Delta}, \star\}$ and define a new mapping γ, to encode Γ using elements of φ, where $\gamma : \Gamma^* \to \mathbb{Z}^{2\times 2}$ is given by:

$$\gamma(\star) = \varphi(a), \gamma(a) = \varphi(bab), \gamma(b) = \varphi(b^2a^2b^2), \gamma(\Delta) = \varphi(b^3a^3b^3)$$
$$\gamma(\overline{a}) = \varphi(\overline{b}\overline{a}\overline{b}), \gamma(\overline{b}) = \varphi(\overline{b}^2\overline{a}^2\overline{b}^2), \gamma(\overline{\Delta}) = \varphi(\overline{b}^3\overline{a}^3\overline{b}^3),$$

and then extending to a monoid homomorphism. Given an instance of FEPCP $P = \{(u_i, v_i)|1 \leq i \leq n\}$, for each $1 \leq i \leq n$ we define the following matrices:

$$A_i = \begin{pmatrix} \gamma(u_i) & 0 \\ 0 & \gamma(v_i) \end{pmatrix}.$$

If the matrix

$$B = \begin{pmatrix} \gamma(\star) & 0 \\ 0 & \gamma(\star) \end{pmatrix}$$

belongs to the semigroup S generated by $G = \{A_1, \ldots, A_n\} \subseteq \mathbb{Z}^{4\times 4}$ then FEPCP has a solution. According to the construction of the automaton, the FEPCP has a solution if the automaton halts and has an infinite number of trajectories leading to its final state. It also follows that the number of different paths leading to B is infinite. Since the existence of an infinite number of trajectories leading to the final state is an undecidable problem then the recurrent matrix problem is also undecidable.

4.3 The Vector Ambiguity Problem

Problem 4. **Vector Ambiguity Problem** - Given a semigroup S of $n \times n$ matrices and an initial n-dimensional vector u. Let V be a set of vectors such that $V = \{v : v = Mu; M \in S\}$. Do S and u generate a non-repetitive set of vectors? In other words the question is whether for every vector v of set V there is a unique matrix $M \in S$ such that $M \cdot u = v$?

Theorem 4. *The vector ambiguity problem is undecidable for matrix semigroups over integers in dimension* 4 *and over rationals in dimension* 3.

Proof. Let M be a two-counter machine that has no halting configuration. The problem of determining if M has a periodic configuration is undecidable. Let us use a construction proposed in Proposition 2, which simulates any two-counter machine by a set of pairs of words. Note that our method does not define a halting state of the machine and in this case we only predefine an initial configuration of a counter machine M.

Assume that the set of pairs of words used to simulate a counter machine is $P = \{(u_i, v_i)|1 \leq i \leq n\}$. Let us construct a set of pairs of 2×2 matrices using the homomorphism φ i.e. $\{(\varphi(\overline{u_1}), \varphi(v_1)), \ldots, (\varphi(\overline{u_n}), \varphi(v_n))\}$.

Instead of equation $u = v$ we consider a concatenation of two words $\overline{u} \cdot v$ which equals ε only in the case where $u = v$. We associate 2×2 matrix C

with a word w of the form $u \cdot \bar{v}$. Initially C is a matrix that corresponds to the initial configuration of the machine which is stored in the first pair (u_1, v_1), so $C = \varphi(\overline{u_1}) \cdot \varphi(v_1)$.

The extension of a word w by a new pair of words $(\overline{u_r}, v_r)$ (i.e., that gives us $w' = \overline{u_r} \cdot w \cdot v_r$) corresponds to the following matrix multiplication

$$C_{w'} = \varphi(\overline{u_r}) \cdot C_w \cdot \varphi(v_r) \tag{4}$$

Let us rewrite operation (4) in more detail.

$$\begin{pmatrix} c_{w'}^{11} & c_{w'}^{12} \\ c_{w'}^{21} & c_{w'}^{22} \end{pmatrix} = \begin{pmatrix} u^{11} & u^{12} \\ u^{21} & u^{22} \end{pmatrix} \cdot \begin{pmatrix} c_w^{11} & c_w^{12} \\ c_w^{21} & c_w^{22} \end{pmatrix} \cdot \begin{pmatrix} v^{11} & v^{12} \\ v^{21} & v^{22} \end{pmatrix} \tag{5}$$

In this case, pairwise multiplication will correspond to an update of the current state according to the operation of a two-counter machine M.

Let us consider the dynamics of changes for the matrix C. It is easy to see that in the case of an incorrectly applied command for a machine M, the pairwise concatenation (multiplication) will lead to increase of the length for a word w and will never end up in a repeated word after that. Therefore after an incorrect application of a command of M, a matrix C will never have the same value again. The correct application of pairwise concatenation of words or multiplication of matrices covers the set of correct configurations of a two-counter machine M. In the case of a periodic two-counter machine, the finiteness of the configuration space will lead to the finiteness of the set X of possible C matrices that can be generated during the correct application of rules for M, since every matrix $C \in X$ corresponds to a unique reachable configuration of M. Thus the set of matrices that can be generated by pairwise multiplication may contain repetitions if and only if the two-counter machine has periodic behavior.

In order to finish the proof of undecidability for the case of an integer matrix semigroup, we represent matrix C as a vector $x = ({c_w}^{11}, {c_w}^{12}, {c_w}^{21}, {c_w}^{22})^T$ increasing the dimension to 4 and rewriting pairwise multiplication as a 4-dimensional linear transformation of vector x.

$$\begin{pmatrix} c_{w'}^{11} \\ c_{w'}^{12} \\ c_{w'}^{21} \\ c_{w'}^{22} \end{pmatrix} = \begin{pmatrix} u^{11} & u^{12} \\ u^{21} & u^{22} \end{pmatrix} \otimes \begin{pmatrix} v^{11} & v^{12} \\ v^{21} & v^{22} \end{pmatrix}^T \cdot \begin{pmatrix} c_w^{11} \\ c_w^{12} \\ c_w^{21} \\ c_w^{22} \end{pmatrix} \tag{6}$$

The same method can be used to prove the undecidability of the vector ambiguity problem in dimension three for rational matrices by instead of using φ, using another homomorphism ψ based on a free group:

$$\psi(a) = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} \quad \psi(b) = \begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix} \quad \psi(\overline{a}) = \begin{pmatrix} 1 & -\frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix} \quad \psi(\overline{b}) = \begin{pmatrix} 1 & -1 \\ 0 & \frac{1}{2} \end{pmatrix}$$

Since we multiply only upper triangular matrices in (5) we have that c_w^{21} is equal to zero in the initial moment and after every next multiplication. When we rewrite (5) into a 4-dimensional linear transformation as is shown in (6), the

third row and the third column can be removed from the 4×4 matrix in (6). Finally, we rewrite (6) as a 3-dimensional linear transformation for the vector $({c_w}^{11}, {c_w}^{12}, {c_w}^{22})^T$.

Note that from the above result it also follows that it is undecidable whether there exists a periodic trace of configurations in a one state blind nondeterministic 4-counter machine with only counter updates in terms of linear transformations.

4.4 The Infinite Post Correspondence Problem

As another application of periodic counter machines we state a new variant of the infinite PCP. It was shown in [5] that the *infinite* Post correspondence problem is undecidable for 105 pairs of words. This result was later improved to 9 by V. Halava and T. Harju [9] by encoding semi-Thue systems and utilizing Claus's construction for PCP. The authors of [9] also show a related result that determining if a particular PCP instance has a solution that is non-ultimately periodic is undecidable.

Lemma 1. *[9] If the termination problem is undecidable for n-rule semi-Thue systems, then it is undecidable for instances of the PCP size $n + 3$ whether or not there exists an infinite solution that is not ultimately periodic.*

We recall that an infinite word is said to be ultimately periodic if it can be written in the form $w = uv^\infty$ where u, v are non-empty, finite words.

Using an encoding of two-counter machines into pairs of words and thus PCP instances and Proposition 3 (shown in [7]), we can derive a related result.

Lemma 2. *There exists a class of instances of Post's correspondence problem which have a guaranteed single infinite solution and no finite solution where it is undecidable whether the solution is ultimately periodic.*

Proof. We use the construction from Proposition 2 which allows us to simulate an arbitrary two-counter machine by a finite set of pairs of words as is done in Post's correspondence problem. We shall use the idea from Proposition 3 in which we start with an initial counter machine M and create a second counter machine M' such that M' *does not halt* and M' has a periodic configuration if and only if M halts on its input in the same way as was originally done in [7]. Since determining if an arbitrary counter machine M halts is undecidable, determining if M' has a periodic configuration is undecidable as explained in Proposition 3.

It is well known that a k-counter machine can be simulated by a two-counter machine. Therefore, from machine M', we create a third machine M'' which has only two counters. Using the construction from Proposition 2, we can simulate M'' via an instance of PCP which we denote by P. However, due to the construction of M', we know it *does not halt*, thus P has no solution.

The counter machine M' is deterministic and has a guaranteed infinite run. Therefore instance P has a single infinite word solution. By the conversion from

a two-counter machine to a PCP instance from Proposition 2, there is a single letter $\gamma_1 \in \Gamma$ which must be used first and then never again. Therefore, the infinite solution of PCP is of the form:

$$w' = \gamma_1 w; \quad \gamma_1 \in \Gamma, w \in (\Gamma \setminus \{\gamma_1\})^\infty.$$

Since determining if M' is periodic is undecidable, it is also undecidable whether w is a periodic word, or equivalently, whether w' is ultimately periodic, thus completing the proof.

Acknowledgements

We would like to thank the referees for their careful checking of this manuscript, especially the referee who noticed a gap in one of our earlier proofs.

References

1. Abdulla, P.A., Jonsson, B.: Undecidable Verification Problems for Programs with Unreliable Channels. Inf. Comput. 130(1), 71–90 (1996)
2. Babai, L., Beals, R., Cai, J., Ivanyos, G., Luks, E.M.: Multiplicative Equations Over Commuting Matrices. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 28–30 (1996)
3. Bell, P., Potapov, I.: Lowering Undecidability Bounds for Decision Questions in Matrices. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 375–385. Springer, Heidelberg (2006)
4. Bell, P., Potapov, I.: Reachability Problems in Quaternion Matrix and Rotation Semigroups. In: Kucera, L., Kucera, A. (eds.) MFCS 2007. LNCS, vol. 4708, Springer, Heidelberg (2007)
5. Blondel, V., Canterini, V.: Undecidable Problems for Probabilistic Automata of Fixed Dimensions. Theory Comput. Syst. 36(3), 231–245 (2003)
6. Blondel, V., Cassaigne, J., Karhumäki, J.: Problem 10.3. In: Blondel, V., Megretski, A. (eds.) Freeness of Multiplicative Matrix Semigroups, Unsolved Problems in Mathematical Systems and Control Theory, pp. 309–314
7. Blondel, V., Cassaigne, J., Nichitiu, C.: On the Presence of Periodic Configurations in Turing Machines and in Counter Machines. Theoretical Computer Science 289, 573–590 (2002)
8. Halava, V., Harju, T., Hirvensalo, M.: Undecidability Bounds for Integer Matrices using Claus Instances, TUCS Technical Report No. 766 (2006)
9. Halava, V., Harju, T.: Undecidability of Infinite Post Correspondence Problem for Instances of Size 9. Theoretical Informatics and Applications 40, 551–557 (2006)
10. Korec, I.: Small Universal Register Machines. Theoretical Computer Science 168, 267–301 (1996)
11. Kurganskyy, O., Potapov, I.: Computation in One-Dimensional Piecewise Maps and Planar Pseudo-Billiard Systems. Unconventional Computation, 169–175 (2005)
12. Lisitsa, A., Potapov, I.: In Time Alone: On the Computational Power of Querying the History. In: TIME 2006, pp. 42–49 (2006)
13. Matiyasevic, Y., Sénizergues, G.: Decision Problems for Semi-Thue Systems with a Few Rules. Theoretical Computer Science 330, 145–169 (2005)

14. Minsky, M.: Recursive Unsolvability of Post's Problem of "tag" and other Topics in Theory of Turing Machines. Annals of Mathematics 74, 437–455 (1961)
15. Minsky, M.: Computation: Finite and Infinite Machines. Prentice-Hall International, Englewood Cliffs (1967)
16. Potapov, I.: From Post Systems to the Reachability Problems for Matrix Semigroups and Multicounter Automata. In: Calude, C.S., Calude, E., Dinneen, M.J. (eds.) DLT 2004. LNCS, vol. 3340, pp. 345–356. Springer, Heidelberg (2004)
17. Sipser, M.: Introduction to the Theory of Computation. PWS Publishing (1997)

From Asynchronous to Synchronous Specifications for Distributed Program Synthesis

Julien Bernet and David Janin

LaBRI, Université de Bordeaux I
351, cours de la Libération
33 405 Talence cedex France
{bernet,janin}@labri.fr

Abstract. Distributed games [7] allow expression of various distributed program synthesis problems. In such games, a team of Process players compete against a unique Environment player, each Process playing on its own arena, without explicit communications with its teammates.

The standard definition of distributed games allows some degree of *asynchrony* : the Environment can play only on part of the arenas, therefore concealing to some Process players that the play is going on. While this is convenient for modeling distributed problems (especially those that themselves make use of asynchrony), these games are by no means easy to manipulate, and the existing constructions are much more tedious.

In this paper, we provide a *uniform* reduction of any finite state distributed game, synchronous or asynchronous, to a *synchronous* one with the same number of players. This reduction is shown to be correct in the sense that it preserves the existence of *finite state* distributed winning strategies for the team of Processes. It is uniform in the sense that it is designed for arbitrary input distributed game *without any prior knowledge* about its satisfiability. The size of the resulting synchronous game is also linear in the size of the original asynchronous one and, moreover, there is no blowup of the size of the winning strategies. Additional expected preservation properties (e.g. information flows) are studied. Surprisingly, it seems that no such reduction exists for arbitrary (infinite state) strategies.

Introduction

Asynchrony in a distributed environment can be defined in an abstract way as the ability of some processes or agents to perform some action in the distributed systems while other processes or agents not only perform no action but are even not aware that they are staying idle. In other words, in a model where every event in a process (e.g. a local action or an incoming or outgoing message) is triggered by the tick of a local clock, asynchrony occurs when there is no global clock on which local clocks are synchronized.

However, if the memory of a distributed system has finitely many possible states, it is commonly understood that synchronous and asynchronous behaviors are, to some extent, equivalent. In fact, the number of relevant asynchronous events between any two synchronous events can be bounded by the number of

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 162–173, 2008.

global states. More precisely, the (presumably asynchronous) distributed system can be modeled (at a more abstract level) by an equivalent synchronous one.

For instance, within a distributed system with an asynchronous message passing mechanism, if the number of messages that have been sent but not yet received is bounded, there cannot be an unbounded number of (relevant) asynchronous events between any two synchronizations.

Of course asynchrony makes a difference at the implementation level. For instance communication protocols do have to cope with asynchrony to be correct. However, at some more abstract level, provided the global memory of a system is finite, it seems that, in some sense, asynchronous and synchronous systems have the same computational power.

The main objective of this paper is to substantiate this intuition. More precisely, in the setting of distributed program synthesis problem, we aim at proving that (1) when the distributed environment is finite state and (2) when one considers only finite state programs, any synchronous or asynchronous distributed program synthesis problem is equivalent to a synchronous distributed program synthesis problem.

An immediate difficulty for this task is however to find the appropriate formalism.

In fact, there exists plenty of models of distributed systems and behaviors; consider for instance the diversity of models of communications between processes or agents: from communication via global shared memory mechanisms to many types of message passing systems. For distributed program synthesis specification, where one not only aims at modeling a single behavior but a full class of behaviors that may, or may not, fulfill the design objectives, there may be even more formalisms. See for instance Pnueli and Rosner's notion of distributed architecture [10,4] or Wonham et al.'s distributed control theory [5,6].

In this paper, we choose to study the relationship between synchronous and asynchronous distributed synthesis problems in a fairly abstract though general model: the model of distributed games [7].

These games, designed after Peterson and Reif's partial information games [9], rely on an explicit modeling of the local information - projection of the global state - available to every agent (or process) for guiding its own behaviors. As a matter of fact, no explicit communication is modeled, henceforth no commitment has to be done in favor of such or such communication mechanism. Moreover, most distributed synthesis problem can be encoded and solved within distributed games problem [2,7].

In a distributed game, a team of Process players compete against a unique Environment player, each Process playing on its own arena - with its own projection of the global state of the system - while Environment player has a complete knowledge of the global state.

Communications are implicitly modeled as follows. In a distributed game definition, one may restrict Environment player. From a given global state - partially known by every Process player - this may force the Environment player to reach some global states with more informative local projections than other global states.

Asynchrony, that induces an extra layer of partial information - absence of global clock - is modeled as follows: in a distributed game, the Environment may play only on part of the arenas, therefore concealing to some Process players that the play is going on elsewhere.

In this paper we prove that, as far as finite systems and programs are concerned, asynchrony does not yield extra expressive power.

More precisely, we provide a reduction of any asynchronous finite state distributed game to a *synchronous* one with the same number of players. This reduction is shown to be correct in the sense that it preserves the existence of *finite state* distributed winning strategies for the team of Processes. The size of the resulting synchronous game is also linear in the size of the original asynchronous one and, moreover, there is no blowup of the size of the winning strategies. Additional expected preservation properties (e.g. information flows) are also studied.

For strategies with unbounded memory, it is conjectured that there is no such a reduction. Still; one may ask whether such a result could be extended to more general classes of programs with infinite memory such as, for instance, pushdown strategies. This remains an open problem. However, even decidability questions are yet not settled for these more general strategies.

1 Notations

For any finite alphabet A, a *word* over A is a function $w : \mathbb{N} \to A$ whose domain is an initial segment of $\mathbb{N}$. The only word such with empty domain is called the empty word, and denoted by ϵ.

The following notations are used : A^* is the set of finite words (*i.e.* whose domain is finite) over A, A^ω is the set of infinite words over A, $A^\infty = A^* \cup A^\omega$ is the set of finite and infinite words over A, and $A^+ = A^* - \{\epsilon\}$ is the set of finite non-empty words over A.

For any finite word w, its length $|w|$ is the cardinal of its domain. For any words $u \in A^*$ and $v \in A^\omega$, the word $u \cdot v$ is the concatenation of words u and v. For any integer k, the word u^k is built by concatenating k copies of u. When $u \neq \epsilon$, the infinite word u^ω is the infinite concatenation of u with itself.

These operations are extended to the languages. Given $X \subseteq A^*$ and $Y \subseteq A^\infty$, we use the notations $X \cdot Y = \{u \cdot v \ : \ u \in X, v \in Y\}$, $X^0 = \{\epsilon\}$, $X^k = \{w \cdot u \ : \ w \in X^{k-1}, u \in X\}$ for any $k > 0$, $X^* = \bigcup_{k \geq 0} X^k$ and, when $\epsilon \notin X$, X^ω for the set of all words that can be defined as an infinite product of words of X. The following additional notations are also used : $X + Y = X \cup Y$, $X - Y = X \backslash Y$ and $X^? = \{\epsilon\} \cup X$.

Given n sets $X_1, \ldots, X_n$, consider their direct product $X = X_1 \times \cdots \times X_n$. For every set $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, consider the natural projection $\pi_I : X \to X_{i_1} \times \cdots \times X_{i_k}$ defined by $\pi_I((x_1, \ldots, x_k) = (x_{i_1}, \ldots, x_{i_k})$. For any $x \in X$, we also use the more convenient notation $x[I] = \pi_I(x)$.

Following the same idea, for any set $Y \subseteq X$, denote by $Y[I]$ the set $\pi_I(Y)$. If $R \subseteq X^m$ is a m-ary relation over X, we should also write use $R[I] = \{(x_1[I], \ldots, x_m[I]) \ : \ (x_1, \ldots, x_m) \in R\}$. When I in an interval of integers of

the form $[i,j]$ we use the notations $x[i,j], Y[i,j], R[i,j]$. Whenever I is reduced to a single integer i, these notations simplify to $x[i], Y[i], R[i]$.

Moreover, these notations are extended to words and languages : for any word $w = x_0 \cdot x_1 \cdots \in X^\infty$, then $w[I] = x_0[I] \cdot x_1[I] \cdots$.

2 Distributed Games

A n-process *distributed arena* is a game arena built from some product of n local standard game arena.

Definition. Given n arenas $G_i = \langle P_i, E_i, T_{P,i}, T_{E,i} \rangle$ for $i \in [1,n]$ with disjoint sets of Process position P_i, sets of Environment position E_i, sets of Process moves $T_{P,i} \subseteq P_i \times E_i$ and sets of Environment moves $T_{E,i} \subseteq E_i \times P_i$, an n-Process *distributed game arena* built from the local game arenas $\{G_i\}_{i\in[1,n]}$ is any game arena $G = \langle P, E, T_P, T_E \rangle$ such that:

- *Environment positions* : $E = \prod_{i\in[1,n]} E_i$,
- *Processes positions* : $P = \prod_{i\in[1,n]}(E_i \cup P_i) - \prod_{i\in[1,n]} E_i$,
- *Processes moves* : T_P is the set of all pairs $(p,e) \in (P \times E)$ such that, for $i \in [1,n]$:
  - **either** $p[i] \in P_i$ and $(p[i], e[i]) \in T_{P,i}$ (Process i is *active in* p),
  - **or** $p[i] \in E_i$ and $p[i] = e[i]$ (Process i is *inactive in* p),
- and *Environment moves* : T_E is **some subset** of the set of all pairs $(e,p) \in (E \times P)$ such that, for $i \in [1,n]$:
  - **either** $p[i] \in P_i$ and $(e[i], p[i]) \in T_{P,i}$ (Environment activates Process i),
  - **or** $p[i] \in E_i$ and $p[i] = e[i]$ (Environment keeps Process i inactive).

When the set T_E of Environment moves is maximal, we call such an arena the *free asynchronous product* of arenas $\{G_i\}_{i\in[1,n]}$ and it is written $G_1 \otimes G_2 \otimes \cdots \otimes G_n$.

Remark. The essential idea behind this definition is to get a definition of a multiplayer game in which a team of Processes compete against a unique Environment to achieve some infinitary goal. The following point is important : this definition allows the Environment to play only on a subset of the arenas, therefore hiding that the play is going on to the Processes on which arenas it does not play. This will be referred in the following as *asynchronous move*, and allows to encode neatly many distributed synthesis problems from the literature, such as [10,4] or [5].

In distributed games, asynchrony occurs when Environment player decides to keep one or more Process players inactive. A synchronous distributed arena can thus be defined as follows.

Definition. An n-process distributed arena $G = \langle P, E, T_P, T_E \rangle$ is a *synchronous distributed arena* when $T_E \subseteq E \times \prod_{i\in[1,n]} P[i]$.

Since a distributed arena is built upon n simple arenas, we need a definition to speak about its local components:

Definition. Given a distributed arena $G = \langle P, E, T_P, T_E \rangle$ as above, given a non empty set $I \subseteq [1, \ldots, n]$ we define the *canonical projection* $G[I]$ of G on I as the arena $G[I] = \langle P', E', T'_P, T'_E \rangle$ given by: $P' = P[I] - E[I]$ (possibly smaller than $P[I]$!), $E' = E[I]$, $T'_P = T_P[I] \cap (P' \times E')$, and $T'_E = T_E[I] \cap (E' \times P')$.

A distributed game arena is, at first sight, a particular case of standard discrete and turn base two player game arena. Standard notions of plays and strategies are still defined. However, in order to avoid confusion with what may happen in the local arena a distributed game is build upon, we shall speak now of a *global play* and *global and local strategy*. Partial information is then captured by means of the notion of *local view of play* and *distributed strategy*.

Definition. Given an n-process distributed arena G, a *global play* from an initial position $e \in E$ is defined as a path in G (seen as a bipartite graph) emanating from position e that is built alternatively by the Environment player and the Process team.

More precisely, from a current position $x \in P \cup E$, either $x \in E$ and it is Environment player turn to play by choosing some position $y \in P$ such that $(x, y) \in T_E$ or $x \in P$ and it is Process team turn to play by choosing some position $y \in E$ such that $(x, y) \in T_P$.

Accordingly, a *global strategy* for the Process team is a partial function $\sigma : (E.P)^+ \to E$ such that for every play of the form $w.p \in dom(\sigma)$ with $w \in E.(P.E)^*$ and $p \in P$ one has $(p, \sigma(w.p)) \in T_P$.

A play w is said *compatible* with strategy σ when, for every integer $n \geq 0$ such that $w[n] \in P$ one has $w[n+1] = \sigma(w[0, n])$ where $w[0, n]$ is the prefix of w of length $n + 1$.

Definition. A game $G = \langle P, E, T_E, T_P, e_0, \mathcal{W} \rangle$ is a game arena $\langle P, E, T_E, T_P \rangle$ equipped with an extra initial position $e_0 \in E$ and a distinguished set $\mathcal{W} \subseteq (E + P)^\omega$ called infinitary condition for the Process team.

We say that global strategy σ is a *winning strategy* for the Process team from position $e_0 \in E$ with condition $\mathcal{W}$ when every maximal plays starting in e_0 and compatible with strategy σ is either finite and ends in an environment position or is infinite and belongs to $\mathcal{W}$.

A *strategy with finite memory* for the Process team is given as a tuple $\mathcal{M} = \langle M, m_0, \mu : M \times (P \cup E) \to M, h : M \times P \to E \rangle$, where M is a finite set of *memory states*, m_0 is the *initial memory*, μ is the *update function*, and h is the *hint function*. The induced strategy $\sigma_{\mathcal{M}} : (E \cdot P)^+ \to E$ is then defined, for any play $w \cdot p \in (E \cdot P)^+$, by $\sigma_{\mathcal{M}} = h(\mu^*(m_0, w), p)$ (where μ^* is defined by $\mu^*(m, \epsilon) = m$, and $\mu^*(m, w \cdot x) = \mu(\mu^*(m, w), x)$ for every $m \in M$, $w \in (E \cup P)^*$, $x \in (E \cup P)$).

In distributed games, it is intended that, within the Process team, every process has only a partial view of a global play. Not only every process only sees its own projection of every global positions, but, when idle, a process is even not aware that the play is going on. This intention is formally defined as follows.

Definition. The *local view* Process i has of a global play in a distributed game G is given by the map $view_i : (E \cdot P)^* \cdot E^? \to (E_i \cdot P_i)^* \cdot E_i^?$ defined in the following way:

- $view_i(\epsilon) = \epsilon$
- $view_i(x) = x[i]$
- $view_i(w \cdot x \cdot y) = \begin{cases} view_i(w \cdot x) \text{ if } x[i] = y[i] \\ view_i(w \cdot x) \cdot y[i] \text{ otherwise.} \end{cases}$

A play $w \in (E \cdot P)^+$ is said to be active for Process i when w ends in a position $p \in P$ such that $p[i] \in P[i]$.

Remark. Observe that in a synchronous distributed arena, as expected, for every play $w \in (E.P)^*.E^?$ one has $view_i(w) = w[i]$, i.e. the local view of a global play is just the projection of this play.

Definition. A global strategy σ for the Process team is a *distributed strategy* when, for every $i \in [1, n]$, there is a process strategy $\sigma_i : (E[i].P[i])^+ \to E[i]$ in the local game $G[i]$, from now on called *local strategy* for Process i, such that, for any play of the form $w \cdot p \in (E \cdot P)^+$, given the set $I \subseteq \{1, \ldots, n\}$ of active processes in the global Processes position p, $\sigma(w \cdot p) = e$ if and only if

- $e[i] = \sigma_i(view_i(w) \cdot p[i])$ for $i \in I$
- $e[i] = p[i]$ for $i \in \{1, \ldots, n\} - I$

In this case, we write $\sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n$ for the distributed strategy σ.

Remark. Observe that when G is synchronous, the distributed strategy $\sigma = \sigma_1 \otimes \cdots \otimes \sigma_n$ can simply be defined for any global play $w \in (E \cdot P)^*$ by :

$$\sigma(w) = (\sigma_1(w[1]), \ldots, \sigma_n(w[n]))$$

Remark. Global strategies are not always distributed. In particular, there are distributed games where Process team has a winning global strategy, but no winning distributed strategy. For more details, the reader can refer to [7].

3 From Asynchronous Game to Synchronous Game

We prove here that every (asynchronous) distributed game is equivalent in some sense to a synchronous distributed game. More precisely:

Theorem 1. *There exists a mapping that maps every distributed game G to exists a distributed game $\widetilde{G}$ such that the Process team have a distributed winning strategy with finite memory in G if and only if they have one in $\widetilde{G}$. Moreover, game $\widetilde{G}$ has the same number of Process players as game G with only a linear increase of number of positions. Moreover, this mapping is defined uniformly on distributed games, be them winning for the process team or not.*

The remaining of this section is devoted to the proof of this result.

Let $G = \langle P, E, T, e_0, \mathcal{W}\rangle$ be a n-processes distributed game. First, we are going to describe the synchronous game $\widetilde{G}$, then we will show that it is equivalent to G in terms of distributed strategies with finite memory.

For any set E_i, define $\widehat{E_i}$ as an equipotent set, such that $E_i \cap \widehat{E_i} = \emptyset$; for any $e \in E_i$, denote by $\widehat{e_i}$ its image in $\widehat{E_i}$. Let $\widehat{E} = \prod_{i\in\{1,\dots,n\}} \widehat{E_i}$ and let $\widehat{P} = \prod_{i\in\{1,\dots,n\}} \widehat{P_i}$ (i.e. we restrict to relevant process positions in a synchronous game: process positions where every process is active).

Consider $\widetilde{G} = \langle \widetilde{P}, \widetilde{E}, \widetilde{T}, e_0, \widetilde{\mathcal{W}}\rangle$, whose positions are:

$$\widetilde{P_i} = P_i \cup \widehat{E_i} \quad ; \quad \widetilde{E_i} = E_i \quad (\text{for all } i \in \{1, \dots n\})$$

For any position $x \in P \cup E$, denote by $\widehat{x}$ the position of $\widetilde{P}$ obtained by replacing in e each component from E_i with their image in $\widehat{E_i}$, i.e. :

$$\widehat{x}[i] = \begin{cases} \widehat{x[i]} \text{ if } x[i] \in E_i \\ x[i] \text{ if } x[i] \in P_i \end{cases}$$

The function that maps any $x \in P \cup E$ to its image $\widehat{x}$ in $\widetilde{P}$ is trivially a bijection. The moves of $\widetilde{G}$ are defined as follows:

$$\begin{aligned} \widetilde{T_i^P} &= T_i^P \cup \{(\widehat{e}, e) : e \in E_i\} \\ \widetilde{T^E} &= \{(e, \widehat{p}) \in \widetilde{E} \times \widetilde{P} \mid (e, p) \in T^E\} \\ &\quad \cup \{(e, \widehat{e}) : e \in E\} \end{aligned}$$

The function $\mathsf{cancel}(\widetilde{E} \cdot \widetilde{P})^* \to (E \cdot P)^*$ erases any asynchronous move from a global play: $\mathsf{cancel}(\epsilon) = \epsilon$, $\mathsf{cancel}(w \cdot e \cdot p) = \mathsf{cancel}(w) \cdot e \cdot p$, and $\mathsf{cancel}(w \cdot e \cdot \widehat{e}) = \mathsf{cancel}(w)$ (where $p \in P$, $e \in E$).

This function is generalized to infinite words by: $\mathsf{cancel}(x_0 \cdot x_1 \cdot \cdots) = \lim_{i\to\infty} \mathsf{cancel}(x_0 \cdots x_i)$ (it is a converging sequence, in the sense of the prefix topology over words, as defined for instance in [8]).

The winning condition of $\widetilde{G}$ is then defined as follows:

$$\widetilde{\mathcal{W}} = \mathsf{cancel}^{-1}(\mathcal{W}) \cup (\widetilde{E} \cdot \widetilde{P})^* \cdot (E \cdot \widehat{E})^\omega$$

Remark. The underlying bipartite graph of G is embedded into the one from $\widetilde{G}$. The graph of the arena of $\widetilde{G}$ is actually nothing more than the subgraph induced by this embedding, where on each position of the environment a loop of size 2 has been added, corresponding to a totally asynchronous move. Moreover, the winning condition $\widetilde{\mathcal{W}}$ is not much more complicated than $\mathcal{W}$: amongst the usual infinitary winning conditions (reachability, safety, parity, Muller, etc. ...), only the safety condition in not preserved by this construction.

Lemma 1 (From asynchronous to synchronous). *For any distributed and winning strategy for the Process team in G, there is a winning distributed strategy for the Process team in $\widetilde{G}$.*

The idea is actually to copy this strategy on $\widetilde{G}$, ensuring in the process that the Process players do not take the asynchronous moves played onto their arena into account.

For any $i \in \{1, \ldots, n\}$, let us define a function $\mathsf{cancel}_i : (\widetilde{E_i} \cdot \widetilde{P_i})^* \to (E_i \cdot P_i)^*$ that erases the local asynchronous moves: $\mathsf{cancel}_i(\epsilon) = \epsilon$, $\mathsf{cancel}_i(w \cdot e \cdot p) = \mathsf{cancel}_i(w) \cdot e \cdot p$, and $\mathsf{cancel}_i(w \cdot e \cdot \widehat{e}) = \mathsf{cancel}_i(w)$ for any $e \in E_i$, $p \in P_i$, and $w \in (\widetilde{E_i}.\widetilde{P_i})^*$.

Consider a winning distributed strategy $\sigma = \sigma_1 \otimes \cdots \otimes \sigma_n$ over G. The distributed strategy $\widetilde{\sigma} = \widetilde{\sigma_1} \otimes \cdots \otimes \widetilde{\sigma_n}$ is defined for any $i \in \{1, \ldots, n\}$, for any local play $w \in \widetilde{E_i} \cdot (\widetilde{P_i} \cdot \widetilde{E_i})^*$, and for any positions $p \in P_i$, $e \in E_i$ by:

$$\widetilde{\sigma_i}(w \cdot p) = \begin{cases} \sigma_i(\mathsf{cancel}_i(w \cdot p)) \text{ if } \mathsf{cancel}_i(w \cdot p) \in Dom(\sigma_i) \\ \text{undetermined} \quad \text{otherwise.} \end{cases}$$
$$\widetilde{\sigma_i}(w \cdot \widehat{e}) = e$$

It is clear that for any $i \in \{1, \ldots, n\}$, the following diagram commutes:

$$\begin{array}{ccc} (\widetilde{E} \cdot \widetilde{P})^* & \xrightarrow{\mathsf{cancel}} & (E \cdot P)^* \\ {\scriptstyle \pi_i}\downarrow & & \downarrow{\scriptstyle \mathsf{view}_i} \\ (\widetilde{E_i} \cdot \widetilde{P_i})^* & \xrightarrow{\mathsf{cancel}_i} & (E_i \cdot P_i)^* \end{array}$$

Therefore, for any global play $w \in Dom(\widetilde{\sigma})$, and for any Process $i \in \{1, \ldots, n\}$ such that $w[i] \in P_i$, we have:

$$\begin{aligned} \widetilde{\sigma_i}(w[i]) &= \sigma_i(\mathsf{cancel}_i(w[i])) \\ &= \sigma_i(view_i \mathsf{cancel}(w)) \end{aligned}$$

For any infinite play w in $\widetilde{G}$ which is consistent with $\widetilde{\sigma}$, the corresponding play $\mathsf{cancel}(w)$ in G is consistent with σ, hence belongs to $\mathcal{W}$ when infinite. Then $w \in \widetilde{\mathcal{W}}$ comes directly. The strategy $\widetilde{\sigma}$ is therefore winning over G.

Lemma 2 (From synchronous to asynchronous). *For any finite state winning distributed strategy for the Process team over $\widetilde{G}$ there exists a finite state winning distributed strategy for the Process team over G with a memory of the same size.*

The problem in proving this lemma is that we will obviously have to cope with any local strategy over $\widetilde{G_i}$ ($i \in \{1, \ldots, n\}$) has the ability to somehow *count* the asynchronous moves, therefore getting additional information on the global play comparing to a local strategy over G.

The answer consists in showing that this counting is in any case useless, since each time the Environment can choose to play a totally asynchronous move. A Process i has therefore no interest in counting the local asynchronous moves, since he does not know whether they are *true* asynchronous moves (corresponding to asynchronous moves in G) or totally asynchronous ones.

The proof technique we use consists in *saturating* the memory of any distributed strategy over G, building in the process a distributed strategy that

behaves like the one over G would do if the Environment played a large number of totally asynchronous moves each time he has to play.

Suppose the following distributed strategy with finite memory is given: $\widetilde{\sigma} = \widetilde{\sigma_1} \otimes \cdots \otimes \widetilde{\sigma_n}$, and suppose it is winning over $\widetilde{G}$ with the following memories:

$$\widetilde{\mathcal{M}_i} = \langle \widetilde{M_i} \ , \ \widetilde{m_{0,i}} \in \widetilde{M_i} \ , \ \widetilde{\mu_i} : \widetilde{M_i} \times (\widetilde{P_i} \cup \widetilde{E_i}) \to \widetilde{M_i} \ , \ \widetilde{h_i} : \widetilde{M_i} \times \widetilde{P_i} \to \widetilde{E_i} \rangle$$

(for $(i \in \{1, \ldots, n\})$.

Since there are finitely many local strategies, each of them with finite memory, pumping lemma arguments show that: there exists an integer L such that for any Process $i \in \{1, \ldots, n\}$, for any memory element m in M_i, for any position $e \in \widetilde{E_i}$, the following holds:

$$\mu_i^*(m, (e \cdot \widehat{e})^L) = \mu_i^*(m, (e \cdot \widehat{e})^{k \cdot L}) \quad \text{for any integer } k > 0 \tag{1}$$

Now, consider the distributed strategy over G $\sigma = \sigma_1 \otimes \cdots \otimes \sigma_n$ with finite memory $\mathcal{M}_i = \langle M_i, m_{0,i}, \mu_i, h_i \rangle$, where $M_i = \widetilde{M_i}, m_{0,i} = \widetilde{m_{0,i}}, h_i = \widetilde{h_i}$, and:

$$\mu_i(m, e) = \widetilde{\mu_i}^*(m, e \cdot (\widehat{e} \cdot e)^{2 \cdot L - 1})$$
$$\mu_i(m, p) = \widetilde{\mu_i}(m, p)$$

for $e \in E_i$ and $p \in P_i$.

We are going to show that σ is winning over G. First of all, define the function $\mathsf{fill} : (E \cdot P)^* \to (\widetilde{E} \cdot \widetilde{P})^*$ as follows:

$$\mathsf{fill}(e_0 \cdot p_0 \cdot e_1 \cdot p_1 \cdots p_n) = e_0 \cdot (\widehat{e_0} \cdot e_0)^{2 \cdot L - 1} \cdot \widehat{p_0} \cdot e_1 \cdot (\widehat{e_1} \cdot e_1)^{2 \cdot L - 1} \cdot \widehat{p_1} \cdots \widehat{p_n}$$

fill can be generalized to infinite words in the same fashion than cancel.

Remark. fill is clearly a map from the plays where the Processes have to play in G to the plays of $\widetilde{G}$. It is moreover easy to figure out that $\mathsf{cancel} \circ \mathsf{fill}$, restricted to the plays of G, is the identity function, and that therefore $\mathsf{fill}(w) \in \widetilde{\mathcal{W}}$ implies $w \in \mathcal{W}$.

Last, the following fact tells that σ behaves over G exactly like $\widetilde{\sigma}$ does over $\widetilde{G}$ if the Environment plays $2 \cdot L - 1$ totally asynchronous moves each time it has to play.

Fact 2.1 *For any infinite play w in G consistent with σ, the play $\mathsf{fill}(w)$ is consistent with $\widetilde{\sigma}$.*

Knowing that $\widetilde{\sigma}$ is winning, and using remark above, we conclude that σ is winning.

Remark. σ is not more complex than $\widetilde{\sigma}$; it actually uses a memory of exactly the same size.

4 Synchronizing Linear Game

It is known that, in general, checking the existence of a winning distributed strategy for the Process team is undecidable, even in the case there are only two

Process players with reachability ($\mathcal{W} = \emptyset$) or safety ($\mathcal{W} = (E + P)^\omega$) winning condition [3]. However, when the information flows satisfies some linearity condition described below, the problem becomes decidable though non elementary [9].

In view of these properties, it occurs that our reduction of asynchronous distributed games to equivalent synchronous one is not that satisfactory. In fact, by introducing global non deterministic Environment moves everywhere, the linearity of the information flows in game G is lost in game $\widetilde{G}$.

We provide in this section a modification of our construction that do preserve such a linearity property (built upon the notion of i-sequentiality in [7]).

Definition. Given an n-Process distributed game $G = \langle P, E, T_P, T_E, e_0, \mathcal{W}\rangle$, we say that game G is a *distributed linear game* when for every $i \in [1, n]$, for every Environment positions e and f, for every Process team positions p and $q \in P$ such that $(e, p) \in T_E$ and $(f, q) \in T_E$:

> If $e[1, i] = f[1, i]$ and if $p[i] = q[i] \in P[i]$ or $p[i] \in E[i]$ or $q[i] \in E[i]$ then $p[1, i] = q[1, i]$.

This (local) linearity property first ensures that before every Environment moves, if a Process player i *knows* (in the epistemic sense) not only his own position $e[i]$ but also the position $e[1, i-1]$ of positions of every Process player with *lower* index, then this remain the case after any (synchronous or asynchronous) Environment move.

Moreover, since Process players knows each other strategies, this properties also ensures that, from a given starting position, given a fixed distributed strategy, every Process player knows (again in the epistemic sense), at any time he is active during a play, the position of every Process player of smaller index.

The next definition gives a construction on distributed games that, when applied to linear games, can be seen as a normalization process shifting from implicit knowledge to explicit knowledge.

Definition. Let $G = \langle P, E, T_P, T_E, e_0, \mathcal{W}\rangle$ be an n-process distributed game, and let $lin(G) = \langle P', E', T'_P, T'_E, e'_0, \mathcal{W}'\rangle$, called the *linearization* of G, be the game defined from game G as follows:

1. for every $i \in [1, n]$:
   (a) $P'_i = P[1, i] - E[1, i] + \{\bot_i\}$,
   (b) $E'_i = E[1, i]$,
   (c) $T'_{P,i} = T_P[1, i] \cap (P'_i \times E'_i)$,
2. and, for every $e \in E' = \prod_{i\in[1,n]} E'_i$:
   (a) either position e is *coherent w.r.t. game* G in the sense that for every $i \in [1, n]$ one has $e[i] = (e[n])[1, i]$, then we put $(e, p) \in T'_E$ for every $p \in P'$ such that $\forall i \in [1, n], (e[i], p[i]) \in T_E[1, i]$,
   (b) or position e is incoherent then we put $(e, \bot) \in T'_E$ with $\bot = (\bot_1, \cdots, \bot_n)$.
3. $e'_0 = (e_0[1], e_0[1, 2], \cdots, e_0[1, n-1], e_0[1, n])$,
4. and $\mathcal{W}' = \{w \in (P' + E')^\omega : w[n] \in \mathcal{W}\}$.

Remark. Observe that, in game $lin(G)$ any time the Process team reach an incoherent position e, Environment player moves to position $\perp$ where the Process team looses. It follows that relevant positions in game $lin(G)$ (positions where the Process team will play to win) are only coherent positions that is to say position $x \in E' + P'$ such that, given $y = x[n] \in P + E$, one has $x = (y[1], y[1,2], \cdots, y[1,n-1], y[1,n])$. In other words, in every global coherent position x of game $lin(G)$, Process i *explicitly knows* position $x[j]$ for every index j such that $1 \leq j \leq i$.

More formally, distributed strategies in game G and $lin(G)$ can be related as stated in the following two lemmas.

Lemma 3. *For every winning distributed strategy $\sigma_1 \otimes \cdots \otimes \sigma_n$ in game G, the distributed strategy $\sigma'_1 \otimes \cdots \otimes \sigma'_n$ in game $lin(G)$ defined, for every $i \in [1,n]$, by $\sigma'_i = \sigma_1 \otimes \cdots \otimes \sigma_i$, is a winning distributed strategy in game $lin(G)$.*

Proof. Immediate from definitions and remark above.

In general, there is no converse to such a lemma. In fact, games of the form $lin(G)$ are linear henceforth existence of winning distributed strategies is decidable which is not true for arbitrary game G. If, however game G is itself linear, a converse hold.

Lemma 4. *If game G is linear, for every winning distributed strategy $\sigma'_1 \otimes \cdots \otimes \sigma'_n$ in game $lin(G)$ there is a winning distributed strategy $\sigma_1 \otimes \cdots \otimes \sigma_n$ in game G such that, for every $i \in [1,n]$, $\sigma'_i = \sigma_1 \otimes \cdots \otimes \sigma_i$.*

Proof. Observe first that, because the distributed strategy $\sigma'_1 \otimes \cdots \otimes \sigma'_n$ is winning in game $lin(G)$ it only goes to coherent positions. Without lost of generality we can thus assume that the local strategies are themselves coherent. In other words, we can assume that, for every i and j with $1 \leq i < j \leq n$, for every global finite play w in game $lin(G)$, $\sigma'_i(view_i(w)) = \sigma'_j(view_j(w))[1,i]$

Now, the statement follows from the study of linear games presented in [1]. The distributed strategy $\sigma_1 \otimes \cdots \otimes \sigma_n$ is defined inductively.

First, strategy σ_1 is just defined to be strategy σ'_1. In fact, up to position $\perp_1$, local games $lin(G)[1]$ and $G[1]$ are essentially isomorphic.

Next, for every $i \in [2,n]$, strategy σ_i is inductively built from strategy $\sigma'_{i-1} = \sigma_1 \otimes \cdots \otimes \sigma_{i-1}$ and strategy σ'_i as follows. The key idea is to *simulate*, from the knowledge of the initial position e_0, the knowledge of strategy σ'_{i-1} and any local play w_i in $G[i]$, the (unique by linearity) play w that has been played on the projection $G[1,i]$ such that $w_i = view_i(w)$. Then, we put $\sigma_i(w_i) = \sigma'_i(w)$. Linearity ensures that this simulation can indeed be performed. □.

Now, it occurs that

Theorem 2. *For every n-Process* linear *distributed game G the game $lin(\widetilde{G})$ is linear and equivalent to game G in the sense that Process team has a finite memory winning strategy in game G if and only if it has one in game $lin(\widetilde{G})$.*

Proof. The proof arguments are similar to the proof arguments for Theorem 1. There, they have been detailed. Here, we only give a sketch of them.

First, any winning distributed strategy $\sigma_1 \otimes \cdots \otimes \sigma_n$ induces, by composing it with function cancel, a winning strategy in game $\widetilde{G}$ that, in turn, applying Lemma 3, induces a winning strategy in game $lin(\widetilde{G})$.

Conversely, assuming there is a finite state distributed winning strategy $\widetilde{\sigma}$ for the Process team in game $lin(\widetilde{G})$, it occurs that one can build, using similar pumping argument, a finite state distributed winning strategy σ' for the Process team in game $lin(G)$. Then, in turn, this strategy induces a finite state winning distributed strategy by applying Lemma 4.

References

1. Bernet, J., Janin, D.: Tree automata and discrete distributed games. In: Liśkiewicz, M., Reischuk, R. (eds.) FCT 2005. LNCS, vol. 3623, pp. 540–551. Springer, Heidelberg (2005)
2. Bernet, J., Janin, D.: Distributed synthesis in zero-delayed architectures with cycles. In: Najm, E., Pradat-Peyre, J.F., Donzeau-Gouge, V.V. (eds.) FORTE 2006. LNCS, vol. 4229, pp. 175–190. Springer, Heidelberg (2006)
3. Janin, D.: On the (high) undecidability of distributed synthesis problems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 320–329. Springer, Heidelberg (2007)
4. Kupferman, O., Vardi, M.Y.: Synthesizing distributed systems. In: LICS 2001. Proc. IEEE Symposium on Logic in Computer Science, pp. 389–398 (2001)
5. Lin, F., Wonham, M.: Decentralized control and coordination of discrete event systems with partial observation. IEEE Transactions on automatic control 33(12), 1330–1337 (1990)
6. Madhusudan, P., Thiagarajan, P.S.: Distributed controller synthesis for local specifications. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 396–407. Springer, Heidelberg (2001)
7. Mohalik, S., Walukiewicz, I.: Distributed games. In: Pandya, P.K., Radhakrishnan, J. (eds.) FST TCS 2003. LNCS, vol. 2914, pp. 338–351. Springer, Heidelberg (2003)
8. Perrin, D., Pin, J.E.: Infinite Words; Automata, Semigroups, Logic and Games. Pure and Applied Mathematics, vol. 141. Elsevier, Amsterdam (2004)
9. Peterson, G.L., Reif, J.H.: Multiple-person alternation. In: FOCS 1979. 20th Annual IEEE Symposium on Foundations of Computer Science, pp. 348–363 (October 1979)
10. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: FOCS 1990. Proc. 31th IEEE Symposium on Foundations of Computer Science, pp. 746–757 (1990)

Exact OBDD Bounds for Some Fundamental Functions
(Extended Abstract)

Beate Bollig, Niko Range, and Ingo Wegener

FB Informatik, LS2, Univ. Dortmund, 44221 Dortmund, Germany
beate.bollig@uni-dortmund.de, niko@range.at, ingo.wegener@uni-dortmund.de

Abstract. Ordered binary decision diagrams (OBDDs) are the most common dynamic data structure or representation type for Boolean functions. Among the many areas of application are verification, model checking, computer aided design, relational algebra, and symbolic graph algorithms. Although many even exponential lower bounds on the OBDD size of Boolean functions are known, there are only few functions where the OBDD size is even asymptotically known exactly. In this paper the exact OBDD sizes of the fundamental functions multiplexer and addition of n-bit numbers are determined.

Keywords: Computational complexity, lower bounds, ordered binary decision diagrams.

1 Introduction and Results

When working with Boolean functions as in circuit verification, synthesis, model checking, and even in graph algorithms, ordered binary decision diagrams, denoted OBDDs, introduced by Bryant (1986), are the most often used data structure supporting all fundamental operations on boolean functions.

Definition 1. *Let $X_n = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A variable order π on X_n is a permutation on $\{1, \ldots, n\}$ leading to the ordered list $x_{\pi(1)}, \ldots, x_{\pi(n)}$ of the variables.*

Definition 2. *A π-OBDD on X_n (see Fig. 1) is a directed acyclic graph $G = (V, E)$ whose sinks are labeled by Boolean constants and whose non sink (or inner) nodes are labeled by Boolean variables from X_n. Each inner node has two outgoing edges one labeled by 0 and the other by 1. The edges between inner nodes have to respect the variable order π, i.e., if an edge leads from an x_i-node to an x_j-node, $\pi^{-1}(i) \leq \pi^{-1}(j)$ (x_i precedes x_j in $x_{\pi(1)}, \ldots, x_{\pi(n)}$). Each node v represents a Boolean function $f_v : \{0,1\}^n \to \{0,1\}$ defined in the following way. In order to evaluate $f_v(b)$, $b \in \{0,1\}^n$, start at v. After reaching an x_i-node choose the outgoing edge with label b_i until a sink is reached. The label of this sink defines $f_v(b)$. The size of the π-OBDD G is equal to the number of its nodes.*

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 174–185, 2008.

Note, that OBDDs are not restricted to the representation of single-output functions. An OBDD represents a Boolean function $f \in B_{n,m}$ by representing simultaneously the outputs $f_1, f_2, \ldots, f_m$ of f.

The size of the reduced π-OBDD representing f is described by the following structure theorem (Sieling and Wegener (1993)). In order to simplify the description, we describe the theorem only for the special case where π equals the identity $id(i) = i$.

Theorem 1. *The number of x_i-nodes of the id-OBDD for $f = (f_1, \ldots, f_m)$ is the number s_i of different subfunctions $f_{j|x_1=a_1,\ldots,x_{i-1}=a_{i-1}}$, $1 \leq j \leq m$ and $a_1, \ldots, a_{i-1} \in \{0,1\}$, essentially depending on x_i (a function g depends essentially on x_i if $g_{|x_i=0} \neq g_{|x_i=1}$).*

The variable order π is not given in advance and we have the freedom (and the problem) to choose a good or even an optimal order for the representation of f. Let π-OBDD(f) denote the π-OBDD size of f.

Definition 3. *The OBDD size of f (denoted by OBDD(f)) is the minimum of all π-OBDD(f).*

It is an obvious aim to determine OBDD(f) for as many of the interesting functions f as exactly as possible. This is similar to other fundamental complexity measures, among them circuit size, formula size, monotone circuit size or algebraic complexity (for such results see Wegener (1987)). Although many even exponential lower bounds on the OBDD size of Boolean functions are known and the method how to obtain such bounds is simple, there are only few functions where the OBDD size is asymptotically known exactly (see, e.g., Bollig and Wegener (2000).) Surprisingly enough, there is only one paper presenting tight bounds on the OBDD size (Wegener (1984)) which has even been published before the notion OBDD was established. For several fundamental functions one believes to know the optimal variable order but has no proof for this conjecture. We start to fill this gap by determining exact OBDD bounds for two fundamental functions, namely multiplexer MUX_n, often also called direct storage access function DSA_n, and binary addition ADD_n.

Definition 4. *The multiplexer* MUX_n *(or direct storage access function* DSA_n*) is defined on $n + k$ variables $a_{k-1}, \ldots, a_0, x_0, \ldots, x_{n-1}$, where $n = 2^k$. The a-variables are called address variables and the x-variables data variables.*
$\mathrm{MUX}_n(a, x) = x_{|a|}$, *where $|a|$ is the number whose binary representation equals $(a_{k-1}, \ldots, a_0)$.*

Definition 5. *Binary addition* ADD_n*:*$\{0,1\}^{2n} \to \{0,1\}^{n+1}$ *maps two n-bit integers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$ to their sum. That is* $\mathrm{ADD}_n(x, y) = s_n \ldots s_0$ *where $x + y = s$, where $s = s_n \ldots s_0$.* $\mathrm{ADD}_{i,n}$ *computes the ith bit s_i of* ADD_n.

The results of the paper are the following ones.

Theorem 2. *OBDD*$(\mathrm{MUX}_n) = 2n + 1$.

Theorem 3. *OBDD*$(\mathrm{ADD}_1) = 6$ *and, for* $n \geq 2$, *OBDD*$(ADD_n) = 9n - 5$.

The upper bounds are contained in Wegener (2000) (Theorem 4.3.2 and Theorem 4.4.3). For binary addition the case $n = 1$ is special since the outputs of ADD_1 are symmetric functions ($x_0 \wedge y_0$ and $x_0 \oplus y_0$) and the π-OBDD size does not depend on π. Hence, it is sufficient to consider one of the two possible variable orders.

In Sections 2 and 3 the lower bounds are proved where it is essential to avoid an inspection of too many cases since the number of variable orders grows exponentially. The following simple observation will be helpful. Given an arbitrary variable order π the number of nodes labeled by a variable x in the π-OBDD representing a given function f is not smaller than the number of x-nodes in a π-OBDD representing any subfunction of f. Furthermore, the proofs of the lower bounds are based on Theorem 1 implying that we do not introduce a new lower bound method. However, we show how to solve some combinatorial problems in order to obtain more precise results than known before.

2 Tight Bounds for the OBDD Size of the Multiplexer

In this section, we determine a lower bound on the size of OBDDs for the representation of the multiplexer.

Lemma 1. *The size of an* OBDD *for the representation of the multiplexer is at least* $2n + 1$.

Proof
Let π be an arbitrary variable order. In order to simplify the description, we assume w.l.o.g. that the sequence of the address variables according to π is $a_0, a_1, \ldots, a_{k-1}$. This assumption is justified because of the observation that the size of an OBDD representing the multiplexer remains the same if we only change the positions of some address variables.

Since the multiplexer essentially depends on all data variables, for each variable x_i, $0 \leq i \leq n-1$, there is at least one node labeled by x_i. Moreover, there have to be two sinks. In the following, our aim is to prove that there exist for each address variable a_i at least 2^i further nodes representing non-constant subfunctions of the multiplexer, such that the number of nodes altogether in the OBDD is at least
$2 + n + \sum_{i=0}^{k-1} 2^i = 2 + n + 2^k - 1 = 2n + 1$.

We fix one of the address variables, called a_i, and use the following notation. Let $T_i(x)$ be the set of the x-variables tested before the variable a_i, $R_i(x)$ describes the set of the remaining x-variables. Now, we consider all possible assignments to the address variables $a_0, \ldots, a_{i-1}$. Our aim is to prove that there exists at least one further node for each assignment. The data variables are partitioned into 2^i disjoint groups such that the indices of the variables of each group agree in their binary representation to the corresponding assignment to the address variables $a_0, a_1, \ldots, a_{i-1}$. Let b_i be an assignment to the address

variables $a_0, a_1, \ldots, a_{i-1}$. The group G_{b_i} contains all data variables x_j such that the i least significant bits of the binary representation of j equals b_i. Obviously, $G_{b_i'} \cap G_{b_i''} = \emptyset$ for different assignments b_i' and b_i'' to the address variables $a_0, a_1, \ldots, a_{i-1}$.

For each assignment b_i to the address variables $a_0, \ldots, a_{i-1}$ we distinguish two cases.

Case 1: $G_{b_i} \cap R_i(x) \neq \emptyset$.
We show that there exists a subfunction corresponding to b_i that essentially depends on a_i, therefore there has to be one further node labeled by a_i in the π-OBDD representing the multiplexer.

For this reason, we consider the subfunction which corresponds to the following assignment to the variables. Let x_j be a variable in $G_{b_i} \cap R_i(x)$. The assignment to the address variables $a_0, a_1, \ldots, a_{i-1}$ is b_i, the assignment to all data variables in $T_i(x)$ is 0. Obviously, the corresponding subfunction essentially depends on x_j. Therefore, different assignments to the address variables $a_0, a_1, \ldots, a_{i-1}$ lead to different subfunctions which have to be represented at different nodes in the π-OBDD. Furthermore, the considered subfunction essentially depends on a_i, since the assignment 1 to x_j, 0 to all other data variables, and the binary representation of j to the address variables has the function value 1 but changing only the assignment to a_i leads to the function value 0.

Altogether, we have shown that there has to be one further a_i-node in the π-OBDD representing the multiplexer.

Case 2: $G_{b_i} \cap R_i(x) = \emptyset$.
This case is more difficult because it is possible that there does not exist a subfunction for which the assignment to the variables $a_0, a_1, \ldots, a_{i-1}$ agrees with b_i and which essentially depends on a_i. We have to inspect this case very carefully in order to guarantee that we count each node of the π-OBDD representing the multiplexer only once.

Let b_i^j, $j \leq i$, be the assignment to the variables $a_0, a_1, \ldots, a_{j-1}$ according to b_i. Let i' be the minimum number in $\{0, \ldots, i\}$ such that $G_{b_i^{i'}} \cap R_{i'}(x) = \emptyset$. Since $G_{b_i} \subseteq G_{b_i^{i'}}$ we know that $G_{b_i} \subseteq T_{i'}(x)$. Now, we consider the assignment $b_i^{i'-1}$ which is unique for b_i. Let $x_{j'}$ be the jth data variable of the set $G_{b_i^{i'}}$ in the sequence according to π. Our aim is to show that there are at least 2^{j-1} $x_{j'}$-nodes in the π-OBDD representing the multiplexer. For this reason, we consider the following 2^{j-1} different assignments to the first variables of the set $G_{b_i^{i'}}$ which are before $x_{j'}$ in the sequence according to π. The address variables are set according to the assignment $b_i^{i'-1}$, the data variables in $T_{i'}(x) \setminus G_{b_i^{i'}}$ are fixed to 0 and for the first $j-1$ variables from $G_{b_i^{i'}}$ according to π we consider all possible assignments. Obviously, the corresponding subfunctions essentially depend on $x_{j'}$. Furthermore, two different assignments to the first $j-1$ data variables of $G_{b_i^{i'}}$ according to π lead to different subfunctions, since each of these data variables can determine the output of the multiplexer. For this we consider the following assignments to the remaining variables. Let $x_{k'}$ be one of

the first $j-1$ data variables under consideration. The remaining data variables are set to 0 and the address variables are set to the binary representation of k'. The output of the multiplexer is equal to the assignment of $x_{k'}$.

Using the fact that $|G_{b_i^{i'}}| = 2^{k-(i'-1)}$ we can conclude that there are at least

$$\sum_{j=1}^{2^{k-(i'-1)}} 2^{j-1} = \sum_{j=0}^{2^{k-i'}} 2^j = 2^{2^{k-i'}+1} - 1$$

nodes labeled by a data variable from $G_{b_i^{i'}}$ in the π-OBDD representing the multiplexer. We have already counted one node for each data variable, therefore we have shown that there are at least

$$(2^{2^{k-i'}+1} - 1) - 2^{k-(i'-1)}$$

further nodes.

On the other hand, there are $2^{\ell-(i'-1)}$ assignments b_ℓ to the address variables $a_0, a_1, \ldots, a_{\ell-1}$, $\ell \geq i'-1$, such that $b_i^{i'-1}$ is equal to $b_\ell^{i'-1}$. Therefore, we can conclude that there are

$$\sum_{\ell=i'-1}^{k-1} 2^{\ell-(i'-1)} = \sum_{\ell=0}^{k-i'} 2^\ell = 2^{k-i'+1} - 1$$

assignments to the address variables corresponding to $b_i^{i'-1}$ that lead to case 2 in our investigation.

Since $2^{2^{k-i'+1}} - 1 \geq 2^{k-i'+2} - 1$, we are done. □

3 Tight Bounds for the OBDD Size of Binary Addition

As noted before, Wegener (2000) has already presented the upper bound of $9n-5$ on the OBDD size of binary addition for two n-bit numbers where $n \geq 2$. In the following, we prove the matching lower bound. Figure 1 shows an OBDD for the binary addition of two 4-bit numbers according to an optimal variable order.

In order to obtain lower bounds on the size of OBDDs one-way communication complexity has become a standard technique (see Hromkovič (1997) and Kushilevitz and Nisan (1997) for the theory of communication theory). In the following, we do not really use methods from communication theory but the notion of a communication matrix which is nothing else but the value table of a function in a different form. A function $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$ can be described by a matrix of size $2^m \times 2^n$. The matrix entry at position (a,b), $a \in \{0,1\}^m$ and $b \in \{0,1\}^n$, is $f(a,b)$. The number of different rows is equal to the number of different subfunctions obtained by the replacement of the first m variables by constants. Since each column is associated with an assignment to the last n variables, a row corresponds to a subfunction essentially depending on a variable z iff there exist two columns associated with two assignments that differ only in the assignment of z and for which the entries in the matrix are different.

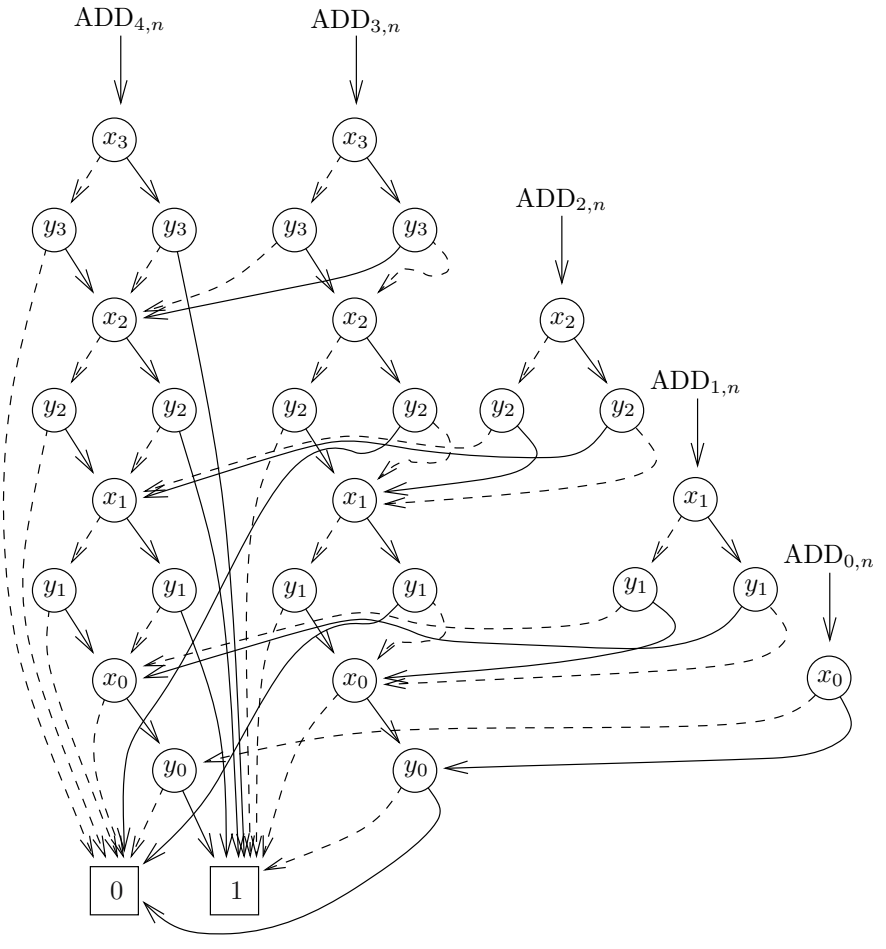


Fig. 1. An OBDD for the binary addition of 4-bit numbers (dotted edges are edges with label 0 and solid edges are edges with label 1)

Since the functions $\mathrm{ADD}_{i,n}$, $0 \leq i \leq n$, are different and non-constant there are at least $n+1$ nodes representing $\mathrm{ADD}_{i,n}$ in an OBDD representing binary addition. Our aim is to show that for almost all pairs (x_i, y_i), $0 \leq i \leq n-1$, there exist at least 8 nodes labeled by x_i or y_i not representing one of the functions $\mathrm{ADD}_{i,n}$, $0 \leq i \leq n$. Together with the two sinks we are done.

We start our investigation with two simple observations. Let π be an arbitrary variable order. Symmetric variables for a given function f are variables that can be exchanged without changing the considered function, i.e. the variables z_i and z_j are symmetric variables for f when $f_{|z_i=0,z_j=1} = f_{|z_i=1,z_j=0}$. In order to simplify the description, we assume w.l.o.g. that for each variable pair (x_i, y_i), $0 \leq i \leq n-1$, the variable x_i ist tested before the variable y_i according to π. This

assumption is justified because of the observation that x_i and y_i are symmetric variables for binary addition.

Since the functions $\mathrm{ADD}_{i,n}$, $0 \leq i \leq n-1$, essentially depend on the variables $x_0, y_0, x_1, \ldots, x_i, y_i$ and $\mathrm{ADD}_{n,n}$ essentially depends on all variables, none of the functions $\mathrm{ADD}_{i,n}$, $0 \leq i \leq n$, can be represented at a node labeled by a y-variable.

Now, we introduce some useful notation. Let X be the set of all x-variables and Y the set of all y-variables. The set $X^{>i}$ contains the variables $x_{i+1}, \ldots, x_{n-1}$. Similar the sets $Y^{>i}$, $X^{<i}$, and $Y^{<i}$ are defined. Let $\Pi_{x_i} = (A_{x_i}, B_{x_i})$, $0 \leq i \leq n-1$, be a partition of the variables in $X \cup Y$ according to a given variable order π, where A_{x_i} contains all variables that are tested before x_i according to π and B_{x_i} the remaining variables. For a subset $S \subseteq X \cup Y$, we denote by $\mathcal{A}(S)$ the set of all possible assignments to the variables in S.

First, we present lower bounds on the number of x-nodes not representing one of the functions $\mathrm{ADD}_{i,n}$, $0 \leq i \leq n$. Afterwards the number of y-nodes is investigated. The general proof strategy is the following one. Using some assumptions suitable assignments to the variables in A_z are identified that lead to a sufficient number of different subfunctions. By carefully choosing assignments to the variables in B_z it is proved that the considered subfunctions are really different and that they essentially depend on the variable z. The consideration of the communication matrices for the functions $\mathrm{ADD}_{i,n} : \mathcal{A}(A_z) \times \mathcal{A}(B_z) \to \{0,1\}$, $0 \leq i \leq n$, simplified the investigations.

The missing proofs can be found in the full version of the paper [1].

Lemma 2. *Let π be an arbitrary variable order and $\Pi_{x_i} = (A_{x_i}, B_{x_i})$ be a partition of the variables in $X \cup Y$ according to π and an arbitrary chosen variable x_i, $0 \leq i \leq n-1$. Let G be a π-OBDD representing ADD_n. If $A_{x_i} \cap X^{>i} \neq \emptyset$, the number of x_i-nodes in G not representing one of the functions $\mathrm{ADD}_{j,n}$, $0 \leq j \leq n$, is at least 2.*

Lemma 3. *Let π be an arbitrary variable order and $\Pi_{x_i} = (A_{x_i}, B_{x_i})$ be a partition of the variables in $X \cup Y$ according to π and an arbitrary chosen variable x_i, $0 \leq i \leq n-1$. Let G be a π-OBDD representing ADD_n. If $A_{x_i} \cap X^{<i} \neq \emptyset$, the number of x_i-nodes in G not representing one of the functions $\mathrm{ADD}_{j,n}$, $0 \leq j \leq n$, is at least 4.*

Combining Lemma 2 and Lemma 3 we obtain the following result.

Corollary 1. *Let π be an arbitrary variable order and let G be a π-OBDD representing ADD_n. The number of x-nodes in G not representing one of the functions $\mathrm{ADD}_{j,n}$, $0 \leq j \leq n$, is at least $2n-2$.*

Lemma 4. *Let π be an arbitrary variable order and $\Pi_{y_i} = (A_{y_i}, B_{y_i})$ be a partition of the variables in $X \cup Y$ according to π and an arbitrary chosen variable y_i, $0 \leq i \leq n-1$. Let G be a π-OBDD representing ADD_n. If $A_{y_i} \cap X^{>i} \neq \emptyset$ and $B_{y_i} \cap Y^{<i} \neq \emptyset$, the number of y_i-nodes in G is at least 6.*

Lemma 5. *Let π be an arbitrary variable order and $\Pi_{y_i} = (A_{y_i}, B_{y_i})$ be a partition of the variables in $X \cup Y$ according to π and an arbitrary chosen variable y_i, $0 \leq i \leq n-1$, where $A_{y_i} \cap X^{>i} \neq \emptyset$ and $B_{y_i} \cap Y^{<i} = \emptyset$. Let G be a π-OBDD representing* ADD_n.

i) The number of y_i-nodes in G is at least 2.
ii) If $|B_{y_i} \cap Y| > 1$, there are at least $2 + |B_{y_i} \cap Y|$ nodes labeled by y_i.
iii) If $|B_{y_i} \cap Y| \in \{2, 3\}$ and $|B_{y_i} \cap X| < |B_{y_i} \cap Y| - 1$, the number of y_i-nodes in G is at least 6.
iv) Let $|B_{y_i} \cap Y| = 3$ and let y_r be the variable in B_{y_i}, where $B_{y_r} \subset B_{y_i}$ and $|B_{y_r}| > 1$, with other words the variable y_i ist tested before y_r but y_r is not the last variable according to π. If $|B_{y_i} \cap X| = |B_{y_i} \cap Y| - 1 = 2$, there are at least 4 y_r-nodes in G and 4 x_r-nodes not representing one of the functions $\mathrm{ADD}_{j,n}$, $0 \leq j \leq n$.

In the following, we show that for almost all pairs (x_i, y_i), $0 \leq i \leq n-1$, the number of nodes not representing one of the functions $\mathrm{ADD}_{j,n}$, $0 \leq j \leq n$, and labeled by x_i or y_i is 8 if $A_{y_i} \cap X^{>i} = \emptyset$.

Lemma 6. *Let π be an arbitrary variable order and $\Pi_{y_i} = (A_{y_i}, B_{y_i})$ be a partition of the variables in $X \cup Y$ according to π and an arbitrary chosen variable y_i, $0 \leq i \leq n-1$, where $A_{y_i} \cap X^{>i} = \emptyset$. Let G be a π-OBDD representing* ADD_n.

i) If $A_{x_i} \neq \emptyset$, the number of x_i-nodes in G not representing one of the functions $ADD_{j,n}$, $0 \leq j \leq n$, is at least 4.
ii) The number of y_i-nodes in G is at least 2.
iii) If $i \leq n-2$, the number of y_i-nodes in G is at least 4.
iv) If $|B_{y_i} \cap Y| > 1$, the number of y_i-nodes in G is at least 4.

Proof. *i)* Since $A_{x_i} \neq \emptyset$ and $A_{y_i} \cap X^{>i} = \emptyset$, we know that there exist a variable x_k before x_i according to π where $k < i$. Therefore, we can apply Lemma 3 and obtain at least 4 x_i-nodes not representing one of the functions $\mathrm{ADD}_{j,n}$, $0 \leq j \leq n$.

ii) We consider two assignments to the variables in A_{y_i} that differ only in the assignment to the variable x_i. In a_j, $j \in \{0, 1\}$, the variable x_i is set to j, the remaining x-variables in A_{y_i} are set to 1, the y-variables in A_{y_i} are set to 0. Our aim is to prove that the subfunctions $\mathrm{ADD}_{i,n|a_0}$ and $\mathrm{ADD}_{i,n|a_1}$ are different and that they essentially depend on y_i. For this reason, we consider the following two assignments to the variables in B_{y_i} that differ only in the assignment to the variable y_i. In b_j, $j \in \{0, 1\}$, the variable y_i is set to j, the x-variables in B_{y_i} are set to 1 and the remaining y-variables are set to 0. Table 1 show part of the communication matrix for $\mathrm{ADD}_{i,n}$. Obviously, the two subfunctions are different and essentially depend on y_i.

iii) Using part *ii)* it remains to prove that there are two further nodes labeled by y_i. Since $A_{y_i} \cap X^{>i} = \emptyset$, we can conclude that there exist a variable y_h, $h > i$, in $B_{y_i} \cap Y$. Now, we consider the subfunctions $\mathrm{ADD}_{h,n|a_1}$ and $\mathrm{ADD}_{n,n|a_1}$ for a_1 chosen as in part *ii)*. Both essentially depend on y_h and are therefore

Table 1. Part of the communication matrix for $\mathrm{ADD}_{i,n}$

$\mathrm{ADD}_{i,n}$	b_0	b_1
a_0	0	1
a_1	1	0

Table 2. Part of the communication matrix for $\mathrm{ADD}_{h,n}$ and $\mathrm{ADD}_{n,n}$

$\mathrm{ADD}_{h,n}$	b_0	b_1
a_1	1	0

$\mathrm{ADD}_{n,n}$	b_0	b_1
a_1	0	1

different from the subfunctions considered in part *ii)*. Table 2 shows part of the communication matrix for $\mathrm{ADD}_{h,n}$ and $\mathrm{ADD}_{n,n}$. Obviously, both subfunctions are different and essentially depend on y_i. Together with the proof of part *ii)* we obtain at least 4 y_i-nodes.

iv) We assume $i = n-1$, otherwise we can use part *iii)* and we are done. As in part *ii)* we consider the assignments a_0 and a_1. Using the proof of part *ii)* we know that there have to be 2 y_{n-1}-nodes representing the subfunctions $\mathrm{ADD}_{n-1,n|a_0}$ and $\mathrm{ADD}_{n-1,n|a_1}$. Our aim is to prove that there have to be two further y_{n-1}-nodes in G representing the subfunctions $\mathrm{ADD}_{n,n|a_0}$ and $\mathrm{ADD}_{n,n|a_1}$. Since $|B_{y_{n-1}} \cap Y| > 1$, there has to be a variable y_l in $B_{y_{n-1}}$, where $l < n-1$. We consider the following four assignments to the variables in $B_{y_{n-1}}$ that differ only in the assignments to the variables y_l and y_{n-1}. In $b_{j_1 j_2}$, $j_1, j_2 \in \{0,1\}$, the variable y_l is set to j_1 and the variable y_{n-1} to j_2. The x-variables are set to 1 and the remaining y-variables are set to 0.

Figure 2 illustrates the replacement of the variables with the exception of x_{n-1}, y_{n-1}, and y_l by constants.

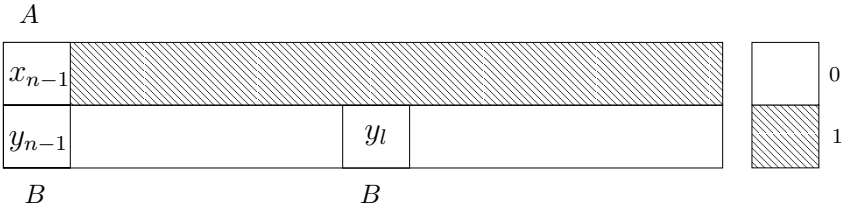


Fig. 2. The replacement of some of the variables by constants in the proof of Lemma 6

Table 3 shows part of the communication matrix for $\mathrm{ADD}_{n-1,n}$ and $\mathrm{ADD}_{n,n}$.

Obviously, the four subfunctions are different and essentially depend on y_{n-1}, therefore there are at least 4 nodes labeled by y_{n-1} in G. □

Table 4 illustrates the minimal number of nodes not representing one of the functions $\mathrm{ADD}_{j,n}$, $0 \le j \le n$, and labeled by x_i or y_i if $A_{y_i} \cap X^{>i} = \emptyset$.

Table 3. Part of the communication matrix for $\mathrm{ADD}_{n-1,n}$ and $\mathrm{ADD}_{n,n}$

$\mathrm{ADD}_{n-1,n}$	b_{00}	b_{01}	b_{10}	b_{11}
a_0	0	1	1	0
a_1	1	0	0	1

$\mathrm{ADD}_{n,n}$	b_{00}	b_{01}	b_{10}	b_{11}
a_0	0	0	0	1
a_1	0	1	1	1

Table 4. The minimal number of x_i- and y_i-nodes if $A_{y_i} \cap X^{>i} = \emptyset$

	$A_{x_i} = \emptyset$	$A_{x_i} \neq \emptyset$
$i = n-1 \wedge \lvert B_{y_i} \cap Y \rvert = 1$	2	6
$i \leq n-2 \vee \lvert B_{y_i} \cap Y \rvert > 1$	4	8

In order to prove the lower bound for binary addition, we only have to combine Lemma 2-6 in a sufficient way. In doing so, we have to rule out the possibility that there are too many y-variables for which we cannot guarantee 6 nodes.

Theorem 4. *The size of an* OBDD *for the representation of binary addition is at least* $9n-5$ *for* $n \geq 2$.

Proof. Let π be an arbitrary variable order and G be a π-OBDD representing ADD_n. Our aim is to prove that G has at least $9n-5$ nodes. There are $(n+1)+2$ nodes in G representing the functions $\mathrm{ADD}_{i,n}$, $0 \leq i \leq n$, and the constant functions 0 and 1. Using Corollary 1 we obtain at least $2n-2$ further nodes labeled by an x-variable. If we can prove that there are at least $6n-6$ further nodes, we are done.

Now, we investigate the number of y-nodes in G. For each variable y_i, $0 \leq i \leq n-1$, exactly one of the Lemmas 4-6 can be applied and for almost all variables y_i it can be proved that G contains at least 6 y_i-nodes. In the following, we book for each variable x_i for which we can prove that there exist at least 4 x_i-nodes in G not representing one of the functions $\mathrm{ADD}_{i,n}$, $0 \leq i \leq n$, 2 nodes by the variable y_i. Next, we look more carefully at the y-variables for which we cannot directly guarantee using Lemma 4-6 that there are at least 6 nodes in G. Figure 3 illustrates all possible cases where the number of nodes accounted for a y-variable can be less than 6.

- C_1 is the set of y_i-variables for which $A_{y_i} \cap X^{>i} \neq \emptyset$ and $\lvert B_{y_i} \cap Y \rvert = 1$.
- C_2 is the set of y_i-variables for which $A_{y_i} \cap X^{>i} \neq \emptyset$, $B_{y_i} \cap Y^{<i} = \emptyset$, and $\lvert B_{y_i} \cap Y \rvert \in \{2,3\}$, where $\lvert B_{y_i} \cap X \rvert = \lvert B_{y_i} \cap Y \rvert - 1$.

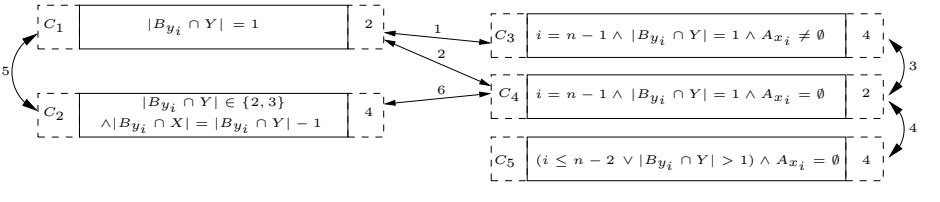


Fig. 3. Possible sets of y-variables for which less than 6 nodes can be directly booked

- C_3 contains the variable y_{n-1} if $|B_{y_{n-1}} \cap Y| = 1$ and $A_{x_{n-1}} \neq \emptyset$.
- C_4 contains the variable y_{n-1} if $|B_{y_{n-1}} \cap Y| = 1$ and $A_{x_{n-1}} = \emptyset$.
- C_5 is the set of y_i-variables, $i \leq n-2$, where $A_{y_i} \cap X^{>i} = \emptyset$, $|B_{y_i} \cap Y| > 1$, and $A_{x_i} = \emptyset$.

The right column of a C_j-row, $1 \leq j \leq 5$, presents the minimal number of nodes accounted for a variable in C_j. Obviously, each variable y_i can be in at most one set C_j, $j \in \{1, \ldots, 5\}$. On the other hand, $|C_j| \leq 1$ for $j \in \{1, 3, 4, 5\}$. Lemma 5 part *iv)* guarantees that C_2 can contain at most one variable. An arrow between a C_{j_1}- and a C_{j_2}-row, $j_1, j_2 \in \{1, 2, \ldots, 5\}$, indicates that there cannot be a variable in C_{j_1} for which only the minimal number of nodes can be accounted for, as well as in C_{j_2}.

It is not difficult to see that $|C_1| + |C_3| + |C_4| \leq 1$, therefore the arrows $1-3$ are justified. Furthermore, using the definition of the sets we can immediatley conclude that $|C_4| + |C_5| \leq 1$ and the arrow 4 follows. For the fifth arrow we have to work a little bit harder.

Claim
If $C_1 \neq \emptyset$ and $C_2 \neq \emptyset$, the minimal number of nodes accounted for the variables in $C_1 \cup C_2$ is at least 8.

Proof. Let y_i be the variable in C_1 and y_j be the variable in C_2. Because of the definition of C_2 we know that $i > j$ and $x_i \in B_{y_j}$. Therefore, we can apply Lemma 3 in order to prove that there are at least 4 x_i-nodes. □

In order to prove that $|C_2| + |C_4| \leq 1$ and therefore the arrow 6 is justified, we assume that $C_2 \neq \emptyset$. Let $y_j \in C_2$ and $y_k \in B_{y_i}$, with other words the variable y_i is tested before the variable y_k according to π. Because of the definition of C_2, more precisely, since $|B_{y_i} \cap X| = |B_{y_i} \cap Y| - 1$, it follows that also $x_k \in B_{y_i}$ and therefore $A_{x_k} \neq \emptyset$. Therefore, the set C_4 has to be empty.

Summarizing, we obtain the following results:

- $|C_i| \leq 1$, $i \in \{1, 2, \ldots, 5\}$.
- If $C_1 \neq \emptyset$, we know that $C_2 = C_3 = C_4 = \emptyset$.
- If $C_4 \neq \emptyset$, it follows that $C_1 = C_2 = C_3 = C_5 = \emptyset$.

Altogether, we have proved that the number of nodes in G accounted for a y-variable is at least $6n - 6$. □

References

1. Bollig, B., Range, N., Wegener, I.: Exact OBDD bounds for some fundamental functions. ECCC TR07-049 (2007)
2. Bollig, B., Wegener, I.: Asymptotically optimal bounds for OBDDs and the solution of some basic OBDD problems. Journal of Computer and System Sciences 61, 558–579 (2000)
3. Bryant, R.E.: Graph-based algorithms for Boolean manipulation. IEEE Trans. on Computers 35, 677–691 (1986)

4. Hromkovič, J.: Communication Complexity and Parallel Computing. Springer, Heidelberg (1997)
5. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)
6. Sieling, D., Wegener, I.: NC-algorithms for operations on binary decision diagrams. Parallel Processing Letters 48, 139–144 (1993)
7. Wegener, I.: Optimal decision trees and one-time-only branching-programs for symmetric Boolean functions. Information and Control 62, 129–143 (1984)
8. Wegener, I.: The Complexity of Boolean Functions. Wiley-Teubner (1987)
9. Wegener, I.: Branching Programs and Binary Decision Diagrams - Theory and Applications. SIAM Monographs on Discrete Mathematics and Applications (2000)

Clustering-Based Similarity Search in Metric Spaces with Sparse Spatial Centers

Nieves Brisaboa[1], Oscar Pedreira[1], Diego Seco[1],
Roberto Solar[2,3], and Roberto Uribe[2,3]

[1] Database Laboratory, University of A Coruña
Campus de Elviña s/n, 15071 A Coruña, Spain
[2] Dpto. Ingeniería en Computación, Universidad de Magallanes
Casilla 113-D, Punta Arenas, Chile
[3] Grupo de Bases de Datos (UART), Universidad Nacional de la Patagonia Austral
Rio Turbio, Santa Cruz, Argentina
{brisaboa,opedreira,dseco}@udc.es, {rsolar,ruribe}@ona.fi.umag.cl

Abstract. Metric spaces are a very active research field which offers efficient methods for indexing and searching by similarity in large data sets. In this paper we present a new clustering-based method for similarity search called SSSTree. Its main characteristic is that the centers of each cluster are selected using *Sparse Spatial Selection* (SSS), a technique initially developed for the selection of pivots. SSS is able to adapt the set of selected points (pivots or cluster centers) to the intrinsic dimensionality of the space. Using SSS, the number of clusters in each node of the tree depends on the complexity of the subspace it represents. The space partition in each node will be made depending on that complexity, improving thus the performance of the search operation. In this paper we present this new method and provide experimental results showing that SSSTree performs better than previously proposed indexes.

Keywords: Similarity search, metric spaces, sparse spatial selection, cluster center selection.

1 Introduction

Searching in metric spaces is a very active research field since it offers efficient methods for indexing and searching by similarity in non-structured domains. For example, multimedia databases manage objects without any kind of structure like images, audio clips or fingerprints. Retrieving the most similar fingerprint to a given one is a typical example of similarity search. The problem of text retrieval is present in systems that range from a simple text editor to big search engines. In this context we can be interested in retrieving words similar to a given one to correct edition errors, or documents similar to a given query. We can find more examples in areas such as computational biology (retrieval of DNA or protein sequences) or pattern recognition (where a pattern can be classified from other previously classified patterns) [1,2].

The problem of similarity search can be formalized through the concept of metric spaces. A metric space $(\mathbb{X}, d)$ is composed of a universe of valid objects $\mathbb{X}$ and

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 186–197, 2008.

a distance function $d : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}^+$ satisfying the properties of *non-negativity* ($d(x, y) > 0$ and if $d(x, y) = 0$ then $x = y$), *symmetry* ($d(x, y) = d(y, x)$) and the *triangle inequality* ($d(x, z) \leq d(x, y) + d(y, z)$). The distance function determines the similarity between any two objects from that universe. A collection of words with the edit distance (computed as the number of characters to insert, delete or modify to transform a word into another), is an example of a metric space. A *vector space* is a particular case of a metric space, in which each object is composed of k real numbers. In vector spaces we can use any distance function from the family $L_s(x, y) = (\sum_{1 \leq i \leq k} |x_i - y_i|^s)^{\frac{1}{s}}$. For example, L_1 is the *Manhattan* distance, L_2 is the Euclidean distance and $L_\infty = max_{1 \leq i \leq k} |x_i - y_i|$ is the maximum distance. The *dimensionality* of a vector space is the number of components of each vector. Although general metric spaces do not have an explicit dimensionality, we can talk about their *intrinsic dimensionality* following the idea presented in [1]. The higher the dimensionality the more difficult the search.

Similarity search can involve different types of queries. Range search retrieves the objects that are within distance r (query range) to the query q, i.e., $\{u \in \mathbb{U} / d(q, u) \leq r\}$. *k-nearest neighbor* search retrieves the k nearest objects to the query q, i.e., the set $A \subseteq \mathbb{U}$ such that $|A| = k$ y $\forall\ u \in A,\ v \in \mathbb{U} - A,\ d(q, u) \leq d(q, v)$. Range search is the most general and k-nearest neighbor search can be implemented in terms of it [1].

Similarity search can be trivially implemented comparing the query with all the objects of the collection. However, the high computational cost of the distance function, and the high number of times it has to be evaluated, makes similarity search very inefficient with this approach. This has motivated the development of indexing and search methods in metric spaces that make this operation more efficient trying to reduce the number of evaluations of the distance function. This can be achieved storing in the index information that, given a query, can be used to discard a significant amount of objects from the data collection without comparing them with the query.

Although reducing the number of evaluations of the distance function is the main goal of indexing algorithms, there are other important features. Some methods can only work with discrete distance functions while others admit continuous distances too. Some methods are static, since the data collection cannot grow once the index has been built. Dynamic methods support insertions in an initially empty collection. Another important factor is the possibility of efficiently storing these structures in secondary memory, and the number of I/O operations needed to access them.

Search methods in metric spaces can be grouped in two classes [1]: *pivot*-based and *clustering*-based search methods. Pivot-based methods select a subset of objects from the collection as pivots, and the index is built computing and storing the distances from each of them to the objects of the database. During the search, this information is used to discard objects from the result without comparing them with the query. The most important pivot-based methods are *Burkhard-Keller-Tree* (BKT) [3], *Fixed-Queries Tree* (FQT) [4], *Fixed-Height FQT* (FHQT) [5], *Fixed-Queries Array* (FQA) [6], *Vantage Point Tree* (VPT)

[7] and its variants [8] [9], *Approximating and Eliminating Search Algorithm* (AESA) [10] and LAESA (*Linear AESA*) [11].

Clustering-based methods partition the metric space in a set of regions or clusters, each of them represented by a cluster center. In the search, complete regions are discarded from the result based on the distance from their center to the query. The most important clustering methods are *Bisector Tree* (BST) [12], *Generalized-Hyperplane Tree* (GHT) [13] and *Geometric Near-neighbor Access Tree* (GNAT) [14]. In section 2 we will briefly review how these methods work.

This paper presents a new index structure for searching in metric spaces called SSSTree. It is a clustering-based search method, and its main feature is that the cluster centers are selected applying *Sparse Spatial Selection* (SSS) [15], a strategy initially designed for pivot selection. SSS is adaptive, dynamic and the set of selected objects are well-distributed in the metric space. Applying SSS, the SSSTree is not balanced, but it is adapted to the complexity of the space, which is an important difference with previously proposed methods. Our hypothesis is that, applying SSS to select the cluster centers, the space partition will be more efficient and the search operation will show a better performance. Experimental results show that this new approach performs better than previously proposed methods.

Next Section briefly explains some basic concepts about clustering-based methods for similarity search. Section 3 analyzes the problem of pivot and cluster center selection and describes SSS. Section 4 presents SSSTree. In Section 5 experimental results are discussed and Section 6 finishes the paper with the conclusions and future work.

2 Previous Work on Clustering-Based Similarity Search

Clustering-based search methods partition the metric space in several regions or clusters, each of them represented by a cluster center. The index stores the information of each cluster and its center. Given a query, complete clusters can be discarded from the result based on the distance from their centers to the query. In those clusters that can not be discarded, the query is sequentially compared with all the objects of the cluster.

There are two pruning criteria to delimit the clusters in clustering-based methods: generalized hyperplane and covering radius. The algorithms based in the hyperplane partitioning divide the metric space in a Voronoi partition. Given a set of cluster centers $\{c_1, c_2, \ldots, c_n\} \subset \mathbb{X}$, the Voronoi diagram is defined as the space subdivision in n areas in such a way that, $x \in Area(c_i)$ if and only if $d(x, c_i) < d(x, c_j)$, for $j \neq i$. In this type of algorithms, given a query (q, r), the clusters with no objects in the result set are directly discarded based in the query range and the distance from the query to the hyperplane that separates that cluster from the next one.

In the case of the algorithms that use the covering radius for pruning, the space is divided in a set of spheres that can intersect, and a query can be included in more than one sphere. The covering radius is the distance from the cluster center to the furthest object from it. Knowing the distance from the query to

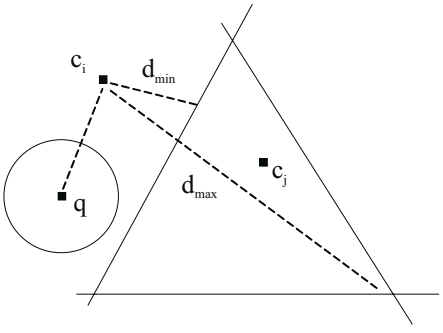


Fig. 1. Use of the max and min distances to discard objects in GNAT

the cluster center, the query range, and the covering radius, we can decide if that cluster contains objects in the result set or if it does not.

Bisector Tree (BST) [12] recursively divides the space storing the information about the clusters in a binary tree. In the root of the tree, two cluster centers are selected to make the division. The objects that belong to the first cluster are assigned to the left child node, and those that belong to the second cluster are assigned to the right child. This process is repeated in each node to recursively partition the space. For each cluster, the cluster center and the covering radius are stored. Given a query, the tree is traversed deciding in each node what branches (clusters) can be discarded from the result. *Generalized Hyperplane Tree* (GHT) [13] also applies a recursive partitioning to the space. However, during the search GHT does not use the covering radius to determine what clusters can be discarded. In this case, this decision is taken using the hyperplane place between the two cluster centers stored in each node.

Geometric Near-neighbor Access Tree (GNAT) [14] uses m cluster centers in each internal node of the tree. Each node of the tree stores a table with m rows (one for each cluster center) and m columns (one for each cluster). Cell (i, j) stores the minimum and maximum distances from the cluster center c_i to an object belonging to $Cluster_j$. During the search, the query q is compared with a cluster center c_i. Now, we can discard the clusters $Cluster_j$ such that $d(q, c_i)$ is not between that minimum and maximum distances. Figure 1 intuitively shows the meaning of these distances and how they can be used to decide in which clusters continue the search.

EGNAT [16] belongs to the group of algorithms based on compact partitions and is a secondary memory optimization of GNAT, in terms of space, disk accesses and distance evaluations.

3 The Problem of Pivots and Cluster Centers Selection

3.1 Previous Work

Something that most of the algorithms we have mentioned have in common is that both pivots and cluster centers are usually selected at random. However, it

is evident that the specific set of selected reference objects has a strong influence in the efficiency of the search. The number of objects, their position in the space and their position with respect to the others, determine the ability of the index for discarding objects without comparing them with the query. Other important problem is how to determine the optimal number of reference objects. For example, in the case of pivot-based algorithms, we could think that the higher the number of elements chosen as pivots, the more efficient the search. But the query has to be compared both with the pivots and the objects that could not be discarded, so we have to reach a good trade-off between the number of pivots and the amount of objects that can be discarded using them.

Previous works have proposed several heuristics for pivot selection. For example, [11] selects as pivots the objects maximizing the sum of the distances to the already selected pivots. [7] and [14] try to obtain pivots far away from each other. [17] extensively analyzes this problem and shows experimentally the importance of this factor in the search performance. In addition, [17] proposes a criterion to compare the efficiency of two sets of pivots of the same size, and three pivot selection techniques based on that criterion. The first one is called *Selection*, which selects N random sets of pivots and finally uses the one maximizing the efficiency criteria. *Incremental* iteratively selects the set of pivots, adding to the set the object that more contributes to the efficiency criteria when added to the current set of pivots. *Local optimum* starts with a random set of pivots and in each iteration substitutes the pivot that less contributes to the efficiency criterion with other object. Although all these techniques select pivots that improve the search performance, in all of them the number of pivots has to be stated in advance. Therefore, the optimal number of pivots has to be computed by trial and error on a significant data collection (and this inevitably makes the search method static).

3.2 Sparse Spatial Selection (SSS)

Sparse Spatial Selection (SSS) [15] dynamically selects a set of pivots well distributed in the space. The hypothesis behind this selection strategy is that, if the pivots are "spatially" sparse in the metric space, they will be able to discard more objects in the search operation. To do this, when a new object is inserted in the database, it is selected as a new pivot if it is far away enough from the pivots already selected. We consider that the new object is far enough if its distance to any pivot is greater than or equal to $M\alpha$, being M the maximum distance between any two objects from the collection, and α a constant parameter that usually takes values are around 0.4. The parameter α has effect on the number of pivots selected. Figure 2 shows with several vector spaces that the optimal values of this parameter are always in the range from 0.35 to 0.4 (note also that the efficiency of the search is virtually the same for all the values included in this interval).

The results presented in [15] show that this strategy is more efficient than others previously proposed in most cases. Furthermore, SSS has other important features. SSS is dynamic, this is, the database can be initially empty, and the

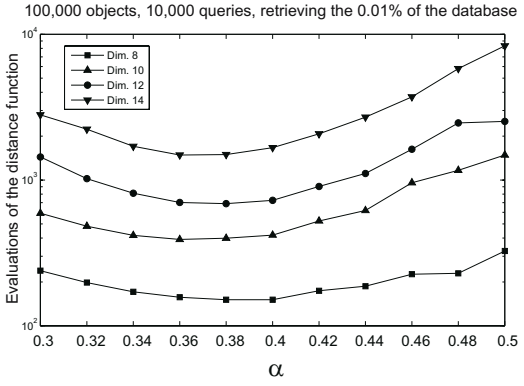


Fig. 2. Search efficiency in terms of the parameter α

pivots will be selected when needed as the database grows. SSS is also adaptive, since it is no necessary to state in advance the number of pivots to select. As the database grows, the algorithm determines if the collection has become complex enough to select more pivots or not. Therefore, SSS adapts the index to the intrinsic dimensionality of the metric space [15].

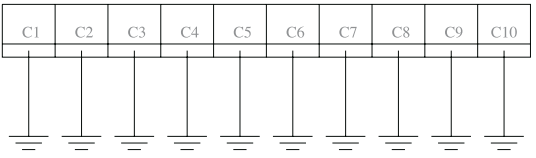
Although SSS was initially designed for pivot selection, in this work it has been used for cluster center selection. Our hypothesis is that if the cluster centers are well distributed in the metric space, the partition of the space will be better and the search operation will be more efficient.

4 SSSTree: Sparse Spatial Selection Tree

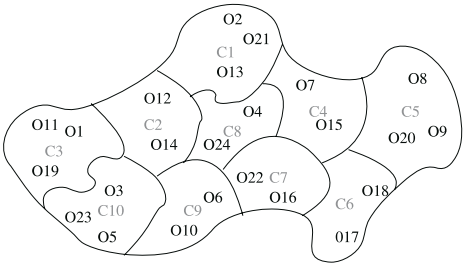
SSSTree is a clustering-based method for similarity search based on a tree structure where the cluster centers of each internal node of the tree are selected applying *Sparse Spatial Selection* (SSS). Our hypothesis is that, using these cluster centers, the partition of the space will better and the performance of the search operation will be increased. In each node, the corresponding subspace is divided in several regions or clusters, and for each of them the covering radius is stored. An important difference with methods like GNAT is that SSSTree is not a balanced tree and not all the nodes have the same number of branches. The tree structure is determined by the internal complexity of the collection of objects.

4.1 Construction

In the construction process we use an additional data structure called *bucket*, which simply contains a subset of the objects of the collection. All the objects of the collection are initially in the bucket. In the root of the tree the cluster centers are selected applying SSS (the maximum distance can be estimated as in [15]). Therefore, the first object of the bucket will be the first cluster center. For the rest of the objects in the bucket, an object is selected as a new cluster



(a) Selected centers



(b) Auxiliar *bucket*

Fig. 3. Situation after the selection of the cluster centers of the root tree

center if it is at a distance greater than $M\alpha$ to the centers already selected. If an object is not a new cluster center, it is inserted in the bucket of its corresponding cluster (the cluster of its nearest center). Figures 3 and 4 show an example of tree construction after the selection of the cluster centers of the root node. The process is recursively repeated in each internal node of the tree (in each node, the objects of its corresponding cluster are stored in a bucket and we can proceed exactly in the same way), until clusters reach a minimum number of elements. The partition applied in each new node depends on the complexity of the cluster associated to it and does not have to use the same number of clusters.

In each internal node we have to estimate again the maximum distance in the cluster associated to it (since the maximum distance of the metric space is not valid for each new cluster). The maximum distance can vary even between different clusters in the same level. In next subsection we explain different ways to efficiently estimate the maximum distance into a cluster. Once estimated, the new cluster centers are selected, and the new subspaces are created. The process stops when a cluster has a number of objects less than or equal to a threshold δ or, alternatively, when the covering radius of the cluster is smaller than a given threshold.

Applying this strategy for the selection of cluster centers, not all the nodes of the tree will have the same number of child nodes. Each cluster is divided in a number of regions which depends on the distribution and complexity of the data of that cluster. This is a very important difference with other structures like GNAT. The index construction adapts the index to the complexity and distribution of the objects of the metric space, and in each level of the tree only those needed clusters will be created. This property is derived from the fact that SSS is a dynamic selection strategy able to adapt the set of reference objects to the complexity of each space or subspace.

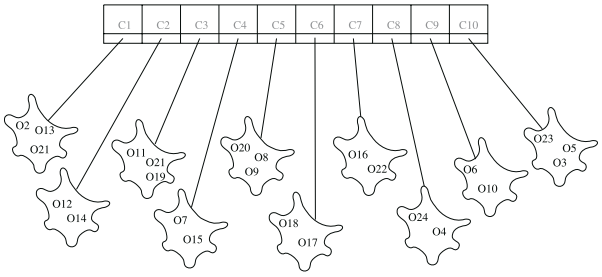


Fig. 4. SSSTree tree after the first space partition

4.2 Estimation of the Maximum Distance M

One of the problems of the construction process is the need of computing the maximum distance M in each cluster. The naive way to compute this distance is to compare each object of the cluster with all the other objects (and this approach is too expensive). Although the index construction is usually an *off-line* process, the value of the maximum distance should be estimated to improve the index construction.

M is the maximum distance between any pair of objects of the cluster, and the covering radius RC is the distance from the cluster center to the furthest object of the cluster. Thus, $M \leq 2 \times RC$. As we can observe in figure 5, if we use $2 \times RC$ as the cluster diameter we can cover the same objects as using M as diameter. Therefore, $2 \times RC$ can be used as a good estimation of the maximum distance during the construction process.

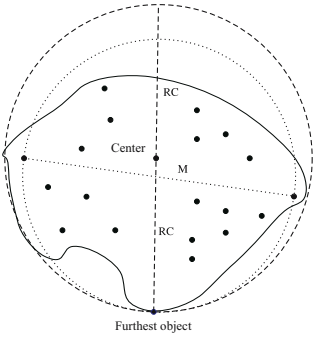


Fig. 5. Maximum distance estimation

4.3 Searching

During the search operation the covering radius of each cluster is used to decide in each step in which clusters we have to continue the search. The covering

radius is the distance from the cluster center c_i, and the object of the cluster furthest to it. Therefore, in each step we can discard the cluster with center c_i if $d(q, c_i) - r > rc(c_i)$, where $rc(c_i)$ is the covering radius of that cluster. When the search reaches the leaves of the tree which could not be discarded, we have to compare the query against all the objects associated to that leaf to obtain the final result of the search.

5 Experimental Results

5.1 Experimental Environment

The performance of SSSTree was tested with several collections of data. First, we used a collection of 100,000 vectors of dimension 10, synthetically generated, and with Gaussian distribution. The Euclidean distance was used as the distance function when working with this collection. In addition to this synthetic vector space, we also worked with real metric spaces. The first one is a collection of 86,061 words taken from the Spanish dictionary, using the edit distance as the distance function. The second one is a collection of 40,700 images taken from the NASA image and video archives. Each image is represented by a feature vector of 20 components obtained from the color histogram of the image. The Euclidean distance was used with this collection to measure the similarity between two images. The algorithm was compared with other well-known clustering-based indexing methods: M-Tree [18], GNAT [14] and EGNAT [16].

5.2 Search Efficiency

The first experiment consisted in the evaluation of the search efficiency obtained with SSSTree compared with other similar methods. Figure 6 shows the results

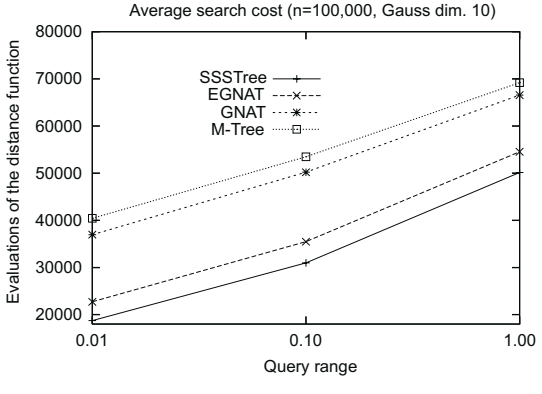


Fig. 6. Evaluations of the distance function with M-Tree, GNAT, EGNAT and SSSTree in a collection of vectors of dimension 10 and Gaussian distribution

obtained with the collection of vectors, expressed as the number of evaluations of the distance function (average of 10,000 queries), in terms of the percentage of the database retrieved in each query (the higher this percentage, the more difficult the search). As we can see in this figure, SSSTree obtains better results than the other methods.

Figures 7 and 8 show that SSSTree is also more efficient in the collection of words taken from the Spanish dictionary and the collection of images from NASA archives, which both are real metric spaces. In the case of the collection of words, the performance of the algorithms was compared for different query ranges, since the higher the range the more difficult the search. These results obtained with both synthetic and real metric spaces show that SSSTree is more efficient than previously proposed methods.

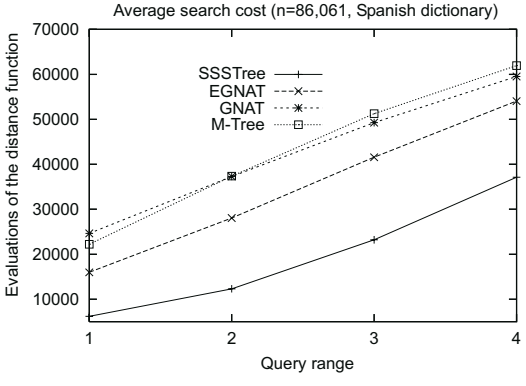


Fig. 7. Evaluations of the distance function with M-Tree, GNAT, EGNAT, and SSSTree, in a collection of words taken from the Spanish dictionary

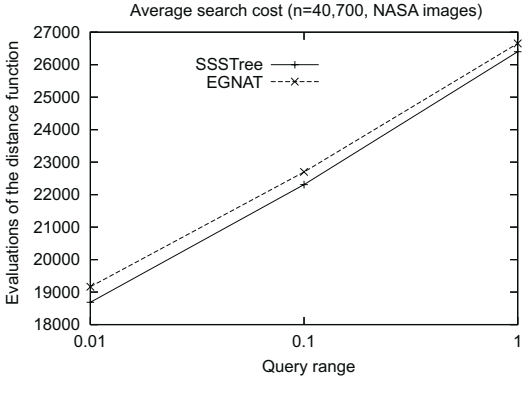


Fig. 8. Evaluations of the distance function with EGNAT and SSSTree, in a collection of images from NASA archives

6 Conclusions and Future Work

In this paper we present a new method for similarity search in metric spaces called SSSTree. Its main characteristic is that the cluster centers are selected applying *Sparse Spatial Selection* (SSS), a selection strategy initially developed for pivot selection. SSS is an adaptive strategy that selects well distributed reference points to improve the search performance. These two properties are also present in SSSTree. The adaptive cluster center selection makes each node of the tree to partition its corresponding subspace as needed in terms of its complexity, a very important difference with previously proposed methods.

The paper also presents experimental results with vector spaces, a collection of words and a collection of images, that show the efficiency of SSSTree against other methods. The obtained improvement is obtained due to the fact that the cluster centers are well distributed in the space and only the clusters needed to cover the complexity of the metric space are created. The space partition is more efficient and this is reflected in the search efficiency.

Our work line still maintains some open questions for future work. First, we are testing the performance of SSSTree with other real metric spaces, as collections of text documents or images. We are also studying the stop condition for the construction process in terms of the covering radius instead of the number of objects contained in the cluster. We are also working in experiments with nested metric spaces [15].

Acknowledgments

This work has been partially supported by: For N. Brisaboa, O. Pedreira and D. Seco by "Ministerio de Educación y Ciencia" (PGE and FEDER) refs. TIN2006-16071-C03-03 and (Programa FPU) AP-2006-03214 (for O. Pedreira), and "Xunta de Galicia" refs. PGIDIT05SIN10502PR and 2006/4. For R. Uribe by Fondecyt 1060776, Conicyt PR-F1-002IC-06, Universidad de Magallanes, Chile y CYTED-GRID Proyecto 505PI0058.

References

1. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Computing Surveys 33, 273–321 (2001)
2. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity search. The metric space approach 32 (2006)
3. Burkhard, W.A., Keller, R.M.: Some approaches to best-match file searching. Communications of the ACM 16, 230–236 (1973)
4. Baeza-Yates, R., Cunto, W., Manber, U., Wu, S.: Proximity matching using fixed-queries trees. In: Crochemore, M., Gusfield, D. (eds.) CPM 1994. LNCS, vol. 807, pp. 198–212. Springer, Heidelberg (1994)
5. Baeza-Yates, R.: Searching: an algorithmic tour. Encyclopedia of Computer Science and Technology 37, 331–359 (1997)

6. Chávez, E., Marroquín, J.L., Navarro, G.: Overcoming the curse of dimensionality. In: CBMI 1999. European Workshop on Content-based Multimedia Indexing, pp. 57–64 (1999)
7. Yianilos, P.: Data structures and algorithms for nearest-neighbor search in general metric space. In: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete Algorithms, pp. 311–321 (1993)
8. Bozkaya, T., Ozsoyoglu, M.: Distance-based indexing for high-dimensional metric spaces. In: SIGMOD 1997. Proceedings of the ACM International Conference on Management of Data, pp. 357–368 (1997)
9. Yianilos, P.: Excluded middle vantage point forests for nearest neighbor search. In: Goodrich, M.T., McGeoch, C.C. (eds.) ALENEX 1999. LNCS, vol. 1619, Springer, Heidelberg (1999)
10. Vidal, E.: An algorithm for finding nearest neighbors in (aproximately) constant average time. Pattern Recognition Letters 4, 145–157 (1986)
11. Micó, L., Oncina, J., Vidal, R.E.: A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear pre-processing time and memory requirements. Pattern Recognition Letters 15, 9–17 (1994)
12. Kalantari, I., McDonald, G.: A data structure and an algorithm for the nearest point problem. IEEE Transactions on Software Engineering 9, 631–634 (1983)
13. Uhlmann, J.K.: Satisfying general proximity/similarity queries with metric trees. Information Processing Letters 40, 175–179 (1991)
14. Brin, S.: Near neighbor search in large metric spaces. In: 21st conference on Very Large Databases (1995)
15. Brisaboa, N.R., Pedreira, O.: Spatial selection of sparse pivots for similarity search in metric spaces. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 434–445. Springer, Heidelberg (2007)
16. Uribe, R., Navarro, G., Barrientos, R.J., Marín, M.: An index data structure for searching in metric space databases. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J. (eds.) ICCS 2006. LNCS, vol. 3991, pp. 611–617. Springer, Heidelberg (2006)
17. Bustos, B., Navarro, G., Chávez, E.: Pivot selection techniques for proximity searching in metric spaces. Pattern Recognition Letters 24(14), 2357–2366 (2003)
18. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: VLDB 1997. Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 426–435 (1997)

A Useful Bounded Resource Functional Language

Michael J. Burrell[1], James H. Andrews[2], and Mark Daley[3]

[1] Department of Computer Science, University of Western Ontario
mburrel@uwo.ca
[2] Department of Computer Science, University of Western Ontario
andrews@csd.uwo.ca
[3] Department of Computer Science and Department of Biology, University of Western Ontario
daley@csd.uwo.ca

Abstract. Real-time software, particularly that used in embedded systems, has unique resource and verification requirements. While embedded software may not have great need for processor and memory resources, the need to prove that computations are performed correctly and within hard time and space constraints is very great. Improvements in hardware and compiler technology mean that functional programming languages are increasingly practical for embedded situations. We present a functional programming language, Ca, built on catamorphisms instead of general recursion, intended for use in static analysis. Ca is not Turing-complete—every program must terminate—but it still provides an excellent framework for building static analysis techniques. Catamorphisms are a general tool which encompass bounded iteration, and allow to traverse any algebraic data structure. We discuss the computational properties of this language, as well as provide a framework for future work in static analysis.

1 Introduction

Software running on embedded systems has requirements unique from that on other systems. While recent work in static analysis has focused on imperative languages, particularly in data flow analysis [5,1], improvements in compiler technology and hardware technology make functional languages increasingly practical. Certain constructs in imperative languages, such as pointer aliasing and global variables, make static analysis difficult, often requiring whole-program analysis. However, referential transparency is inherent in many functional languages and allows one to reason about the semantics of a program piecemeal. This makes a language with referential transparency well-suited for situations where the behaviour of a program needs to be verified.

Software for embedded systems is often written as an event loop, responding to events or to conditions reported by sensors and giving some response in turn. In this way, a component of an embedded system can be viewed as a computation

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 198–210, 2008.

which takes an event as input, and produces a response as output. This computation can be abstracted away from the need to perform side effects.

Section 2 discusses previous work in the area of static analysis aimed at embedded systems, and previous work in functional language research. Section 3 introduces a new programming language designed specifically for suitably in writing embedded software; Section 4 gives the language's operational semantics. Sections 5 and 6 discuss the theoretical computational properties of this programming language, specifically its termination and computational power respectively. Finally, the paper will discuss applying this framework to static analysis and future work.

2 Previous Work

Static analysis is of great potential, and has especial importance in embedded systems. Recent work has focused on imperative languages [6,8,10]. In the case of determining the Worst Case Execution Time (WCET) of a program for example, current techniques typically focus on analysing imperative programs by looking at program control flow and data flow [6,10]. This has the disadvantage that, due to effects like aliasing and loop-variant side-effects, worst case estimates must either be very conservative, or very expensive, involving whole-program analysis, or even so far as abstract interpretation to improve accuracy [8].

We focus on functional languages, which offer the principle of referential transparency. That is, the behaviour of a function depends solely on its arguments, and never on any global state of the program.

3 Programming Language

We present a programming language, Ca, based on catamorphisms which is purely functional and strongly typed, with user-specified types. Ca can be regarded as a restricted form of Haskell, and a programming language in its own right. However, it can also be regarded as a set of restrictions to apply to programs of other functional languages. Ca bears resemblance to Charity [7], another language built on catamorphisms, and other categorical languages built on the work of Hagino [9]. However, Charity is a much more complex language, aimed at researching program transformations, rather than static analysis, and has an anamorphism construct, which increases its computational power [3].

The grammar describing the expressions in Ca is shown in figure 1. The syntax is similar to that of many small purely functional languages. For instance, Ca allows non-recursive function declarations similar to many functional languages.

The most important restriction of Ca is that general recursion is prohibited. While function calls are allowed, a function may only call functions which have been previously declared. Similarly, while `let` expressions are allowed, there is no mechanism to allow `letrec`-like expressiveness.

```
Expression     → 'let' identifier '=' Expression1 '; in' Expression
               | Expression Catamorphism
               | Expression1
Expression1    → Expression1 binaryOperator Expression2
               | Expression2
Expression2    → identifier ArgumentList | Expression3
ArgumentList   → ArgumentList Expression3 | ε
Expression3    → identifier
               | @ identifier
               | integerLiteral
               | '(' Expression ')'
Catamorphism   → '{' PatternList '}'
PatternList    → PatternExpr | PatternExpr PatternList
PatternExpr    → Pattern '→' Expression ';'
Pattern        → identifier | Pattern identifier
```

Fig. 1. The syntax expressions in the Ca programming language

3.1 Catamorphisms

Since recursion is disallowed, a catamorphism construct is added. A catamorphism [12] is an operation that works on a (typically recursive) data type, and allows for recursively traversing the constructors in a data structure and combining the results. When applied to a list, a catamorphism allows a general way of applying an operation to every element in the list; when applied to a natural number, a catamorphism allows a way to do something a bounded number of times. A catamorphism must be applied to some object, such as a list or natural number.

Catamorphisms in Haskell. Readers familiar with functional programming in a function language that allows recursion, such as Haskell, will recognize the right fold function. In Haskell, a right fold function might be expressed as:

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr c n [] = n
foldr c n (x:xs) = c x (foldr c n xs)
```

The right fold is a catamorphism over lists. The Nil constructor is replaced by the object n, and the Cons constructor is replaced by the result of calling the function c. Since the Nil constructor has an arity of 0, the n object also has arity 0. Since the Cons constructor has arity 2 (it has two data: the head of the list, and the tail of the list), the c function also has arity 2. In effect, the `foldr` function allows a very general way to perform an operation (given by the c function) to every element of a list. The return type, b, can be the same list type or any other type, but it must be the same as the type of n and the return type of c.

Such a function can be written for any data type. Consider a hypothetical binary tree data type, defined as: `data Tree a = Node (Tree a) (Tree a) | Leaf a`. The catamorphism function for `Tree` in Haskell would look as follows:

```
sumLeaves t = t {
  Node l r -> @l + @r;
  Leaf x -> x;
}
```

```
cataTree n l (Leaf x) = l x
cataTree n l (Node left right)
  = n (cataTree n l left) (cataTree n l right)
sumLeaves t = cataTree (λ suml sumr ->
  suml + sumr) (λ (Leaf x) -> x) t
```

Fig. 2. The correspondence between the catamorphism notation of CA, left, and the equivalent program in Haskell, right

```
cataTree :: (b -> b -> b) -> (a -> b) -> (Tree a) -> b
cataTree n l (Leaf x) = l x
cataTree n l (Node lt rt) = n (cataTree n l lt) (cataTree n l rt)
```

The catamorphism could, for instance, be used to calculate the sum of all the nodes of a tree of integers as seen in Figure 2.

Catamorphisms in CA. The catamorphism construct in CA is a description of a function that recurses down a data structure. The application of catamorphism `c` to expression `e` is written in postfix, as `e c`. A catamorphism construct associates patterns with corresponding expressions, like a case-based function definition. Within a catamorphism we can use an expression of the form `@v` to signify the application of the catamorphism to `v`, if `v` is a variable from the corresponding pattern which is of the same type as the argument of the catamorphism.

Figure 2 shows how a catamorphism written in CA corresponds to a recursive program in Haskell. In the `Node` case, `l` and `r` (the left and right branches of the node) are of type `Tree`. However, we can use `@l` and `@r`, which stand for the results of the left and right branches, respectively, when the catamorphism is applied to them. Thus, since the catamorphism returns integers, `@l` and `@r` are integers, not trees. `@l` stands for the sum of the left branch, and `@r` stands for the sum of the right branch, and thus the sum of the tree is simply `@l + @r`.

Example 1. The traditional `append` function and the `sumList` function that sums the members of a list can be programmed in CA as follows.

```
append a b = a { Nil -> b; Cons x xs -> Cons x @xs; }
sumList a = a { Nil -> 0; Cons x xs -> x + @xs; }
```

Note that, beyond allowing bounded iteration, catamorphisms also allow a mechanism for case matching. Consider a non-recursive type, such as the booleans, defined as: `data Boolean = True | False`. Performing case matching on a boolean object can be done simply via catamorphism:

Example 2. Using catamorphisms to simulate an if-then-else construct.

```
(x < 4) {
  True -> 4;
  False -> x;
}
```

In general, the expression *if* p *then* e_1 *else* e_2 is represented in Ca as `(p) { True -> e1; False -> e2; }`.

3.2 Tuples

Tuples of any length can be simulated via data types and catamorphisms. For example, consider a 4-tuple. We can define a data type as follows:

```
data Tuple4 a b c d = Tuple4 a b c d
```

We can then represent the expression (x, y, z, w) in Ca as the construction `Tuple4 x y z w`. We can also define projection functions, denoted by the letter π, such that $\pi_i e$ will be the ith element in the tuple e. For example, π_2 in the context of a 4-tuple would be represented as:
π_2 `e = e { Tuple4 x y z w -> y; }`
In the rest of the paper, we assume the existence of the tuple types and projection functions.

3.3 Natural Numbers

From a theoretical basis, natural numbers and integers are often defined recursively. They have a zero constructor and a successor constructor, as we have been using so far. In practical implementations, natural numbers and integers will not be defined recursively, but rather will be implemented as words to be stored in machine registers. Since data types in Ca must be recursively defined, we will treat natural numbers as being so, even if they are not so in practical implementations.

4 Operational Semantics

In this section we use the notation $e[a_1 \mapsto e_1, \ldots, a_n \mapsto e_n]$ to stand for the expression e with all instances of the variables a_i replaced by the expressions e_i. Figure 3 provides the list of operational semantics rules. In all rules, we use C to stand for a constructor. Without loss of generality, we assume all defined data types group recursively-defined subexpressions at the end, e.g. as $C\ x_1 \cdots x_n\ y_1 \cdots y_m$, where each y_i is of the same type as the expression, and each x_i is of a different type.

- Rule 1 explains the semantics of a catamorphism when the expression we are folding over is of a non-recursive constructor (e.g., the Nil constructor of a list). This is where we stop executing the catamorphism.
- Rule 2 explains the semantics of a catamorphism when the expression we are folding over is of a recursive constructor (e.g., the Cons constructor of a list). In this case we recurse on the recursive elements of the constructor (e.g., the "tail" of the list).

$$\frac{\Gamma \vdash e_0 \Rightarrow C\, b_1 \cdots b_n \quad \Gamma \vdash e_1[x_1 \mapsto b_1, \cdots, x_n \mapsto b_n] \Rightarrow e_2}{\Gamma \vdash e_0\{\cdots C\, x_1 \cdots x_n \rightarrow e_1; \cdots\} \Rightarrow e_2} \tag{1}$$

$$\frac{\Gamma \vdash e_0 \Rightarrow C\, b_1 \cdots b_n d_1 \cdots d_m \quad \Gamma \vdash d_1\, cata \Rightarrow d_1' \ \cdots\ \Gamma \vdash d_m\, cata \Rightarrow d_m' \quad \Gamma \vdash e_1' \Rightarrow e_2}{\Gamma \vdash e_0\ cata \Rightarrow e_2} \tag{2}$$

where *cata* contains $(C x_1 \cdots x_n\ y_1 \cdots y_m) \mapsto e_1$,
and $e_1' = e_1[x_1 \mapsto b_1, \ldots, x_n \mapsto b_n, y_1 \mapsto d_1', \ldots, y_m \mapsto d_m']$

$$\frac{\Gamma \vdash e_1 \Rightarrow e_1' \quad \cdots \quad \Gamma \vdash e_n \Rightarrow e_n' \quad \Gamma \vdash d[p_1 \mapsto e_1', \cdots, p_n \mapsto e_n'] \Rightarrow e}{\Gamma \vdash f e_1 \cdots e_n \Rightarrow e} \tag{3}$$

where $(f \mapsto d) \in \Gamma$

$$\frac{\Gamma \vdash e_1 \Rightarrow e_1' \quad \cdots \quad \Gamma \vdash e_n \Rightarrow e_n'}{\Gamma \vdash C\, e_1 \cdots e_n \Rightarrow C\, e_1' \cdots e_n'} \tag{4}$$

$$\frac{}{\Gamma \vdash n \Rightarrow n} \tag{5}$$

where $n \in \mathbb{N}$

$$\frac{\Gamma \vdash e_1 \Rightarrow e_1' \quad \Gamma \vdash e_2 \Rightarrow e_2'}{\Gamma \vdash e_1 \oplus e_2 \Rightarrow e_1' \hat{\oplus} e_2'} \tag{6}$$

where $\oplus$ is a binary operation, and $\hat{\oplus}$ is its semantic equivalent

Fig. 3. The operational semantics of CA

$$\frac{\overline{\Gamma \vdash [\,]\{\texttt{Nil} \rightarrow [9,10]; \cdots\} \Rightarrow [9,10]} \quad \overline{\Gamma \vdash \texttt{Cons}\ 1\ [9,10] \Rightarrow [1,2,3,4,9,10]}}{\Gamma \vdash [1]\{\texttt{Nil} \rightarrow [9,10]; \texttt{Cons}\ x\ xs \rightarrow \texttt{Cons}\ x\ @xs;\} \Rightarrow [1,9,10]} \tag{7}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\Gamma \vdash 0 \Rightarrow 0}}{\Gamma \vdash [\,]\{\texttt{Nil} \rightarrow 0; \cdots\} \Rightarrow 0} \quad \overline{\Gamma \vdash 0 + 10 \Rightarrow 10}}{\Gamma \vdash [10]\{\cdots\} \Rightarrow 10} \quad \overline{\Gamma \vdash 9 + 10 \Rightarrow 19}}{\Gamma \vdash [9,10]\{\cdots\} \Rightarrow 19} \quad \overline{\Gamma \vdash 1 + 19 \Rightarrow 20}}{\Gamma \vdash [1,9,10]\{\texttt{Nil} \rightarrow 0; \texttt{Cons}\ x\ xs \rightarrow x + @xs;\} \Rightarrow 20} \tag{8}$$

$$\dfrac{\dfrac{\overline{\Gamma \vdash [1] \Rightarrow [1]} \quad \overline{\Gamma \vdash [9,10] \Rightarrow [9,10]} \quad \textit{Tree 7}}{\Gamma \vdash \textit{append}\ a\ b \Rightarrow [1,9,10]} \quad \textit{Tree 8}}{\Gamma \vdash \textit{sumList}\ (\textit{append}\ [1]\ [9,10]) \Rightarrow 20} \tag{9}$$

Fig. 4. Tree 9 shows an execution of the program given in figure 1. Γ holds definitions of the functions `append`, `sumList`, `a` and `b`.

– Rule 3 describes the semantics of function calls. Recall that an expression can only call a function, f, if it has previously been defined. Thus, the environment Γ will already have a definition for f.
– Rule 4 describes the semantics of constructions.
– Rule 5 shows that a number evaluates to itself.
– Finally, rule 6 describes the semantics of binary operations. For example, for addition, $\oplus$ would be the syntactic addition operator `+`, and $\hat{\oplus}$ would be the mathematical operation of addition.

Figure 4 shows an example execution of the CA program `sumList (append [1] [9, 10])`, using functions from example 1. For reasons of brevity, we use the notation $[e_1, \ldots, e_n]$ to stand for a Cons-Nil list of elements e_1 through e_n.

5 Termination

We provide a proof that every well-defined CA program terminates. In this context, we say that a program, P, terminates if and only if all of the functions defined in P terminate on all arguments. Other programming languages built upon category theory have guaranteed termination [9].

Notation 1. Where f and g are functions defined in a program, P, we denote $g \lhd f$ to mean f *ultimately calls* g. Thus, $g \lhd f$ if and only if either there is an expression or sub-expression in f which calls g; or there is a function $h \in P$ such that $g \lhd h$ and $h \lhd f$.

Lemma 1. *For any well-defined* CA *program, P, the relation, $\lhd$, provides a strict partial order over the functions defined in P.*

Proof. Recall that f refers to g only if g appears earlier in the program. Thus, f cannot refer to f, and so $\lhd$ is irreflexive. Further, if f can refer to g, then g cannot refer to f, and so $\lhd$ is asymmetric. Transitivity follows from the definition.

The ramification of lemma 1 is that the "call graph" of any well-defined CA program will be a directed acyclic graph. This allows us to provide a measure of where in the call graph a particular function is.

Definition 2. *For a well-defined* CA *program, P, with a finite set of functions, F, we define a function, $f \in F$, to have* ***function number****, denoted $\mathring{f}$, which is defined as:*

$$\mathring{f} = \begin{cases} 1 & , \textit{if } \nexists g \in P \textit{ such that } g \lhd f \\ i+1 & , \textit{otherwise, where } i = \max\limits_{g \in F, g \lhd f} \mathring{g} \end{cases}$$

In other words, if a function, f, is a leaf function, then $\mathring{f} = 1$. If $\mathring{f} = 2$, then it calls only leaf functions, and so on.

Towards the goal of proving termination, we can now introduce the complexity of an expression in a CA program.

Definition 3. *The **complexity of an expression** e, denoted $|e|$, is an ordinal. Let ω be the smallest transfinite ordinal. $|e|$ is defined as follows:*

1. *If e is a constant or identifier, $|e| = 0$;*
2. *If e is a composition of expressions, i.e., a construction, arithmetic operation, or logical operation, with associated sub-expressions $e_1, \ldots, e_n$ for some $n \in \mathbb{N}$, then $|e| = \sum_{i,1 \leq i \leq n} |e_i| + 1$;*
3. *If e is a catamorphism over expression e_c, with patterns $p_1 \to e_1, \ldots, p_n \to e_n$ for some $n \in \mathbb{N}$, then $|e| = \omega(|e_c| + \max_{i,1 \leq i \leq n} |e_i|)$;*
4. *If e is a function call to function f with arguments $e_1, \ldots, e_n$ for some $n \in \mathbb{N}$, then $|e| = \omega^{\omega \mathring{f}} + \sum_{i,1 \leq i \leq n} |e_i|$.*

Lemma 2. *If, in the context of an environment, Γ, for a well-defined expression, e, there exists an expression, e', such that $\Gamma \vdash e \Rightarrow e'$, then $|e'| < \omega$.*

Proof. We can prove by induction that each expression on the right hand side of the $\Rightarrow$ must be an expression built up from integer literals and constructors. These expressions all have a complexity less than ω.

Lemma 3. *If $\Gamma \vdash e \Rightarrow e'$, then $|e'| \leq |e|$.*

Proof. If $|e| \geq \omega$, then by Lemma 2, $|e'| < \omega < |e|$. If $|e| < \omega$, we can prove by induction on $|e|$ that the evaluation process will produce a value of lesser or equal complexity.

Lemma 4. *Let $|e| < \omega^{\omega n}$, and let e' be $e[x \mapsto t]$ for some variable x and expression t such that $|t| < \omega$. Then it is also the case that $|e'| < \omega^{\omega n}$.*

Proof. By induction on the structure of e.

If e is a constant or a variable other than x, then $e' = e$, and so $|e'| = 0$. If e is the identifier x, then $e' = t$, and so $|e'| < \omega < \omega^{\omega n}$.

Otherwise, we assume that the lemma holds for all sub-expressions of e.

If e is a composition of expressions, i.e., a construction or binary operation, then $|e| = \sum_{i,1 \leq i \leq n} |e_i| + 1$, so $|e_i| < \omega^{\omega n}$. By the induction hypothesis, $|e_i'| < \omega^{\omega n}$ for each $|e_i|$, where $e_i' = e_i[x \mapsto t]$. That is, for each $|e_i|$, the biggest term must be less than to $\omega^{\omega(n-1)}k$ for some $k < \omega$. Thus, $|e'| < \omega^{\omega(n-1)}k$.

If e is a catamorphism over expression e_c with patterns $p_1 \to e_1, \ldots, p_n \to e_n$, then $|e| = \omega(|e_c| + \max_{i,1 \leq i \leq n} |e_i|)$. Similarly to the proof for a composition of expressions, $|e'| < \omega(\omega^{\omega(n-1)}k + \omega^{\omega(n-1)}l)$ for some $k, l < \omega$, and so $|e'| < \omega^{\omega n}$.

If e is a function call to function f with arguments $e_1, \ldots, e_n$ for some $n \in \mathbb{N}$, then, because e is well-defined, it must be that $\mathring{f} < n$. $|e| = \omega^{\omega \mathring{f}} + \sum_{i,1 \leq i \leq n} |e_i|$.

Similarly to the proof for a composition of expressions, $\sum_{i,\leq i\leq n} |e'_i| < \omega^{\omega(n-1)}k$ for some $k < \omega$, where $e'_i = e_i[x \mapsto t]$ for each e_i. Thus, $|e'| < \omega^{\omega \mathring{f}} + \omega^{\omega(n-1)}k < \omega^{\omega n}$.

Proposition 1. *For any well-defined expression, e, and environment, Γ, there is an expression, e', such that $\Gamma \vdash e \Rightarrow e'$, by the semantics defined in figure 3.*

Proof. By transfinite induction on the size of e.

If $|e| = 0$, then, by definition, e is a constant, and $e = e'$ by rule 5.

If $|e|$ is a successor ordinal, then it must be that e is a composition of subexpressions, i.e., a construction or binary operation. We know that $|e| = \sum_{i,1\leq i\leq n} |e_i| + 1$. $|e_i| < |e|$ and thus, by the induction hypothesis, there exist expressions e'_i such that $\Gamma \vdash e_i \Rightarrow e'_i$. If e is a construction then, by rule 4, $\Gamma \vdash e \Rightarrow C\, e'_1 \cdots e'_n$. If e is a binary operation, the proof follows by rule 6.

If $|e|$ is a limit ordinal, e must be a catamorphism or function call.

In the case that e is a function call, let e be f applied to arguments $e_1, \ldots, e_n$. $|e| = \omega^{\omega \mathring{f}} + \sum_{i,1\leq i\leq n} |e_i|$. $|e_i| < |e|$, so by the induction hypothesis, there are expressions e'_i such that $\Gamma \vdash e_i \Rightarrow e'_i$. There is a mapping $(f \mapsto e_f) \in \Gamma$. Let $e_{fv} = e_f[x_1 \mapsto e'_1, \ldots, x_n \mapsto e'_n]$ where $x_1, \ldots, x_n$ are the arguments of f. Because e_f can only refer to functions with a function number strictly less than $\mathring{f}$, $|e_f| < \omega^{\omega \mathring{f}}$. Since, by lemma 2, $|e'_i| < \omega$, it must be by lemma 4 that $|e_{fv}| < \omega^{\omega \mathring{f}} < |e|$. By the induction hypothesis, then, $\exists e'_{fv}$ such that $\Gamma \vdash e_{fv} \Rightarrow e'_{fv}$.

We then consider that e is a catamorphism over e_c with pattern list $p_1 \rightarrow e_1, \ldots, p_n \rightarrow e_n$. $|e| = \omega(|e_c| + \max_{i,1\leq i\leq n} |e_i|)$. By the induction hypothesis, there is an expression, $e'_c = C\, e_{c,1} \ldots e_{c,k}$ for some $k \in \mathbb{N}$ such that $\Gamma \vdash e_c \Rightarrow e'_c$. e_c matches some pattern p_j for some $j \in \mathbb{N}$ such that $1 \leq j \leq n$.

If C is a non-recursive constructor, let $e'_j = e_j[x_1 \mapsto e_{c,1}, \ldots, x_k \mapsto e_{c,k}]$ where, without loss of generality, $x_1, \ldots, x_k$ are the variables bound by pattern p_k. By the induction hypothesis, $\exists e'$ such $\Gamma \vdash e'_j \Rightarrow e'$.

If C is a recursive constructor, then let $x_1, \ldots, x_m$ be C's non-recursive attributes—those that do not have the same type as e—and $y_1, \ldots, y_p$ be C's recursive attributes. $\forall i \in \mathbb{N}, p < i \leq k$, let $t_i = e_{c,i}\{p_1 \rightarrow e_1, \ldots, p_n \rightarrow e_n\}$. By lemma 2, $|e_{c,i}|$ is finite and, by extension, $|e_{c,i}| < |e_c|$. Thus, $|t_i| < |e|$. By the induction hypothesis, $\exists t'_i$ such that $\Gamma \vdash t_i \Rightarrow t'_i$. Let $e_{jv} = e_j[x_1 \mapsto e_{c,1}, \ldots, x_m \mapsto e_{c,m}, @y_1 \mapsto t'_1, \ldots, @y_p \mapsto t'_p]$. Each t'_i is finite, and so $|e_j| \leq \omega|e| + b$ for some $b \in \mathbb{N}, b < \omega$, $|e_{jv}| < |e|$. By the induction hypothesis, $\exists e'$ such that $\Gamma \vdash e_{jv} \Rightarrow e'$.

6 Computational Power

As every Ca program must halt, Ca is not a Turing-complete language; there are computable functions that are not computable by Ca. Here we prove that Ca computes the primitive recursive functions. We believe this class of problems, which includes strictly more than the exponential hierarchy, is a reasonable level of expressiveness for use as one cycle of an embedded system event loop.

Notation 4. We use Knuth's "up arrow" notation [11] to express power towers. The algebraic expression m^n, for $m, n \in \mathbb{N}$, can be expressed as $m \uparrow n$. $m \uparrow^2 n$ is equivalent to $m^{m^{\cdot^{\cdot^{\cdot^{m}}}}}$ where the m is raised n times. Stated formally:

$$m \uparrow^k n = \begin{cases} m^n & , \text{ if } k = 1 \\ 1 & , \text{ if } n = 0 \\ m \uparrow^{k-1} (m \uparrow^k (n-1)) & , \text{ otherwise} \end{cases}$$

Lemma 5. *For any $k \in \mathbb{N}$, it is possible to express the binary* Ca *function* `exp_k m n`*, which computes $m \uparrow^k n$.*

Proof. By induction on k.

If $k = 1$, we define: `exp_1 m n = n { Zero -> 1; Succ n -> m * @n; }`.

If $k > 1$, we assume by the induction hypothesis that there exists a function `exp_l m n` which computes $m \uparrow^l n$ where $l = k - 1$. We define:

```
exp_k m n = n { Zero -> 1; Succ n -> exp_l m @n; }
```

This computes `exp_l m (exp_l m (exp_l m ...1))` with n applications of `exp_l`. Thus, it computes $m \uparrow^l (m \uparrow^l (m \uparrow^l \cdots 1))$ for n applications of the $\uparrow^l$ operator. Thus, it computes $m \uparrow^k n$.

Notation 5. For a function $f : \mathbb{N} \to \mathbb{N}$, we define $\mathrm{O}(f) = \{g \mid g : \mathbb{N} \to \mathbb{N}, \exists n_0, c \in \mathbb{N}$ such that $\forall n \in \mathbb{N}$ such that if $n > n_0, g(n) \leq c \cdot f(n)\}$

Proposition 2. *If, for a language $L \subseteq \Sigma^*$, there is a Turing machine, M, such that M decides L and terminates in time $tf(n)$ for $tf \in \mathrm{O}(n \uparrow^k n)$ where $k \in \mathbb{N}$, and input of length n, then there is a* Ca *program which decides L.*

Proof. Since $tf \in \mathrm{O}(n \uparrow^k n)$, there is a Turing machine which decides L in time $c(n \uparrow^k n)$ for some $c \in \mathbb{N}$, as states can always be added after M accepts or rejects to consume time. We assume, without loss of generality, that $tf = c(n \uparrow^k n)$. By lemma 5, there is a Ca function which computes $n \uparrow^k n$. We can multiply any result by a constant, and so there is a Ca function to compute tf.

The proof follows by construction. We assume M is a one-tape, deterministic Turing machine. First, introduce a data type enumerating all states of M and the symbols of the tape of M.

```
data Q = State0 | State1 | State2 | State3 | ...
data Symbol = Blank | S0 | S1 | ...
data Dir = Left | Right
```

Next, write a transition table (the following table would say that, in state 0, given a blank symbol on the tape, transition to state 3, write a 1, and move the head left):

```
next q s = q {
  State0 -> s {
    Blank -> (State3, S1, Left);   -- as an example
```

```
    S0 -> ...
  };
  State1 -> s { ... };
  ...
}
```

Finally, we provide a catamorphism, bounded by the time needed to compute. We assume the existence of the `at` function (to return the ith element of a list), `update` function (to replace the ith element of a list with a given symbol) and the `length` function (to return the length of a list).

```
simulate startState input = (tf (length input)) {
  Zero -> (startState, input, 0);
  Succ prevComp -> let (curState, curTape, curHead) = @prevComp;
      (nextState, sym, dir) = next curState curTape curHead; in
    (nextState, update curTape curHead sym,
      dir { Left -> curHead - 1; Right -> curHead + 1; });
}
```

Consider a word $w \in \Sigma^*$. There is a cognate Ca list of symbols, `input`. We prove, by induction on $tf(n)$ that, if M halts in state q on input w with tape t and head position h, then the `simulate` function above returns the triple `(q, t, h)` when given the appropriate starting state and input as arguments.

If $tf(n) = 0$, M cannot make any moves. Consequently, it halts in the start state, with the original tape, with the head at position 0. The catamorphism given in function `simulate` matches the `Zero` case immediately, and returns a value of `(startState, input, 0)`, where `input` is the original tape.

If $tf(n) > 0$, then after $tf(n) - 1$ steps of execution w, M halts in state q' with tape t' and the head in position h'. Let `input'` be the cognate Ca string corresponding to string w. By the induction hypothesis, the `simulate` function returns `(q', t', h')`. Thus, when `simulate` is run on input `input`, the catamorphism falls through to the `Succ` case. Note `curState` is `q'`, `curTape` is `t'` and `curHead` is `h'`. If M's transition table directs M to write symbol σ, then the `next` function updates `curTape` to write a σ. If M's transition table directs M to move the head left, then 1 is subtracted from `curHead`, and thus `curHead` matches the position of the head in M. Similarly for if M moves the head right. If M's transition table directs M to transition to state q, then the `next` function returns a `nextState` which matches q. Consequently, if M halts in state q with tape t and head in position h, then `simulate` returns `(q, t, h)`.

We write another function which determines which states are accept states, such as:

```
isAccept q = q {
  State0 -> True;   -- as an example
  ...
}
```

And thus, to decide if $w \in L$ for some word w, assuming state 0 is the start state, we write: `decideWord w = isAccept (`π_1 `(simulate State0 w))`, and it follows directly that `decideWord` returns `True` if and only if M accepts w, and `decideWord` returns `False` if and only if M rejects w.

Proposition 2 shows that CA is capable of computing any program where the complexity is an exponential, tetration, etc. function. It is known that this class is equal to that of the primitive-recursive functions [2], referred to as PR.

7 Conclusion and Future Work

We have provided a functional programming language, CA, which does not allow general recursion, but rather allows bounded recursion through data structures via catamorphisms. Computational properties of CA were explored, and it was proven that CA contains the primitive recursive functions. This is powerful enough to capture programs that would be used in embedded systems, while still allowing a framework for performing static analysis.

Future work will focus primarily on developing static analysis techniques based on CA programs: for instance, algorithms to efficiently determine running time of CA programs, number of memory allocations, amount of memory consumed, and cache state. Further, extensions to the language to allow more natural and efficient iteration over multiple data simultaneously, as described in [4], will be explored.

References

1. Ball, T., Rajamani, S.K.: The SLAM project: debugging software via static analysis. In: 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 1–3 (2002)
2. Brainerd, W.S., Landweber, L.H.: Theory of computation. Wiley, New York (1974)
3. Cockett, R.: Charitable thoughts. Lecture notes, University of Calgary (1996)
4. Colson, L.: About primitive recursive algorithms. Theoretical Computer Science 83(1), 57–69 (1991)
5. Dhurjati, D., Das, M., Yang, Y.: Path-sensitive dataflow analysis with iterative refinement. Technical Report MSR-TR-2005-108, Microsoft Corporation (2005)
6. Ermedahl, A., Stappert, F., Engblom, J.: Clustered calculation of worst-case execution times. In: Proceedings of the 2003 International conference on Compilers, architecture, and synthesis for embedded systems, San Jose, California, USA, pp. 51–62 (2003)
7. Fukushima, T., Tuckey, C.: Charity User Manual. University of Calgary (January 1996)
8. Gustafsson, J., Ermedahl, A., Lisper, B.: Towards a flow analysis for embedded system c programs. In: 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pp. 287–300 (2005)
9. Hagino, T.: A Categorical Programming Language. PhD thesis, University of Edinburgh (1987)

10. Steven Li, Y.-T., Malik, S., Wolfe, A.: Efficient microarchitecture modeling and path analysis for real-time software. In: 16th IEEE Real-Time Systems Symposium, p. 298 (1995)
11. MathWorld. Arrow notation (1999), http://mathworld.wolfram.com/ArrowNotation.html
12. Meijer, E., Fokkinga, M.M., Paterson, R.: Functional programming with bananas, lenses, envelopes and barbed wire. In: Hughes, J. (ed.) Functional Programming Languages and Computer Architecture. LNCS, vol. 523, pp. 124–144. Springer, Heidelberg (1991)

On Reachability Games of Ordinal Length*

Julien Cristau[1] and Florian Horn[1,2]

[1] LIAFA, Université Paris 7, Case 7014, 2 place Jussieu, F-75251 Paris 5, France
[2] Lehrstuhl für Informatik VII, RWTH, Ahornstraße 55, 52056 Aachen, Germany
{jcristau,horn}@liafa.jussieu.fr

Abstract. Games are a classical model in the synthesis of controllers in the open setting. In particular, games of infinite length can represent systems which are not expected to reach a correct state, but rather to handle a continuous stream of events. Yet, even longer sequences of events have to be considered when infinite sequences of events can occur in finite time — Zeno behaviours.

In this paper, we extend two-player games to this setting by considering plays of ordinal length. Our two main results are determinacy of reachability games of length less than ω^ω on finite arenas, and the PSPACE-completeness of deciding the winner in such a game.

1 Introduction

Games are a classical model for the synthesis of controllers in open settings, with numerous applications. Although finite games seems more natural, there has been a huge interest for games of infinite duration [GTW02]. They have strong connections with logic (*e.g.* parity games and μ-calculus [EJ91]), and provide useful models in economy. In verification, they are used to represent reactive systems which must handle a continuous stream of events [Tho95]. However, some behaviours cannot be described by this model, when infinite sequences of events happen in finite time. Such behaviours — Zeno behaviours — especially need to be considered in timed systems, when successive events can be arbitrarily close. The classical discrete-time framework used by Alur and Dill in their seminal paper [AD94] prevents such behaviours, while several papers about real-time models limit their results to non-Zeno runs [AM99] or force the players to ensure that they can not happen [dAFH+03]. Since Büchi in the 1960's, several extensions of automata to words of ordinal length have been proposed [BC01, BÉ02]. Demri and Nowak propose in [DN05] an extension of LTL to ordinals of length ω^n. They also formalise a problem of open specification, where only the environment has the opportunity to play more than ω moves, and which was solved in [Cac06].

In this paper, we use the methods of [BC01] in order to define game arenas admitting plays of ordinal length. We show that these reachability games of ordinal length are determined, through a reduction to Muller games. We also

* This paper was supported in part by the French ANR DOTS.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 211–221, 2008.

show that, for several natural ways of representing the transitions, the problem is PSPACE-complete.

Overview of the paper. In Section 2, we recall the definitions of automata on words of ordinal length and games of infinite duration, and we introduce our model of games of ordinal length. Section 3 shows the determinacy of these games on finite arenas, and Section 4 considers the complexity issues. Finally, Section 5 summarises our results, and presents several interesting perspectives for future work about these games.

2 Definitions

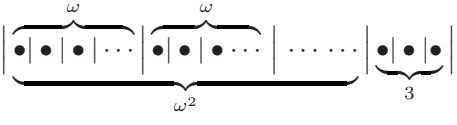
2.1 Ordinals and Automata on Words of Ordinal Length

We consider ordinals, *i.e.* totally ordered sets where any non-empty subset has a least element. In particular, every finite ordered set is an ordinal, as is the set of natural numbers with the usual ordering (usually called ω).

One extends the usual operators $+$ and $\cdot$ to ordinals: $I + J$ is defined by $I \uplus J$ ordered in a way such that $i < j$ if $i \in I$ and $j \in J$; $I \cdot J$ is the set $I \times J$ ordered lexicographically.

A *cut* of an ordinal J is a partition (K, L) of J such that $\forall k \in K, l \in L, k < l$. The set of cuts of J is an ordinal, denoted by $\hat{J}$. For an element j of J, we define the cuts j^- by $(\{i \in J \mid i < j\}, \{i \in J \mid i \geq j\})$ and j^+ by $(\{i \in J \mid i \leq j\}, \{i \in J \mid i > j\})$.

Example 1. $\omega^2 + 3$ is obtained by adding 3 elements to ω^2, which are greater than all others. We represent it below, with bullets for the elements and vertical lines for the cuts:



A word of ordinal length J over an alphabet Σ is a mapping from J to Σ. Let ρ be such a word, and j an element of J. The *prefix* of ρ of length j denoted by $\rho_{<j}$ is defined as $(\rho_i)_{i<j}$. The limit of ρ, denoted $\lim \rho$, is the set: $\{a \in \Sigma \mid \forall j \in J,\ \exists i > j, \rho_i = a\}$.

Bruyère and Carton define in [BC01] an automaton $\mathcal{A}$ on these words as a tuple $(Q, \Sigma, \mathcal{E}, \mathcal{T}, \mathcal{I}, \mathcal{F})$. Q is a finite set of states, Σ is a finite alphabet, $\mathcal{E}$ and $\mathcal{T}$ are respectively the successor and limit transition relations, $\mathcal{I} \subseteq Q$ is the set of initial states, and $\mathcal{F} \subseteq Q$ is the set of final states. The successor transitions of $\mathcal{E}$ are usual transitions, of the form $p \xrightarrow{a} q \in Q \times \Sigma \times Q$. The *limit transitions* of $\mathcal{T}$ are of the form $P \xrightarrow{\lim} q \in \mathcal{P}(Q) \times Q$.

A run of $\mathcal{A}$ on a word $x = (x_j)_{j \in J}$ is a word ρ of length $\hat{J}$ on Q, verifying the following conditions:

- if c is the initial cut, $\rho_c \in \mathcal{I}$;
- if $j \in J$, $\rho_{j^-} \xrightarrow{x_j} \rho_{j^+} \in \mathcal{E}$;

– if c has no predecessor, $\lim \rho_{<c} \xrightarrow{\lim} \rho_c \in \mathcal{T}$;
– if c is the final cut, $\rho_c \in \mathcal{F}$.

Example 2. Figure 1 shows a simple automaton over the alphabet $\{a, b\}$.

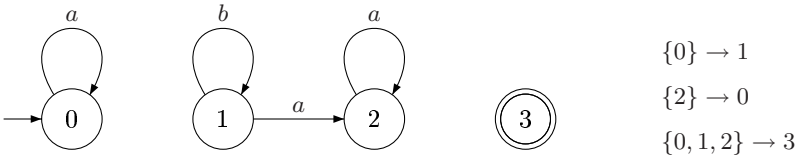


Fig. 1. Automaton recognising $(a^\omega b^\star a^\omega)^\omega$

From the results of Choueka in [Cho78], one can derive Theorem 3:

Theorem 3. *In an automaton with n states where the transitions are of the form $P \xrightarrow{\lim} q \notin P$, there are no runs of length greater than ω^n.*

2.2 Infinite Games

We recall here the usual concepts related to infinite duration games. Such a game $\mathbb{G}$ is played by two players called Eve and Adam on an *arena* of the form $(\mathbb{Q}, \mathbb{E})$, which is a directed graph partitioned between Adam's vertices ($\mathbb{Q}_A$, represented by $\Box$) and Eve's vertices ($\mathbb{Q}_E$, represented by $\bigcirc$). The *winning condition* $\mathbb{W} \subseteq \mathbb{Q}^\omega$ describes the plays won by Eve. We refer the reader to [Tho95] for more details on infinite games.

A *play* of $\mathbb{G}$ is a (finite or infinite) path in the arena. We assume that every state has at least one successor, so any finite play can be prolonged into an infinite one. Prolonging a finite play by one vertex is called a *move* in the game. During the play, when the last vertex of the current prefix is in $\mathbb{Q}_E$, Eve chooses the next move, otherwise Adam does. Eve wins the play if and only if it is in $\mathbb{W}$.

A *strategy* for Eve is a function $\sigma : \mathbb{Q}^*\mathbb{Q}_E \to \mathbb{Q}$ such that for every finite prefix w ending in a state $q \in \mathbb{Q}_E$, $(q, \sigma(w)) \in \mathbb{E}$. A play $\rho = \rho_0\rho_1\rho_2\ldots$ is *consistent* with a strategy σ (for Eve) if for every n such that $\rho_n \in \mathbb{Q}_E$, $\rho_{n+1} = \sigma(\rho_0\rho_1\ldots\rho_n)$. A strategy σ is *winning* for Eve if every play consistent with σ is won by Eve. Strategies and winning strategies for Adam are defined likewise. A *strategy with memory* M for Eve is defined by a transducer (M, ν, μ) and an *initial memory state* $\Omega_0 \in M$. The two functions $\nu : M \times \mathbb{Q} \to \mathbb{Q}$ and $\mu : M \times \mathbb{Q} \to M$ respectively give the next move when the token is in Q_E, and update the memory.

We use Muller games in our proofs. In these games, the winning condition is defined by a subset $\mathbb{M}$ of $\mathcal{P}(\mathbb{Q})$. Eve wins if the set of states occurring infinitely often during the play belongs to $\mathbb{M}$. For a play ρ, this set is denoted by $\mathrm{Inf}(\rho)$ ($\mathrm{Occ}(\rho)$ denotes the set of states occurring in ρ).

When considering complexity issues, the representation of the winning condition is important, as it directly influences the input size. The most straightforward is to list the elements of $\mathbb{M}$ — the *explicit* representation. Other possibilities include colouring, Zielonka trees and DAGs, win-set conditions, and Emerson-Lei conditions. We will only define Emerson-Lei conditions here, and refer to [Zie98] and [HD05] for more details.

Emerson-Lei games were introduced in [EL85] and are equivalent, in terms of expressive power, to the usual Muller games. The winning condition is defined by a Boolean formula φ using elements of $\mathbb{Q}$ as variables. The play ρ is winning for Eve if the truth assignment mapping every state of $\mathrm{Inf}(\rho)$ to `true` and every other state to `false` satisfies φ.

2.3 Games of Ordinal Length

As done in [BC01] for finite automata, we extend the classical model of infinite games to arenas admitting paths of ordinal length by adding limit transitions. A reachability game of ordinal length is defined as a tuple $(Q, Q_E, Q_A, \mathcal{E}, t, \ , \otimes)$. The special states and $\otimes$ are the only two states without successors. The function t maps $\mathcal{P}(Q)$ to $Q \cup \{\ , \otimes\}$ in a way such that $t(P) \notin P$. A play is a word ρ of ordinal length on $Q \cup \{\ , \otimes\}$. Every play that does not end in $\{\ , \otimes\}$ can be prolonged through a move or a limit transition, and by Theorem 3, there are no plays of length greater or equal to ω^ω. Eve wins if the play ends in , while Adam wins when the token reaches $\otimes$. For technical reasons, we suppose without loss of generality that our arenas are *semi-alternating*, *i.e.* that the successors of a state of Adam belong to Eve[1]. The notion of strategy is extended naturally, by considering ordinal prefixes rather than finite ones. Strategies with memory are extended likewise, with a memory transducer on ordinals. Notice that restricting plays to lengths smaller than ω^ω makes sense in the verification problem: infinite sequences represent events of very different durations, and an infinite hierarchy of infinitesimality seems far-fetched.

3 Solving Ordinal Reachability Games

In this section, we consider the problem of deciding the winner in an ordinal reachability game. Our result is formalised as Theorem 4, and this section will mainly be devoted to its proof.

Theorem 4. *Two player reachability games of ordinal length are determined on finite arenas.*

We prove this theorem through a reduction from an ordinal reachability game G to a Muller game $\mathbb{G}$. This construction is described in Section 3.1. Section 3.2 gives the main steps of the proof of Lemmas 5 and 6 by strategy translation.

Lemma 5. *If Eve wins in $\mathbb{G}$ from $q \in Q$, she also wins in G from q.*

[1] This allows us to only define t (and later o) on sets containing a state of Eve.

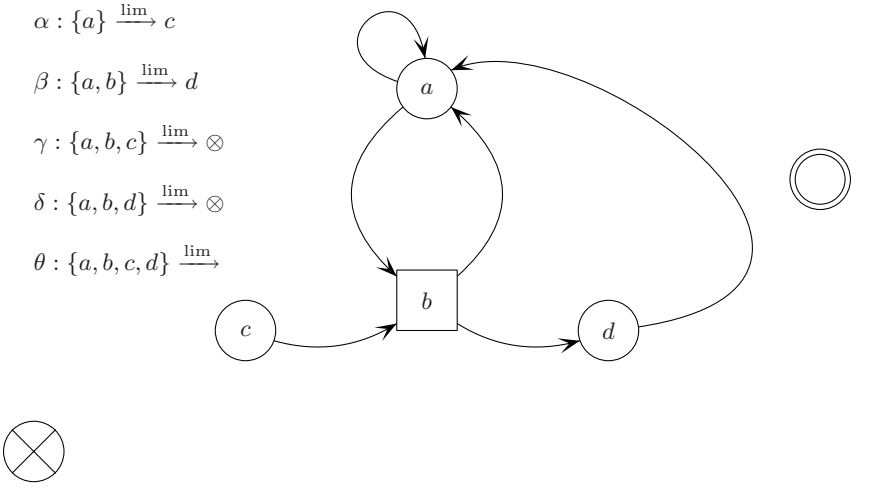


Fig. 2. Game of ordinal length

Lemma 6. *If Adam wins in* $\mathbb{G}$ *from* $q \in Q$*, he also wins in* G *from* q*.*

From these two Lemmas and Theorem 7, we derive Theorem 4.

Theorem 7 ([Mar75]). *Muller games are determined.*

3.1 Reduction to Muller Games

We describe here a reduction from a reachability game of ordinal length G to a Muller game $\mathbb{G}$. The idea is to compel the players to simulate the limit transitions, in such a way that an uncooperative player will lose the play. In this regard, our approach is similar to the one by Chatterjee, de Alfaro, Jurdzinski and Henzinger in [CJH03] and [CdAH05], where they use parity conditions in order to simulate randomness for qualitative winning regions. In both approaches, there is an identity between the winning regions of the original game and the reduced one, but *not* between the actual plays.

The fundamental idea of this reduction is that a word of length less than ω^ω can be described by a finite word with "shortcuts" in lieu of limit transitions. These shortcuts have to be taken in two steps, guaranteeing that both players agree to take it. The "widget" we use is described in Figure 3: for each set of states P containing a state of Eve[2], we distinguish one state $o(P)$ in $P \cap Q_E$. In addition to its original successors, this state now leads to a new state $\chi(P)$, which belongs to Adam. There, he can either accept the transition, and proceed to $t(P)$, or refuse it, and go to another clone of $o(P)$, called $\xi(o(P))$. This clone only leads to the original successors of $o(P)$ in G, not to $\chi(P)$. The definition of the Muller condition guarantees that no one can block the play without losing. It contains all the sets of the form $P \cup \{\chi(P)\}$, so if Adam repeatedly declines to

[2] See Footnote 1 on page 214.

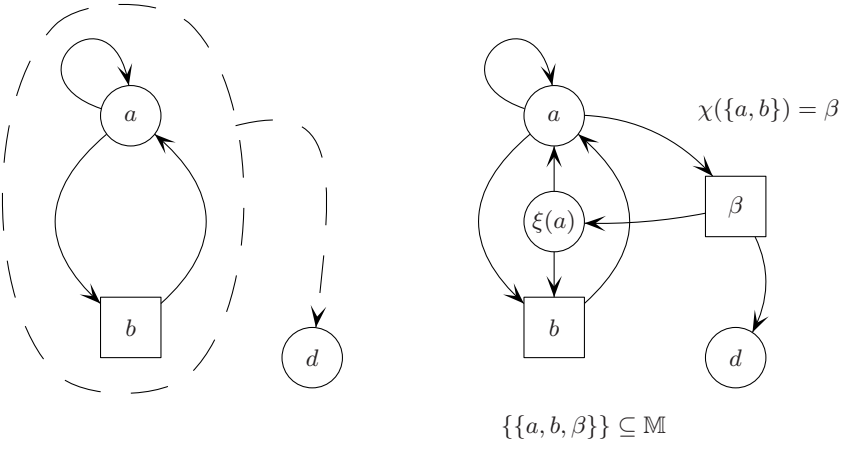


Fig. 3. Widget for $\{a,b\} \xrightarrow{\lim} d$

take a legitimate proposition, he will lose. Formally, the reduced game $\mathbb{G}$ from $G = (Q, Q_E, Q_A, \mathcal{E}, \mathcal{T}, \quad, \otimes)$ is defined by:

$$\begin{aligned}
\mathbb{Q}_E &= Q_E \cup \{\xi(q) \mid q \in Q_E\} \\
\mathbb{Q}_A &= Q_A \cup \{\chi(P) \mid P \in \mathcal{P}(Q)\} \\
\mathbb{E} &= \mathcal{E} \\
&\quad \cup\{(o(P), \chi(P)) \mid P \in \mathcal{P}(Q)\} \\
&\quad \cup\{(\chi(P), t(P)) \mid P \in \mathcal{P}(Q)\} \\
&\quad \cup\{(\chi(P), \xi(o(P))) \mid P \in \mathcal{P}(Q)\} \\
&\quad \cup\{(\xi(p), q) \mid (p,q) \in \mathcal{E}\} \\
&\quad \cup\{(\quad , \quad), (\otimes, \otimes)\} \\
\mathbb{M} &= \{P \cup \mathbb{H} \mid P \subseteq Q, \mathbb{H} \cap Q = \emptyset, \chi(P) \in \mathbb{H}\}
\end{aligned}$$

3.2 Strategy Translation: From Muller to Ordinal Reachability

Lemmas 5 and 6 are proved through a similar notion of *strategy translation*: from a winning strategy[3] in the reduced Muller game $\mathbb{G}$ we can derive a winning strategy in the ordinal game G. The memory states of this new strategy are plays of the Muller game that are consistent with the original strategy.

The memory will evolve during the course of a play in G, moving along the tree of all plays consistent with the strategy. Successor transitions extend the current play, lengthening the memory. Limit transitions will branch to another prefix in the tree, under suitable assumptions. An exemple of this whole process (for both players) is given in Figure 4.

[3] It is not possible to translate a losing strategy with our technique, not even to a losing one.

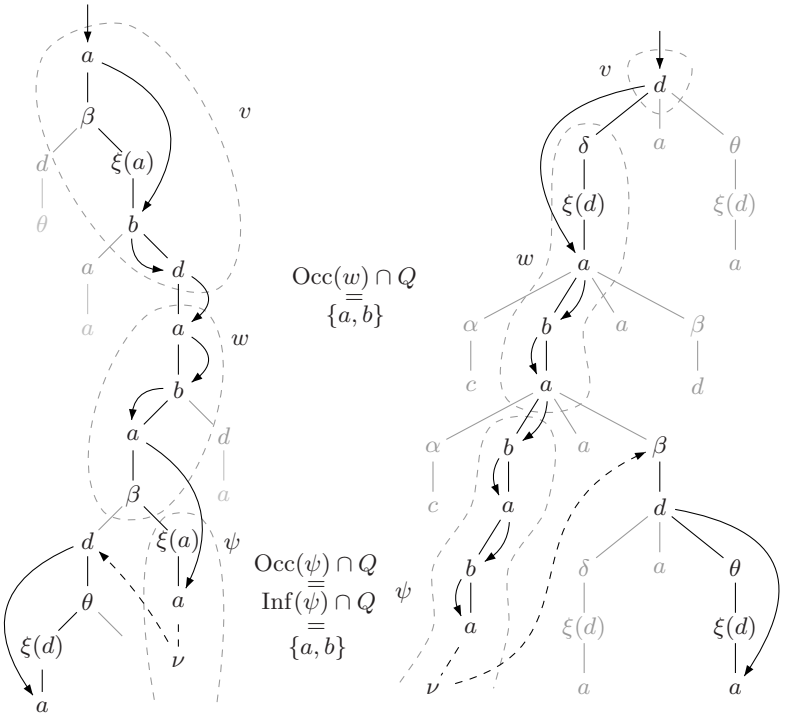


Fig. 4. Strategy translation from $\mathbb{G}$ to G

Successor Transitions. The successor transitions always lengthen the memory, and guarantee that it remains consistent with the original strategy. The basic idea is to copy the current move in the memory. However, we have to be cautious with the states of Eve: she must have the possibility to choose, either as proponent or opponent, to go to a state of the Muller game that does not belong to the original game. This case is treated differently depending whether we consider a strategy for Eve or for Adam.

Eve: For a strategy σ of Eve, the problem arises when, in a state $q \in Q$ with memory $\Omega \in \mathbb{Q}^*$, $\sigma(\Omega \cdot q)$ does not belong to Q, but is a new state $\chi(P)$. To deal with this case, we add three moves to the memory instead of one: $\mu(q, \Omega) = \Omega \cdot q \cdot \chi(P) \cdot \xi(q)$. This is still a finite play consistent with σ, so Eve can now send the token to the location $\sigma(\Omega \cdot q \cdot \chi(P) \cdot \xi(q))$, which is a successor of q in G by definition of $\xi(q)$.

Adam: In Adam's case, the problem is not what the strategy can do — Adam's options in the states of Q are the same in G and in $\mathbb{G}$ — but how to interpret what Eve does. Supposing that she always keeps to states of Q is not correct, since almost any strategy wins against such behaviors. We will thus consider that Eve always chooses to propose transitions when Adam refuses them: if the

token is in state $q \in Q_E$ and Eve sends the token to q', we first look for a set P such that:

- $o(P) = q$ (Eve can propose the transition)
- $\tau(\Omega \cdot q \cdot \chi(P)) = \xi(P)$ (Adam's strategy τ would refuse it)

If we do not find one, the memory is simply updated to $\Omega \cdot q$. Otherwise, we denote by P the one such that $\chi(P)$ did not occur in Ω for the longest time, and update the memory to $\Omega \cdot q \cdot \chi(P) \cdot \xi(P)$. Here also, the resulting memory is still consistent with τ.

Limit Transitions. When the play goes through a limit transition, the idea is to go back to a suitable shortcut in the past of the memory. To explain how we do it, we first fix some notations: we consider a play ρ consistent with our new strategy, and a transition $P \xrightarrow{\lim} q$ occuring at position j — *i.e.* $\lim \rho_{<j} = P$ and $\rho_j = q$. Furthermore, we denote by $(\Omega_i)_{i<j}$ the (transfinite) sequence of memory states occuring in the course of $\rho_{<j}$.

One first problem is that there is no last memory state before j from which to work. Propositions 8 and 9 compensate for this:

Proposition 8. *The sequence $(\Omega_i)_{i<j}$ has a limit, denoted by $\Omega_{<j}$, which is an infinite play in $\mathbb{G}$ consistent with the original winning strategy.*

Proposition 9. $\lim \rho_{<j} = \mathrm{Inf}(\Omega_{<j})$

With these two propositions, we can now update the memory. We have to be careful when choosing where to branch, in order to ensure that the memory still grows with respect to a possible higher order transition. It is done by keeping one copy of the limit set in the resulting memory: We divide $\Omega_{<j}$ in three factors v, w, and ψ:

- v contains all occurrences of states in $Q \setminus \mathrm{Inf}(\Omega_{<j})$;
- w contains an occurrence of each state in $\mathrm{Inf}(\Omega_{<j}) \cap Q$. Furthermore, it must end at a suitable branching point: for Eve, it means ending with an occurence of $\chi(P)$; for Adam, it must end with an occurence of $o(P)$, and be such that $\tau(v \cdot w \cdot \chi(P)) = t(P)$;
- ψ contains the remainder of $\Omega_{<j}$.

The factor $v \cdot w$ remains as a prefix of the new memory. Instead of ψ, there is now an accepted shortcut: $v \cdot w \cdot t(P)$ or $v \cdot w \cdot \chi(P) \cdot t(P)$, depending on whether we are building a strategy for Eve or for Adam. This process is described in Figure 4.

Once the soundness of our construction is accepted, it is not difficult to show that it produces winning strategies: a full play always ends in or in $\otimes$, and the current state is systematically added to the memory. As this memory can only contain plays consistent with the original strategy, the "bad" state ($\otimes$ if we build a strategy for Eve, if it is for Adam) cannot occur in the memory. Thus, the final state of a play consistent with our new strategy is necessarily the "good" one. This completes the proof of Lemmas 5 and 6.

4 Complexity

We now consider the complexity of solving ordinal reachability games. As in Muller games, we need to specify precisely how the transitions are represented. In the case where transitions are represented as relevant sets, colour sets, a Zielonka DAG or Boolean formulae, we get Theorem 10.

Theorem 10. *Deciding the winner in a reachability game of ordinal length whose limit transitions are represented as relevant sets, colour sets, a Zielonka DAG or Boolean formulae is* PSPACE*-complete.*

We will prove the membership part in Section 4.1, and the hardness part in Section 4.2. The complexity in the case of transitions represented explicitly, or as Zielonka Trees is left open.

4.1 Reduction to Emerson-Lei Games

Lemma 11. *Deciding the winner in a reachability game of ordinal length whose transitions are represented as Boolean formulae is* PSPACE*.*

Proposition 12. *The reduced game $\mathbb{G}$ is equivalent to an Emerson-Lei game $\mathbb{L}$ of size polynomial in the size of G, if the transitions of G are represented as Boolean formulae.*

Proof. In order to get a polynomial reduction, we need to avoid the exponential blow-up that occurs when we add a state $\chi(P)$ for each set of states P. It can be done, by noticing that if two sets P and P' are such that $o(P) = o(P')$ and $t(P) = t(P')$, $\chi(P)$ and $\chi(P')$ have exactly the same neighbours in $\mathbb{G}$. In the definition of $\mathbb{L}$, we can thus replace them both by a single state $\kappa(o(P), t(P))$. This limits the number of new states to $|Q|^2 + |Q|$.

The winning condition of $\mathbb{L}$ can now be described in Emerson-Lei formalism:

$$\varphi = \quad \vee \bigvee_{q \in Q \cup \{\circledcirc, \otimes\}} \left(\varphi_q \wedge \bigvee_{p \in Q} \kappa(p, q) \right)$$

The size of formula φ is $O(\sum_{q \in Q \cup \{\circledcirc, \otimes\}} (|\varphi_q| + n))$, which is polynomial in the size of G. This modified reduction is still fair. □

The last step of the proof is Theorem 13:

Theorem 13 ([HD05]). *Deciding the winner in an Emerson-Lei game is* PSPACE*-complete.*

Lemma 11 follows directly from Property 12 and Theorem 13. Corollary 14 follows from the results of [HD05] about succinctness.

Corollary 14. *Deciding the winner in a reachability game of ordinal length whose limit transitions are represented as relevant sets, colour sets, or a Zielonka DAG is* PSPACE*.*

4.2 Hardness Results

The hardness result also derives from Theorem 13. Indeed, any classical Muller game $(\mathbb{Q}, \mathbb{Q}_E, \mathbb{Q}_A, \mathbb{E}, \mathbb{M})$ can be represented as the ordinal reachability game $(\mathbb{Q}, \mathbb{Q}_E, \mathbb{Q}_A, \mathbb{E}, t, \ , \otimes)$, with $t(P) =$ for all $P \in \mathbb{M}$ and $t(P) = \otimes$ for all $P \notin \mathbb{M}$. The strategies and plays will be the same in both games. The only difference is that after ω moves in the reachability game, the token will take a limit transition to or $\otimes$, depending on whether the infinite play is winning for Eve or Adam in the Muller game. This reduction can be done for any representation of the Muller condition.

Lemma 15. *In a reachability game of ordinal length whose limit transitions are represented as relevant sets, colour sets, a Zielonka DAG or Boolean formulae, deciding the winner is* PSPACE*-hard.*

5 Conclusion

We have extended the classical model of infinite games to games of ordinal length. These games, that generalise all regular games, are determined, and the winner is decidable through a reduction to Muller games. If the limit transitions are represented as relevant sets, colour sets, a Zielonka DAG or Boolean formulae, the problem is PSPACE-complete.

We intend now to use this formalism in the context of timed games, following for example the work of [JT07]. Another perspective concerns the minimal quantity of memory that is necessary to define winning strategies in ordinal games. Finally, we would like to consider less general games, where the transitions can be represented in a more compact way — especially parity — and study the effects on complexity and memory.

References

[AD94] Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)

[AM99] Asarin, E., Maler, O.: As soon as possible: Time optimal control for timed automata. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 19–30. Springer, Heidelberg (1999)

[BC01] Bruyère, V., Carton, O.: Automata on linear orderings. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, pp. 236–247. Springer, Heidelberg (2001)

[BÉ02] Bloom, S.L., Ésik, Z.: Some remarks on regular words. Technical Report RS-02-39, 27 (September 2002)

[Cac06] Cachat, T.: Controller synthesis and ordinal automata. In: Graf, S., Zhang, W. (eds.) ATVA 2006. LNCS, vol. 4218, pp. 215–228. Springer, Heidelberg (2006)

[CdAH05] Chatterjee, K., de Alfaro, L., Henzinger, T.A.: The complexity of stochastic Rabin and Streett games. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 878–890. Springer, Heidelberg (2005)

[Cho78] Choueka, Y.: Finite automata, definable sets, and regular expressions over ω^n-tapes. Journal of Computer and System Sciences 17(1), 81–97 (1978)

[CJH03] Chatterjee, K., Jurdzinski, M., Henzinger, T.A.: Simple stochastic parity games. In: Baaz, M., Makowsky, J.A. (eds.) CSL 2003. LNCS, vol. 2803, pp. 100–113. Springer, Heidelberg (2003)

[dAFH+03] de Alfaro, L., Faella, M., Henzinger, T.A., Majumdar, R., Stoelinga, M.: The element of surprise in timed games. In: Amadio, R.M., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, Springer, Heidelberg (2003)

[DN05] Demri, S., Nowak, D.: Reasoning about transfinite sequences. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 248–262. Springer, Heidelberg (2005)

[EJ91] Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy. In: Proceedings of FOCS 1991, pp. 368–377. IEEE, Los Alamitos (1991)

[EL85] Emerson, E.A., Lei, C.-L.: Modalities for model checking: Branching time strikes back. In: Proceedings of POPL 1985, pp. 84–96 (1985)

[GTW02] Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games. LNCS, vol. 2500. Springer, Heidelberg (2002)

[HD05] Hunter, P., Dawar, A.: Complexity bounds for regular games. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 495–506. Springer, Heidelberg (2005)

[JT07] Jurdziński, M., Trivedi, A.: Reachability-time games on timed automata. In: ICALP 2007. LNCS, vol. 4596, pp. 838–849. Springer, Heidelberg (2007)

[Mar75] Martin, D.A.: Borel determinacy. Annals of Mathematics 102, 363–371 (1975)

[Tho95] Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 1–13. Springer, Heidelberg (1995)

[Zie98] Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theorical Computer Science 200(1-2), 135–183 (1998)

An Algorithm for Computation of the Scene Geometry by the Log-Polar Area Matching Around Salient Points

Bogusław Cyganek

AGH - University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
cyganek@uci.agh.edu.pl

Abstract. Computation of a scene geometry belongs to the one of the fundamental problems of computer vision. It can be computed from point correspondences found in a pair of stereo images or a video sequence. This is achieved by an image matching algorithm. However, found correspondences usually are burdened with large errors due to noise and outliers. In this paper we propose an improvement to the point matching algorithm which is twofold. At first the salient points are found which are corners detected by the structural tensor. Then the log-polar representations are computed around found salient points and the matching is done in the extended log-polar space. Such representation has very desirable properties which allow detection of a local change of scale and rotation of the matched areas. This feature is employed in the matching algorithm to eliminate outliers. The proposed method can be used in variety of computer vision tasks, such as stereovision, recovering shape and motion from video, or camera calibration and autocalibration.

1 Introduction

Efficient point matching among multiple views of a scene is a key technique for majority of computer vision methods such as stereovision, tracking, SLAM, camera calibration, to name a few. The problem is difficult and ill posed due to physical characteristic of a camera − a 3D point is mapped onto the camera plane which is "only" two-dimensional. This way all points on a line of sight are mapped into a single point. This inevitably leads to loss of information. Therefore to recover the 3D geometry of a scene a second (or more) camera(s), as well as a point matching methods, are necessary. However, finding corresponding points, i.e. image matching, is not an easy task due to signal quantization, noise and distortions, occlusions, etc. In this paper we propose a new approach to salient point detection and their matching. Detection is based on the structural tensor which allows fast and accurate computation of corners in areas of the strongest signal response. The latter is measured by the lowest eigenvalue of a local structure. Then area matching around the found points is performed. However, instead of the intensity signal its log-polar (LP) representation is used

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 222–233, 2008.

which was shown to be more discriminative [12]. Then, an analysis of the local changes in rotation and scale of the matched regions allows elimination of the outliers. Such strategy improves overall accuracy of the method.

2 Computation of the Scene Geometry from Multiple Images

The corresponding image points **a** and **b** in Fig. 1 are related by the following formula [3,6]:

$$\mathbf{a^T F b = 0} \tag{1}$$

where **F** is a fundamental matrix which determines the epipolar geometry of a camera setup, $\mathbf{a}=[a_1, a_2, a_3]^T$ and $\mathbf{b}=[b_1, b_2, b_3]^T$ are left and right image points expressed in the homogeneous coordinate systems associated with their image planes. It is assumed that the cameras can be approximated by the pinhole model. Once the matrix **F** is known, the epipolar lines, $\mathbf{l}_a$ for the left and $\mathbf{l}_b$ for the right camera, can be determined as follows:

$$\mathbf{l_a = Fb}, \qquad \mathbf{l_b = F^T a}. \tag{2}$$

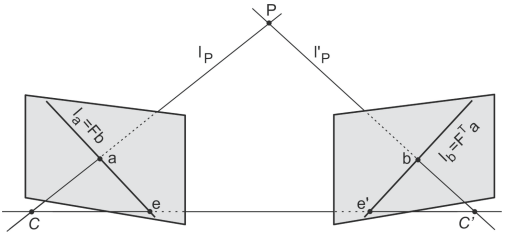


Fig. 1. Epipolar geometry of the two cameras (stereo) system

In a case of a stereo setup this limits the search to one dimension, i.e. along the epipolar lines. Taking into account homogeneous representation of the points, the formula (1), can be expressed in the following form:

$$\mathbf{q^T f} = r = \sum_{i=1}^{9} q_i f_i = 0, \tag{3}$$

where r is called a residual, whereas vectors **q** and **f** are given as follows

$$\mathbf{q} = [a_1 b_1, a_2 b_1, b_1, a_1 b_2, a_2 b_2, b_2, a_1, a_2, 1]^{\mathbf{T}} \tag{4}$$

$$\mathbf{f} = [F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}]^{\mathbf{T}}.$$

Each pair of the corresponding points gives one equation of the type (3). Then, K≥8 pairs of the corresponding points are gathered into a compound matrix $\mathbf{Q}_{K\times 9}$. From **Q** the so called moment matrix $\mathbf{M}=\mathbf{Q}^T\mathbf{Q}$ is created which is of

size 9x9 [10]. The matrix $\mathbf{F}$ is found as an eigenvector $\mathbf{w}$ of $\mathbf{M}$ which corresponds to the lowest eigenvalue of $\mathbf{M}$. This way found $\mathbf{F}$ minimizes the sum of squares of the algebraic residuals $E = \sum_{k=1}^{K} \rho_k$. Therefore, computation of the matrix $\mathbf{F}$ can be stated as the following optimization problem

$$\min \{E\}, \tag{5}$$

where the functional E is given as follows [10]:

$$E = \sum_{k=1}^{K} \rho_k = \sum_{k=1}^{K} \frac{r_k^2}{\mathbf{f}^T \mathbf{J} \mathbf{f}} = \sum_{k=1}^{K} \frac{\mathbf{a}_k^T \mathbf{F} \mathbf{b}_k}{\mathbf{f}^T \mathbf{J} \mathbf{f}} = \frac{\mathbf{f}^T \mathbf{M} \mathbf{f}}{\mathbf{f}^T \mathbf{J} \mathbf{f}}, \tag{6}$$

where $\mathbf{J}$=$\mathbf{J}_1$=*diag[1,1,. . . ,1]* is the normalization matrix which corresponds to the optimization constraint in the form: $||f|| = \sum_i f_i^2 = 1$. The denominator in the above represents an optimization constraint which allows a solution from an equivalence class, excluding the trivial zero results at the same time. Solution to (5) and (6) is obtained as an eigenvector $\mathbf{f}_s$=$\mathbf{w}$ that corresponds to the lowest eigenvalue λ_k of the moment matrix $\mathbf{M}$. To impose the rank two of the computed matrix $\mathbf{F}$ we set its smallest singular value to 0 and then recalculate the fundamental matrix. The way to estimate the given point configuration is just to measure how close to 0 is the smallest singular value of $\mathbf{M}$. Hence optimizing for the smallest singular value leads to a measure of a quality of the point matching.

However, instead of $\mathbf{J}$=$\mathbf{J}_1$ Torr and Fitzgibbon proposed to apply a constraint which is invariant to the Euclidean transformations in the image planes [10]. They showed that the Frobenius norm of the form $f_1^2 + f_2^2 + f_4^2 + f_5^2$ fulfils such invariance requirement. This corresponds to $\mathbf{J}$=$\mathbf{J}_2$=*diag[1,1,0,1,1,0,0,0,0]*. Finding $\mathbf{f}_s$ in this case is more complicated since it is equivalent to solving the generalized eigenvector problem: $\mathbf{f}^T\mathbf{J}\mathbf{f} - \mathbf{f}^T\mathbf{M}\mathbf{f} = 0$. However, the faster and more stable solution can be obtained by the procedure originally proposed by Bookstein and also cited by Torr & Fitzgibbon in [10]. The methodology consists in partitioning $\mathbf{f}$ into $\mathbf{f}_1 = [f_1, f_2, f_4, f_5]$ and $\mathbf{f}_2 = [f_3, f_6, f_7, f_8, f_9]$. Then $\mathbf{f}_1$ is obtained as an eigenvector solution to the equation

$$\mathbf{D}\mathbf{f}_1 = \lambda \mathbf{f}_1, \tag{7}$$

where

$$\mathbf{D} = \mathbf{M}_{11} - \mathbf{M}_{12}\mathbf{M}_{22}^{-1}\mathbf{M}_{12}^T, \qquad \textit{and} \qquad \mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{12}^T & \mathbf{M}_{22} \end{bmatrix}. \tag{8}$$

$\mathbf{M}$ is divided into $\mathbf{M}_{ij}$ in such a way that

$$\mathbf{f}^T\mathbf{M}\mathbf{f} = \mathbf{f}_1^T\mathbf{M}_{11}\mathbf{f}_1 + 2\mathbf{f}_1^T\mathbf{M}_{12}\mathbf{f}_2 + \mathbf{f}_2^T\mathbf{M}_{22}\mathbf{f}_2. \tag{9}$$

Then $\mathbf{f}_2$ is found from $\mathbf{M}_{12}$, $\mathbf{M}_{22}$, and $\mathbf{f}_1$, as follows:

$$\mathbf{f}_2 = -\mathbf{M}_{22}^{-1}\mathbf{M}_{12}^T\mathbf{f}_1. \tag{10}$$

Point coordinates in the equations (1)-(10) are integer values measured in discrete pixel coordinates. Thus the individual entries in these equations can vary by two or three orders of magnitude. This can cause excessive errors when computing $\mathbf{F}$ due to finite precision of a computer representation of data. Hartley showed that a special normalization procedure can increase stability of these computations [5]. The normalization is done by an affine transformation $\mathbf{T}$, consisting of a translation and scaling, so that the centroid of the reference points is at the origin of the coordinate space and the root-mean-square distance of the points from the origin is $\sqrt{2}$. This can be written as follows:

$$\mathbf{a}' = \mathbf{T_a a} = \begin{bmatrix} s_a & 0 & -m_{a1}s_a \\ 0 & s_a & -m_{a2}s_a \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ 1 \end{bmatrix}, \tag{11}$$

where $\mathbf{m}_a = [m_{a1}, m_{a2}, 1]$ is a mean point, $s_a = \sqrt{2}/d_{av}$ and d_{av} is an average point distance from the origin point *[0, 0, 1]*.

Now, in the light of (11) the equation (1) takes the following form:

$$\mathbf{a^T F b} = \left(\mathbf{T_a^{-1} a'}\right)^{\mathbf{T}} \mathbf{F} \left(\mathbf{T_b^{-1} b'}\right) = \mathbf{a'^T T_a^{-T} F T_b^{-1} b'} = 0. \tag{12}$$

Thus, computing in the domain of transformed coordinates we actually obtain:

$$\mathbf{F}' = \mathbf{T_a^{-T} F T_b^{-1}}. \tag{13}$$

Therefore at the end it is necessary to recover $\mathbf{F}$, which can be done as follows:

$$\mathbf{F} = \mathbf{T_a^T F}' \mathbf{T_b}. \tag{14}$$

The normalization is done separately for each image of the stereo-pair. The denormalization (14) is done once. Finally, estimation of the trifocal tensor is very similar to the outlined computation of the fundamental matrix [6][3].

3 Salient Points from the Structural Tensor

The idea of an upright image analysis consists of dividing an image into local neighbourhoods around each pixel and then determining local phase, magnitude and coherence of each of the neighbourhoods. Subsequent image analysis is based on the computed local parameters. We are trying to describe each local neighbourhood(LN) Ω of pixels with a single orientation, denoted by a vector $\mathbf{w}$ in Fig. 2. This could be possible if LN shows some regularity of its intensity signal. This can be measured by gradient vectors in each point of the LN. If these are approximately oriented in the direction of $\mathbf{w}$, then we can say that $\mathbf{w}$ represents well a given LN. For this purpose we need to introduce a measure of a 'goodness' of such a fit. It can be defined as an average of the squared modules of vectors $\mathbf{s}$ which are perpendicular projections of the gradients $\mathbf{g}_i$ onto the sought orientation vector $\mathbf{w}$. From Fig. 2 we see that

$$\mathbf{s} = \mathbf{g} - \mathbf{r} = \mathbf{g} - \frac{\mathbf{w}}{\|\mathbf{w}\|} \|\mathbf{r}\| = \mathbf{g} - \frac{\mathbf{w}\mathbf{g}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}}. \tag{15}$$

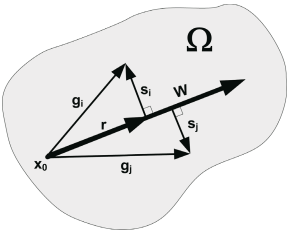


Fig. 2. A local neighbourhood Ω is represented with a single orientation vector $\mathbf{w}$, which is accurate if most of the gradients $\mathbf{g}_i$ in Ω coincide with $\mathbf{w}$. This is equivalent to minimization of a cumulative sum of the residual vectors $\mathbf{s}_i$ in Ω which is also equivalent to finding eigenvectors of the structural tensor.

Assuming that

$$\mathbf{w}^T\mathbf{w} = 1, \tag{16}$$

the error function can be now defined as follows [1]:

$$e(\mathbf{x}, \mathbf{x}_0) = \|\mathbf{s}\| = \left\|\mathbf{g}(\mathbf{x}) - \mathbf{w}(\mathbf{x}_0)\,\mathbf{g}^T(\mathbf{x})\,\mathbf{w}(\mathbf{x}_0)\right\|. \tag{17}$$

The total error E is obtained by integrating the square of $e(\boldsymbol{x},\boldsymbol{x}_0)$ over all possible locations $\mathbf{x}$ in the neighbourhood Ω, using a Gaussian soft averaging filter G_σ:

$$E(\mathbf{x}_0) = \int_{\Omega} e^2(\mathbf{x}, \mathbf{x}_0) G_\sigma(\mathbf{x}, \mathbf{x}_0) d\mathbf{x}. \tag{18}$$

In robust approach, the G_σ is substituted with *a robust function* [2] which makes filtering *dependant* on a local structure. This makes the orientation vector $\mathbf{w}$ well adapted. However, this implies also an iterative solution which sometimes is too slow. Inserting (17) to (18) and expanding we obtain

$$E = -\int_{\Omega} \mathbf{g}^T\mathbf{g} G_\sigma(\mathbf{x}, \mathbf{x}_0)\, d\mathbf{x} + \mathbf{w}^T \left(\int_{\Omega} \mathbf{g}^T\mathbf{g} G_\sigma(\mathbf{x}, \mathbf{x}_0)\, d\mathbf{x} \right) \mathbf{w}. \tag{19}$$

Now, to find $\mathbf{w}$ we have to solve the following optimization problem

$$\min_{\mathbf{w}} \|E\|, \tag{20}$$

subject to the constraint (16). Substituting (19) into (20) we obtain

$$\min_{\mathbf{w}} \|E\| = \min_{\mathbf{w}} \left\| \int_{\Omega} \mathbf{g}^T\mathbf{g} G_\sigma(\mathbf{x}, \mathbf{x}_0)\, d\mathbf{x} - \mathbf{w}^T \left(\int_{\Omega} \mathbf{g}^T\mathbf{g} G_\sigma(\mathbf{x}, \mathbf{x}_0)\, d\mathbf{x} \right) \mathbf{w} \right\|. \tag{21}$$

Since it holds that $\mathbf{g}^T\mathbf{g} \geq 0$, then solving (21) is equivalent to the following minimization problem

$$\min_{\mathbf{w}} \|E\| = \max_{\mathbf{w}} \left\{ \mathbf{w}^T \left(\int_{\Omega} \mathbf{g}^T\mathbf{g} G_\sigma(\mathbf{x}, \mathbf{x}_0)\, d\mathbf{x} \right) \mathbf{w} \right\} \Bigg|_{\mathbf{w}^T\mathbf{w}=1}, \tag{22}$$

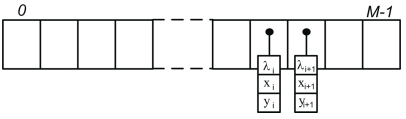


Fig. 3. A priority queue for selection of image points with strongest responses

where the expression

$$\mathbf{T} = \int_{\Omega} \mathbf{g}^T \mathbf{g} G_\sigma \left(\mathbf{x}, \mathbf{x}_0\right) d\mathbf{x}, \tag{23}$$

is called *a structural tensor* (ST). Using the method of Lagrange multipliers we find that the sought orientation $\mathbf{w}$ is an eigenvector of $\mathbf{T}$ corresponding to its largest eigenvalue. This way we obtain $\mathbf{w}$, as follows [1]:

$$\mathbf{w} = \left[T_{11} - T_{22} \; 2T_{12} \right]^T. \tag{24}$$

where T_{ij} are components of $\mathbf{T}$.

It appears that an analysis of the eigenvalues of $\mathbf{T}$

$$\lambda_{1,2} = \frac{1}{2} \left[(T_{11} + T_{22}) \pm \sqrt{(T_{11} - T_{22})^2 + 4T_{12}^2} \right], \tag{25}$$

can provide information on a type of the local structures in an image [1]. By this method, the corner points (x_i, y_i) which we use for matching, can be described as such points which fulfil the following condition:

$$\lambda_1(x_i, y_i) \geq \lambda_2(x_i, y_i) \geq \mu \tag{26}$$

where μ is a threshold for the lowest eigenvalue. The advantage of this approach is that it leads to a natural ordering of the detected points based on their strength, i.e. value of the lowest eigenvalue λ_2 in (25). Moreover we can avoid strict definition of a threshold using a priority queue depicted in Fig. 3. It allows also a control over the mutual distances among the found points which results in a more uniform distribution of the salient points. For an even more uniform spread of the salient points, corners are searched for in a separate tiles of the image. In practice it is sufficient to divide an image into for example 16×16 tiles.

4 Image Matching in the Log-Polar Domain

The log-polar transformation takes points (x, y) from the Euclidean space into the (r,φ) points in the polar space. This process is defined as follows [11]:

$$r = \log_B \left(\sqrt{(x - x_0)^2 + (y - y_0)^2} \right), \quad \varphi = \arctan \tfrac{y - y_0}{x - x_0}, \; for \; x \neq x_0 \tag{27}$$

for a point (x, y), where $O = (x_0, y_0)$ is a centre of transformation, $B > 1$ denotes base of a logarithm. In practice it is chosen to encompass the maximal area determined by a distance r_{max} from the centre O.

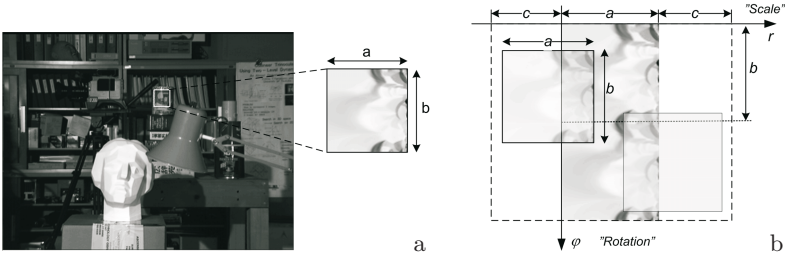


Fig. 4. A matching area from the Tsukuba image and a view of its LP transformation (a). Matching a pattern in the extended LP space: For each position search is two-dimensional to account for the scale and rotation - this results in a 4D search (b).

In the proposed method matching in the LP space follows the technique presented by Zokai & Wolberg [11]. The search is four dimensional and starts with building a reference area which is transformed into the LP space. Then, for each position (x_i, y_i) in the second image(s) a region of the exact size is selected and also transformed into the LP representation. Fig. 4a depicts a left image from the "Tsukuba" stereo pair with a selected region of size 30×30 pixels and its LP transformed version. This region of size $a \times b$ pixels is matched with the right image of "Tsukuba". Contrary to the simple intensity matching, the matching of the LP transformed signals is done in an extended space depicted in Fig. 4b. A test pattern is wrapped around to $2b$ to allow for inherent rotations. The scale range is extended by a distance c, which can reach up to width a of a template. Each position in the extended space is then matched pixel-by-pixel with help of the cross correlation ρ (covariance-variance) measure, given as follows [8]:

$$\rho = \frac{\sum\limits_{(i,j)\in U} \left(I_1(i,j) - \overline{I_1(U)} \right) \cdot \left(I_2(i,j) - \overline{I_2(U)} \right)}{\sqrt{\sum\limits_{(i,j)\in U} \left(I_1(i,j) - \overline{I_1(U)} \right)^2 \cdot \sum\limits_{(i,j)\in U} \left(I_2(i,j) - \overline{I_2(U)} \right)^2}}, \tag{28}$$

where $\overline{I_k(U)}$ denotes an average intensity in a pixel region U in a *k-th* image. This way found position of the best match in the extended space contains an additional information on local scale and rotation *(r,φ)* between input and the template. Thus, for each checked position in the input image, the four parameters (x_i, y_i, r, φ) of the best matches are stored. Selection of a best match can be done at least in two different ways. One is to set a fixed threshold value and accept only correlation measures above this threshold. The second is to build again a priority queue of a fixed length N, which stores N best matches.

The ability of the LP matching to detect local rotation and scale can be used to *sieve out* the outliers. Our idea here is to reject all the matching pairs for which their local rotation or scale deviates significantly from 0. Such a strategy comes from an observed phenomenon that large variations of these parameters in stereo images or consecutive frames of a video stream are highly unusual.

This process is controlled by setting specific threshold values for these parameters. However, to alleviate the problem of finding a suitable threshold values we independently sort the found matches by r and φ. The justification of this procedure is an assumption of local consistency of the matched images. It means that the corresponding points and their neighbouring areas exhibit topological and geometrical similarity. In other words, except for a limited number of image points such as occlusions or noise, it is assumed to be very unusual to have local matching areas which differ significantly in rotation or scale. This is safe procedure since due to over-determined set of equations even if we reject some good matches from further considerations, the left ones are sufficient for computation of the scene geometry. Our assumptions were verified experimentally with help of many real images. The third parameter which can indicate outliers is a value ρ (28) of each match. However, a simple thresholding won't work here. A possible solution is to sort the points in the order of ρ and then disregard a number of points with the lowest scores.

Yet we can do one step more to limit influence of the outliers. After the first sieve it is still possible that the set of matched points will contain outliers. Such situation is more probable when using small regions for the LP matching. For this reason for the computation of the scene parameters we use the robust estimation. The most common is the RANSAC algorithm proposed by Fischler and Bolles [4]. It is also used in our method. It consists in random choice of samples from the set of all measurements, which are then used to the computation of a tentative model. Then a number of other points that are in consensus with this model estimate is checked. The process is repeated and the best fit, i.e. an estimate supported by the maximal number of measurements is left as a solution. All other points are treated as outliers.

The last parameter that needs to be controlled is setting of the search range in the second image. In the simplest approach it can be the whole image. However, this means an exhaustive 2D×2D search which is time consuming and is more probable to produce outliers. Thus, additional information on a matched scene can help to limit the potential search space. For example, in a tracking problem the correspondences are assumed to lie in a limited distance from their reference points in the previous image, although they can be moved in any direction. This is also the case of stereo matching where disparities follow camera setup, e.g. in a horizontal alignment of cameras we expect to have horizontal disparities rather than vertical ones, etc.

5 Experimental Results

The code was written in C++ on the Microsoft® .NET 2005 IDE. The tests were performed on the laptop computer with the Intel® Core Duo processor 2MB L2 cache (T2600 with 2.16GHz speed) and the 2GB of RAM. Architecture of the software for image matching is depicted in Fig. 5. Our methodology of testing the system consists mostly in measuring the following parameters: matching accuracy, speed, and robustness of the method to noise and distortions. Fig. 6

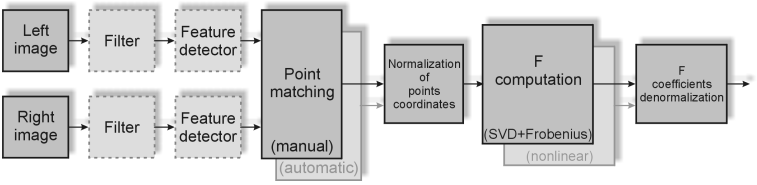


Fig. 5. Architecture of the software for computation of the scene geometry

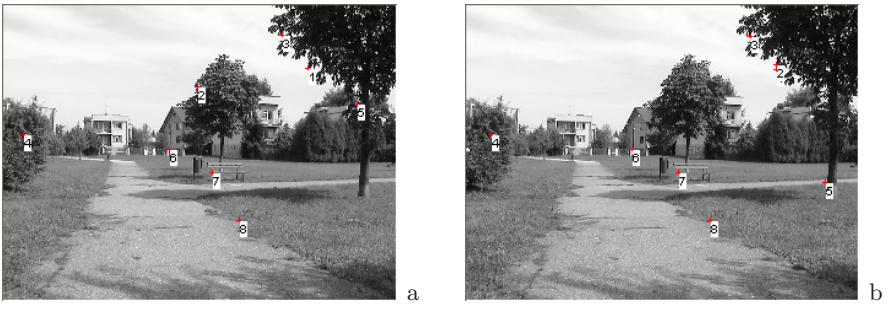


Fig. 6. Matching of the exemplary video sequence. First frame (a) with salient points obtained by the structural tensor detector. The next (5th) frame in a sequence with the matched points (b). Outliers are the pairs of points no. 2 and 5.

depicts two frames from an exemplary video sequence. The corner points with the strongest signal response were detected with the ST. Then the points were matched in the LP space. Each matching area was of the size 17×17. An average execution time is 3.35 [s]. Table 1 contains coordinates of the matched points as well as their parameters: ρ, as well as the local rotation and scale. It is visible that the points which have their large local rotation and/or scale are candidates for outliers which indeed is the case for this example (see Fig. 6). Fig. 7 presents matching of the "Car" stereo pair. Left image with salient points obtained by the structural tensor detector is visible in Fig. 7a; The right image with the matched points in Fig. 7b. Two outliers are detected – see Table 2.

Table 1. Matching results of the images from Fig. 6. Outliers entries no.2 and no.5.

No	Left pt. (x, y)	Right pt. (x, y)	Best match val. ρ	LP (*scale, rotation*)
1	248, 51	(253, 52)	0.929104	(0, 0)
2	158, 66	(253, 48)	0.924074	(15, 2)
3	227, 24	(232, 25)	0.912579	(0, 0)
4	16, 105	(20, 105)	0.840755	(0, 0)
5	288, 80	(293, 144)	0.634407	(-14, 4)
6	135, 117	(135, 117)	0.746453	(0, 1)
7	170, 136	(173, 136)	0.876785	(0, 0)
8	192, 175	(199, 175)	0.846410	(0, 0)

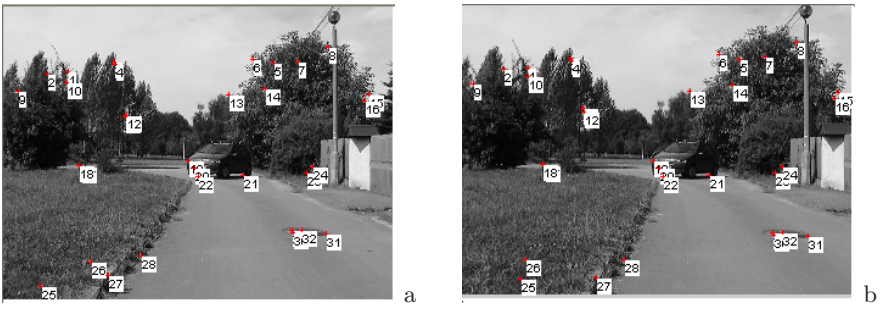


Fig. 7. Matching of the "Car" stereo pair. Left image (a) with salient points from the ST detector. The right image with the matched points (b). Two outliers.

Table 2. Results of the LP matching of the images in Fig. 7. Outliers no. 25 and 26.

No	Left pt. (x, y)	Right pt. (x, y)	Best match val. ρ	LP (*scale, rotation*)
1	(53,53)	(54,51)	0.929504	(0,0)
2	(36,55)	(34,52)	0.826554	(0,0)
...	...	...	...	...
25	(31,229)	(48,224)	0.807608	(0,13)
26	(72,209)	(52,208)	0.795199	(0,0)
...	...	...	...	...
32	(245,183)	(263,186)	0.976108	(0,0)

Matrix $\mathbf{F}$ obtained from point correspondences in Fig. 7 can be seen in the following formula. Computation is performed in accordance with the formulas (5) and (6) presented in the previous sections.

$$\mathbf{F} = \begin{bmatrix} 0.492126 & -0.343832 & 0.1690080 \\ -0.269639 & -0.190649 & 0.0927313 \\ 0.167368 & 0.117041 & -0.0574843 \end{bmatrix} \tag{29}$$

The achieved accuracy is in the order of 10^{-6}. Prior to matching the outliers were removed and points normalized in accordance with (11).

Although LP search is a four dimensional one, usually this does not pose a problem if a geometry can be computed off line. Otherwise, the hardware accelerated computations are an option.

Fig. 8 and Fig. 9 present computation of the point correspondences in the stereo pairs "Parkmeter" and "Tsukuba", respectively. For the "Tsukuba" stereo-pair each of the LP patches has size of 27×27 pixels, corners are detected independently in 4×4=16 tiles. Found point correspondences were used for computation of the fundamental matrix with the described method (2).

The method allows detection of the point correspondences with simultaneous rejections of outliers. Based on the experiments we found that the best results are obtained if the square LP windows are of size from 15 to 35 pixels. This parameter depends on contents of an image and its resolution.

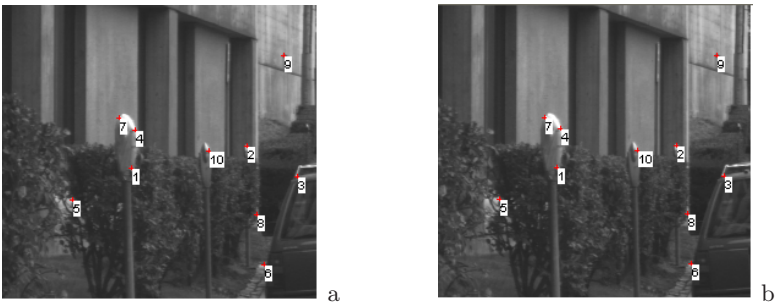


Fig. 8. Matching of the "Parkmeter" stereo pair. Left image (a) with salient points obtained by the ST detector. The right image with the matched points (b). No outliers.

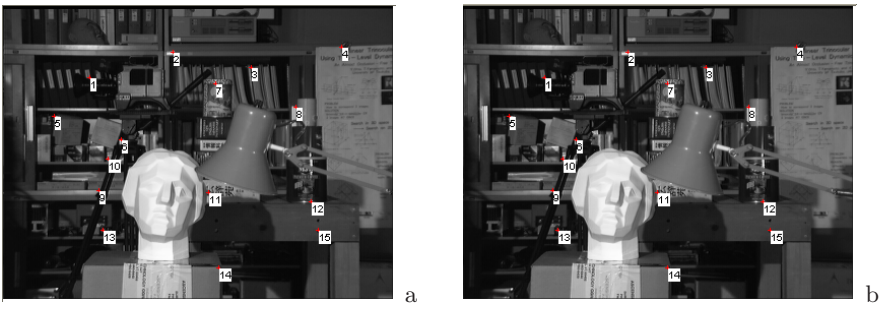


Fig. 9. Matching the "Tsukuba" stereo-pair divided into 4×4 tiles. 27×27 LP window used for matching. Left (a) with detected corners. The right with the matched points (b). No outliers.

6 Conclusions

In this paper a novel method for efficient computation of a scene geometry is presented. It is composed of two stages. The first one consists in computation of the salient points (corners). For this purpose the structural tensor is employed which allows fast selection of the strongest signal responses which also exhibit high discriminative properties. The second step consists in region matching around salient points. However, instead of bare intensities their log-polar representations are used for this purpose. One of the advantageous of this approach is possible detection of outliers based on their excessive local rotation and/or scale. From the point correspondences a geometry of multiple views can be recovered. For two views this is achieved by computation of the fundamental matrix F. A brief description of this process is also presented. The method was verified experimentally on the video sequences and stereo images, allowing accurate computation of a scene geometry.

Acknowlegement

This work was supported from the Polish funds for the scientific research in 2007.

References

1. Bigün, J., Granlund, G.H., Wiklund, J.: Multidimensional Orientation Estimation with Applications to Texture Analysis and Optical Flow. IEEE PAMI 13(8), 775–790 (1991)
2. Brox, T., van den Boomgaard, R., Lauze, F., van de Weijer, J., Weickert, J., Mrázek, P., Kornprobst, P.: Adaptive Structure Tensors and their Applications. In: Weickert, J., Hagen, H. (eds.) Visualization and Processing of Tensor Fields, pp. 17–47. Springer, Heidelberg (2006)
3. Faugeras, O.D., Luong, Q.-T.: The Geometry of Multiple Images. MIT Press, Cambridge (2001)
4. Fischler, M.A., Bolles, R.C.: Random sample consensus. Communications of the ACM 24(6), 381–395 (1981)
5. Hartley, R.I.: In Defense of the Eight-Point Algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(6), 1064–1075 (1997)
6. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
7. Morita, T., Kanade, T.: A Sequential Factorization Method for Recovering Shape and Motion From Image Streams. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(8), 858–867 (1997)
8. Pratt, W.K.: Digital Image Processing, 3rd edn. Wiley, Chichester (2001)
9. Shapiro, L.S., Zisserman, A., Brady, M.: 3D Motion Recovery via Affine Epipolar Geometry. Int. Journal of Computer Vision 16, 147–182 (1995)
10. Torr, P.H.S., Fitzgibbon, A.W.: Invariant Fitting of Two View Geometry. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5), 648–650 (2004)
11. Zokai, S., Wolberg, G.: Image Registration Using Log-Polar Mappings for Recovery of Large-Scale Similarity and Projective Transformations. IEEE Tr. on Image Processing 14(10), 1422–1434 (2005)

The Power of Tokens: Rendezvous and Symmetry Detection for Two Mobile Agents in a Ring

Jurek Czyzowicz[1], Stefan Dobrev[2], Evangelos Kranakis[3], and Danny Krizanc[4]

[1] Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec J8X 3X7, Canada. Research supported in part by NSERC
[2] School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada, on leave from Slovak Academy of Sciences, Bratislava, Slovakia. Research supported in part by NSERC
[3] School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6, Canada. Research supported in part by NSERC and MITACS
[4] Department of Mathematics, Wesleyan University, Middletown, Connecticut 06459, USA

Abstract. Rendezvous with detection differs from the usual rendezvous problem in that two mobile agents not only accomplish rendezvous whenever this is possible, but can also detect the impossibility of rendezvous (e.g., due to symmetrical initial positions of the agents) in which case they are able to halt. We study the problem of rendezvous with and without detection of two anonymous mobile agents in a synchronous ring. The agents have constant memory and each of them possess one or more tokens which may be left at some nodes of the ring and noticed later. We derive sharp bounds for the case of at most two tokens per agent and also explore trade-offs between the number of tokens available to the agents and the time needed to accomplish rendezvous with detection.

1 Introduction

The mobile agent *rendezvous* problem is a search optimization problem that seeks an algorithm specifying how should anonymous mobile agents move along the vertices of a network in order to determine whether or not they can meet at some node of the network. The mobile agents are autonomous entities that move along vertices of the network acting on the information collected following a given protocol.

In certain instances, if the rendezvous problem is impossible to solve the mobile agents will be executing a rendezvous algorithm that may never terminate. It is therefore crucial for accomplishing rendezvous what knowledge the mobile agents have about the network configuration (e.g., size of the network, relation between start positions of the mobile agents). In this paper we distinguish between two types of rendezvous problem: *rendezvous without detection* (also simply *rendezvous*, abbreviated $\mathcal{RV}$) and *rendezvous with detection* (abbreviated $\mathcal{RD}$). In the former case, the agents know that the rendezvous problem

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 234–246, 2008.

has a solution either for the given system configuration or regardless of the system configuration and they just want to accomplish rendezvous at a node of the ring in, say, minimum number of steps. (For example, in a ring of size n the rendezvous problem is always solvable for m mobile agents if m and n are relative primes). In the latter case, we are also interested in the decision problem which in addition to rendezvous requires a solution of the *halting problem* for rendezvous. I.e., we look for an algorithm that detects feasibility of a solution for all starting positions after a *finite number of steps* (usually dependent either on their distance or the size of the network). Thus, if rendezvous is possible then rendezvous is achieved, while if rendezvous is not possible then all agents stop and know that rendezvous is not possible.

In this paper, of interest will be mobile agents that have constant memory-space (independent of the size of the ring and the number of mobile agents in the system) and thus are unable to remember either their past actions or the current conditions of the system. Conditions for the mobile agents we consider may include 1) constant memory-space and 2) a given number of tokens per mobile agent. In addition, it is assumed that the tokens are indistinguishable from each other as well as from the tokens of the other agent(s). In other words, when an agent finds a token that has been released at a node in the ring neither can it distinguish it from its own token(s) nor from the token(s) of the other mobile agents. We distinguish two cases: unidirectional ring, where agents may move in a single, say counterclockwise, direction, and bidirectional ring - where clockwise and counterclockwise moves are allowed.

1.1 Results and Outline of the Paper

The paper studies rendezvous with detection ($\mathcal{RD}$) and explores trade-offs between the number of tokens available to the agents and the time needed to rendezvous with detection on an n node ring. In more detail, Section 2 includes impossibility, as well as upper and lower bound results for two mobile agents with constant memory and at most two tokens each. Main results of Section 2 are summarized in Table 1. The first column depicts the number of tokens available per mobile agent, the second indicates the number of directions on the ring (1 means unidirectional and 2 bidirectional), while the third and fourth columns

Table 1. Time bounds for two mobile agents with constant memory to detect if rendezvous is possible ($\mathcal{RD}$) and to rendezvous when input is asymmetric ($\mathcal{RV}$) on an n node synchronous uni-, bi-directional ring with one or two tokens

Conditions on		Time Required for	
Tokens #	Dirs #	$\mathcal{RD}$	$\mathcal{RV}$
1	1	∞	∞
1	2	∞	$\Theta(n^2)$
2	1	$\Theta(n^2)$	$\Theta(n^2)$
2	2	$\Theta(n^2)$	$\Theta(n^2)$

indicate the time required to solve the problem indicated. The memory required for all the algorithms depicted is $O(1)$. In Section 3 we show how the time complexity of $\mathcal{RD}$ algorithm may be improved if more than two tokens per agent are allowed. In particular, we look at the case where each mobile agent has $t \geq 3$ tokens and memory $O(\log t)$. We give an $O(mn)$ upper bound for $\mathcal{RD}$, where m is the smallest integer such that $\binom{m-1}{t-2} \geq n-1$.

1.2 Mobile Agent Model

In this paper we consider an anonymous, synchronous, either unidirectional or bidirectional ring network on n nodes. The nodes of the ring are identical and do not have distinct identities. Two mobile agents are situated on the nodes of the ring. Each mobile agent owns a set of tokens that it may release at any node of the ring. The tokens are indistinguishable from each other. At any single time unit, the mobile agent occupies a node of the ring and may 1) stay at its current position, 2) move left or right from its current position, 3) detect the presence of one or more tokens at the node it is occupying, and 4) release/remove one or more tokens to/from any node it is occupying. Initially, a mobile agent may occupy any node of the network and the start node of the mobile agent is also called its *home node* or *home base*. Both agents start their algorithm at the same time. Mobile agents may communicate and exchange information with each other only when they find themselves at the same node. We say that one or more mobile agents rendezvous when either they meet at the same node or else traverse the same edge in opposite directions. They can see, remove or add tokens, and can also see each other if either they are crossing over an edge or arriving at a node at the same time. The computation is synchronous and both agents work in lock-step.

Definition 1 (Finite automaton). *More formally, the two agents holding t tokens each are starting at their home bases which are located at different nodes of a ring of unknown size n. In a bidirectional ring an agent is a finite automaton with k states represented by a* state-transition *function $\sigma : S \times T \times T \to S \times T \times \{L, R, W\}$ where $S = \{1, 2, \ldots, k\}$ and $T = \{1, 2, \ldots, 2t\}$. Moreover, whenever $\sigma(s, t_1, t_2) = (s', t_1', d)$ it must hold $t_1' \leq t_1 + t_2$. The parameters mentioned above are to be interpreted as follows: 1) The set S represents the agents' states, not including the number of tokens the agent holds. 2) $t_1 \in T$ is the number of tokens the agent holds when making the decision, and $t_2 \in T$ is the number of tokens the agents see in the current node. t_1' is the new number of tokens the agent holds after making a state transition. (Note that t_1, t_2, but also $t_1 + t_2$ are always at most $2t$, as the total number of tokens in the system is $2t$. Moreover, as the agent can pick up only the tokens it sees, the number of tokens it carries after the state transition is bounded by $t_1 + t_2$.) 3) L, R, W represent the "movement" decision by the agent – either moving one position* Left, Right *(according to the specified global orientation) or* Wait. *(For a unidirectional ring the moves L, R, W have to be modified to M, W meaning either* Move *or* Wait*, respectively.)*

We note that this definition of the automaton represents only a reference model and is meant to be used only as a guide. In actual proofs we will avoid strict conformance to its specific workings as it would lead to long and tedious constructions.

1.3 Related Work

The rendezvous problem was first considered by the operations research community as a search game for two or more players in various network topologies (see [1], [2]). In its current algorithmic form with one token per agent the problem was first considered in [10] and more extensively in the PhD thesis of [8]. The $\mathcal{RD}$ problem is mentioned in [7] and studied in the case of the torus in [6] where also the power of tokens for rendezvous is investigated. A related "whiteboard" model can be found in [3] whereby an agent is allowed to write messages of certain size on a white board that can be read by the other agent. So, in a sense, the token model presented here can be thought of as the "weakest possible" form of the white board model. For additional information on algorithmics for mobile agent models the reader is advised to consult [5]. General discussion of the relation of the rendezvous problem to P2P networks can be found in the edited volume [9].

2 Mobile Agents with at Most Two Tokens

In this section we consider time upper bounds for rendezvous with detection in a ring when the mobile agents have constant memory-space and at least one token each. We consider an n-node ring and two mobile agents each having a number (same for both mobile agents) of indistinguishable tokens. It goes without saying that the mobile agents cannot know the size n of the ring since they have only constant size memory, independent of n (unless n itself is a constant).

2.1 Upper Bounds for Rendezvous with Detection

First consider the case where each mobile agent has a single token and the ring is bidirectional. In this case the asymmetric rendezvous problem is solvable, as shown in the following theorem.

Theorem 1. *In bidirectional rings, the rendezvous problem ($\mathcal{RV}$) is solvable in $O(n^2)$ time for two mobile agents having constant memory and one token each.*

Proof. Consider the *pendulum-like* algorithm presented below. First, note that if the agents have different notion of which direction is right, they will start moving towards each other and will meet in $O(n)$ time. Hence, it is sufficient to consider the case where the agents have the same sense of direction. Let us call the agents A and B and the initial distances between them be d and $n-d$, respectively. Let $t_1, t_2, t_3, \ldots$ be the times when the agent A finds a token (i.e. t_i is the time when i-th iteration of the loop begins). Let $t'_1, t'_2, t'_3, \ldots$ be those times for the agent B.

Algorithm 1. Algorithm One-Token.

```
1: Drop your token at your home base.
2: Go right until a token is found or you meet the other agent, counting (in your state)
       the value of x equal to the distance traveled modulo three. Let this distance be
       x (remember it in your state).
3: if not met the other agent
4:     repeat
5:         Pick the token, reverse direction, move one step and drop the token there.
6:         Continue in current direction until a token is found, counting the value of
       y equal to the distance travelled mod3.
7:     until y ≡ (x − 1) mod 3 or met the other agent
8:     if did not meet the other agent.
9:         Stop and wait for the other agent.
10:    endif
11: endif
```

Without loss of generality assume that $d < n - d$ and set $\delta = (n - d) - d$. Note that $d = t_1 \equiv x_A \bmod 3$ and $n - d = t'_1 \equiv x_B \bmod 3$. Observe that as long as $t_{i+1} < t'_i$, the agent B will move a token before the agent A arrives to it, and A will find the distance between the tokens has not changed. However, as $t_1 \neq t'_1$, it is easy to prove by induction that the value of $t'_i - t_i$, which is the difference between the times when both agents start their i-th iteration, increases by δ, i.e. $t'_i - t_i = i\delta$. That means that after $\lceil t_1/\delta \rceil$ iterations $t_{i+1} \geq t'_i$. If $t_{i+1} = t'_i$, the agents meet over the token. Otherwise, agent A arrives to the token before B had moved it, i.e. A traveled distance $t_1 - 1 \equiv x_A - 1 \bmod 3$. At that moment, A will stop and start waiting for B. As B has so far measured only equal distances, it will continue and eventually arrive at the place where A is waiting. Since there are t_1/δ iterations of t_1 steps each, plus final time at most t'_1 for the agent B to arrive to the meeting place, the rendezvous will happen in time $O((t_1/\delta)t_1 + t'_1)$. Since $\delta \geq 1$, $t_1 < n/2$ and $t'_1 = t_1 + \delta < n$, the resulting time is $O(n^2)$. This completes the proof of Theorem 1.

Algorithm One-Token solves $\mathcal{RV}$ but not $\mathcal{RD}$, as it will run forever if the agents are initially in a symmetric configuration. Next we show that rendezvous with detection can be solved if we endow each agent with two tokens, even in unidirectional rings.

Theorem 2. *Rendezvous with detection ($\mathcal{RD}$) is solvable in a unidirectional ring for two mobile agents with constant memory and two tokens each, in time $O(n^2)$.*

Proof. We present an algorithm that at the cost of using two tokens per mobile agent detects the possibility of rendezvous and can eventually rendezvous when possible. Formally we have the following algorithm. Each mobile agent leaves one token at its home node and the other token at the neighboring node located to its right. Then it travels right and moves every second token one position to the right (note that this will keep the home node tokens at their original

Algorithm 2. Algorithm Two-Tokens.

```
1: Drop first token at your home base and second token to node located to the right.
2: repeat
3:     Travel right and move every second token you meet one position to the right.
4: until agent detects two tokens on top of each other.
5:     if two tokens are found on top of each other go around and check if other two
       tokens are also on top of each other.
6:         if yes then rendezvous is not possible else agent waits at last position.
7:         endif
8:     endif
```

locations). The process is repeated until the agent detects two tokens at the same node. When this happens, the agent continues traveling to check whether the other two tokens are also at a same node. If they are, then the home nodes were $n/2$ away, the whole computation was symmetric and the agents can never rendezvous. If the other pair of tokens are not at a same node, then the agent waits as the other agent will eventually come to meet it. The running time of the algorithm is as claimed. This completes the proof of Theorem 2.

2.2 Impossibility Results

As shown in Theorem 2, two tokens per agent and unidirectional ring are sufficient to solve $\mathcal{RD}$ (and hence also $\mathcal{RV}$). On the other hand, the one-token, bidirectional ring algorithm from Theorem 1 fails to solve $\mathcal{RD}$ if the initial configuration is symmetric, as it will cycle forever. Note, that $\mathcal{RD}$ is trivially solvable if agents possess $\Omega(\log n)$ memory, i.e. if the number of states exceeds the size n of the ring, even if the agents have only one token each and the ring is unidirectional. In such a case, the agent leaves the token at its home base and counts in states the distance to the token of the next agent, and checks whether this is equal to the distance to the next (his own) token. (In fact $O(\frac{\log n}{\log \log n})$ states are sufficient. See [7].) Hence, the remaining interesting cases, addressed in this section, are agents with constant memory and one token each, for either unidirectional or bidirectional rings.

Let us first introduce some tools that will be used in the impossibility and lower bound proofs. Consider an agent moving through a tokenless path of the ring. On one hand, if the agent carries no tokens, after at most $k+1$ moves it will repeat a previously encountered state and will continue moving in the same direction until a token is encountered. On the other hand, if the agent carries a token, after at most $2k+1$ moves two identical states will be repeated, which means that the agent has fallen into a cycle and will repeat its activity until it encounters another token. (Note that it is not necessary for the agent to carry the token the whole time – it may leave it momentarily and pick it up after a few steps and move it, but all this process is cyclically repeated, the agent is in fact carrying the token.) If no identical states with the agent carrying the token appear within the first k steps, the agent will leave the token and then it

will continue moving, carrying no token, until it arrives to another token. (This idea can be extended to unidirectional rings and t tokens.) This leads us to the following definition of the *base* of an agent.

Definition 2 (Agent's Base). *A base is a contiguous segment of the ring delimited by nodes containing tokens or agents carrying a token. An* initial base *is formed when an agent leaves a token for the first time, and it consists of the single node containing this token. As the agents start in the same state, this means that initially there are two bases, at the same distance as the starting distance between the agents. When an agent holding a token leaves a base, the base expands to contain that token if and only if the agent has not entered a cyclic behavior that will make it carry the token all the way to the opposite base. Otherwise, that token ceases to be part of the base and the base shrinks to enclose its remaining tokens.*

Note that if the agents have a single token and the bases are left tokenless according to the above definition, the agents will start an infinite cycle carrying the tokens and chasing each other. If the agents have two tokens and they start carrying a token to the other base, the bases shrink to contain a single token and the algorithm effectively resets itself (if this happens more then k times, the algorithm will cycle forever).

Definition 3 (Time shift). *Let us denote by the increasing integer $t_1, t_2, \ldots$ the times when agent A arrives to alternating bases. More precisely, t_1 is the time when A arrives for the first time to the base of B, t_2 is the first time after t_1 that A arrives to its own base, t_3 is the first time after t_2 that A arrives to B's base). Analogously, we define the time sequence $t'_1, t'_2, \ldots$ for agent B. Let $\delta_i = |t'_i - t_i|$ and call it* time shift of round i.

Let us denote by s_A^t and s_B^t the state (including the number of tokens being carried) of agents A and B, respectively, at time t. The following lemma forms the core of our impossibility and lower bound proofs. It states that it is possible to choose the initial distances between the agents in a large enough ring in such a way that the agents essentially arrive to the same state (but, possibly, time-shifted) for long enough time. Note that the lemma does not assume unidirectional ring.

Lemma 1. *Let the two initial distances between the agents on a ring be M and $M + xk!$ for some $M \geq k!$ and an arbitrary $x \geq 0$. As long as $\delta_r < M$ and the distance between the bases is at least $k + 1$, it holds that $\forall i < r, \forall j \leq \min(t_{i+1} - t_i, t'_{i+1} - t'_i) : s_A^{t_i+j} = s_B^{t'_i+j}$ (and those states would be the same for any other natural number x). Moreover, at time t_i, the base to which A arrived has exactly the same configuration as the base to which agent B arrived at time t'_i.*

Proof. The proof is by induction on r. Initially, we have $t_0 = t'_0 = 0$, $s_a^0 = s_b^0$ and the initial bases have the same configuration (single node with single token each), as the agents start at the same state and at the same time. *Induction step:* Assume that $\delta_r < M$ and the distances between the bases is at least $k + 1$.

Moreover, assume that the base of agent A at time t_{r-1} and the base of agent B at time t'_{r-1} are in the same configuration. We will prove that the base of A at time t_r is in the same configuration as the base of B at time t'_r and that $s_A^{t_{r-1}+j} = s_B^{t'_{r-1}+j}$ for all $j \leq \min(t_r - t_{r-1}, t'_r - t'_{r-1})$.

As at time t_{r-1} agent A arrives to a base, the configuration of the base captures its state (and the same is true for agent B at time t'_{r-1}). Since the base configurations are the same (by induction hypothesis), that means that $s_A^{t_{r-1}} = s_B^{t'_{r-1}}$ and the agents arrive to the same side of the corresponding bases. The fact that the base configurations are the same also means that the agents continue behaving the same as long as the environment they see is the same. As the bases contain all tokens in the ring, the first moment when there is difference in what the agents see is when one of them arrives to the opposite base, i.e. after $\min(t_r - t_{r-1}, t'_r - t'_{r-1})$ time steps. Note that up to that moment, the configurations of both bases stayed equal (as the agents did the same modifications).

If $x = 0$, the situation is symmetric and both agents arrive to the bases at the same time and in the same state and the lemma holds, Consider now the case $x \neq 0$. Without loss of generality assume that agent A is the first one to arrive to the opposite base (at time t_r). Note that as the distance between the bases is at least $k + 1$, A must have been in a cycle (of states), moving towards the opposite base. As up to this moment B behaved the same, B must also be in such cycle, moving towards its opposite base. This means it cannot return to the base to modify it, and the base configurations remain equal.

It remains to be shown that when agent B arrives to its opposite base at time t'_r, it will be in the same state (i.e. $s_B^{t'_r} = s_B^{t'_{r-1}+t_r-t_{r-1}} = s_A^{t_r}$). From the fact that initial distances between the bases differed by $xk!$ and the base configurations remain the same, it follows that the distances between the bases remain different by $xk!$, i.e. at time $t'_{r-1} + t_r - t_{r-1}$, B is at distance $xk!$ from its opposite base. We already know that at this moment B is moving in a cycle of states of length at most k. Let l be the forward distance traveled by B in this cycle. Obviously $l \leq k$ (it can be $l < k$ if B zig-zags). However, that means that l divides $xk!$, i.e. when B arrives to its opposite base, it will be in exactly the same state. This proves Lemma 1.

Theorem 3. *Neither the rendezvous problem ($\mathcal{RV}$) nor rendezvous with detection ($\mathcal{RD}$) is solvable for two identical mobile agents having constant memory and one token each in a unidirectional ring.*

Proof. Consider the two mobile agents with one token each in a unidirectional ring and suppose they are represented by identical automata with k states. We will exhibit two different configurations, one symmetric and one asymmetric, that the two agents cannot distinguish. First, we consider a symmetric configuration whereby the two mobile agents start at distance $k!$ from each other in a ring of size $2k!$. As long as the agents do not drop their tokens, they see the same environment (no tokens) and therefore behave (state changes and moves performed) identically. Hence, in order to rendezvous or detect that it is not

possible, the agents will eventually drop their tokens, and they will be at distance $k!$ from each other at that moment.

Consider now two rings with agents starting $k!$ apart, one symmetrical of size $2k!$, the other asymmetrical of size $3k!$. Note that the conditions of Lemma 1 are satisfied, as the first ring corresponds to $M = k!$, $x = 0$, while for the second $M = k!$ and $x = 1$. Observe that because the ring is unidirectional, we get $\forall i : t_{2i} = t'_{2i}$ in both cases (even if distances between the bases are not equal, each agent alternates between traversing the short and long distance). That means that Lemma 1 holds for all r. In particular, $\forall i : s_A^{t_{2i}} = s_B^{t'_{2i}}$ and those states are the same for all x. What that means is that either the algorithm never terminates, or, if it terminates, it produces the same output for a symmetric ring of $2k!$ nodes and for the asymmetric ring of $3k!$ nodes, i.e. the algorithm is incorrect.

Next we consider the case of bidirectional rings. In view of Theorem 1 it is no longer true that $\mathcal{RV}$ is unsolvable. However, it can be shown that $\mathcal{RD}$ remains unsolvable if we limit ourselves to agents with one token and constant memory.

Theorem 4. *Rendezvous with detection ($\mathcal{RD}$) is not solvable for two identical mobile agents having constant memory and one token each in a bidirectional ring.*

Proof. The overall structure of the proof is similar to the proof of Theorem 3: we apply Lemma 1 to two configurations (one symmetric and other asymmetric) and show that either the algorithm does not terminate in the symmetric configuration, or produces wrong output in the asymmetric case. The symmetric case is a ring of size $n = 2N + 2k!$ (corresponding to $M = N + k!$, $x = 0$, where N is determined later), the asymmetric case uses the same M but $x = 1$. Assume that the algorithm correctly decides in the symmetric ring, after r rounds (r, cf. Lemma 1, corresponds to how many times an agent switched bases). Since the agents have k states, we get that r must be at most k (otherwise the algorithm would cycle forever). We will define N in such a way that Lemma 1 applies for at least $k + 1$ rounds (for the faster agent) in the asymmetric case. That would ensure that in the asymmetric execution the same deciding state appears and the algorithm decides incorrectly.

How much can δ_i grow in each round? The only difference the agents experience is that the distances between the bases differ by $k!$. In one round, each agent crosses this distance exactly once. Let c represent the maximal time it takes an agent to travel one step (it can be more than one, as the agent can zig-zag, but it is a constant as the agent has k states). Then, each round the time shift can grow by at most $ck!$ and after i rounds the time shift is at most $ick!$. Choosing $N > ic(k + 1)!$ ensures that Lemma 1 applies for at least $k + 1$ rounds.

2.3 Lower Bounds for Rendezvous

In Section 2.1 we have shown $O(n^2)$ upper bounds for settings not excluded by the impossibility results from Section 2.2. An obvious question to ask is

whether these upper bounds are tight. In this section we provide an affirmative answer by proving an $\Omega(n^2)$ lower bound for $\mathcal{RV}$ in bidirectional rings for agents with two tokens. This implies the same lower bound also for $\mathcal{RD}$, as well as for unidirectional rings and single-token agents. The main result of this section is the following theorem.

Theorem 5. *The rendezvous problem ($\mathcal{RV}$) for two mobile agents having constant memory and two tokens each require $\Omega(n^2)$ time in a bidirectional ring of size n.*

Proof. Consider a ring of size $n = 2N + k!$ with the agents starting at distance N from each other. Again, Lemma 1 applies while $\delta_r < N$ and the distance between the bases is at least $k+1$. We show that (1) if $\delta_r < N/3$ holds until the bases reach size $N/3$, then it takes $\Omega(N^2)$ time to increase the bases to size $N/3$, and (2) if the base sizes are at most $N/3$, then it takes $\Omega(N^2)$ time to increase δ_r to $N/3$. When combined with a choice of $N \geq k!$ this yields the desired $\Omega(n^2)$ lower bound.

Assume first that the bases reach size $N/3$ before δ_r exceeds $N/3$. Note that at the moment a base has 2 tokens for the first time, the size of the base is at most k (otherwise an agent is carrying the second token in a cycle and by definition it is not a part of the base). Moreover, an agent can increase the size of the base by at most k before it enters a cyclic behavior – which will either take it to the other base, or to the other end of the current base.

Let x_j be the size of the base at the moment an agent is leaving the base's endpoint in a cycle for the j-th time (called *quasi-round* j). Let p_j and p'_j, respectively, be the times this happens for the two bases. We get that $x_{j+1} \leq x_j + k$ and $p_{j+1} \geq p_j + x_j$: The base grows either when an agent traverses it from one end to another (obviously taking x_j time), or when an agent moves from one base to another, yielding $p_{j+1} \geq p'_j + (N - x_j)$. As the time difference is at most $N/3$ and $x_j \leq N/3$, we get $p_{j+1} \geq p'_j + (N - x_j) \geq p'_j + 2N/3 \geq p_j + N/3 \geq p_j + x_j$. As $x_{j+1} - x_j \leq k$ there must be at least $N/3k$ quasi-rounds. Let f be the final quasi-round, i.e. $x_f \geq N/3$. Summing over all rounds we get $p_f = \sum_{j=1}^{f} x_j = \sum_{j=0}^{f-1} x_{f-j} \geq \sum_{j=0}^{f-1}(N/3 - jk) \in \Omega(N^2)$, as $f \geq N/3k$ and k is a constant.

Consider now the case that δ_r exceeds $N/3$ before the bases reach size $N/3$. Using the same argument as in the proof of Theorem 4, we get that $\delta_i \leq ick!$, i.e. $r \geq N/3ck!$. As the base sizes are at most $N/3$, each round takes at least $2N/3$ time steps (each round involves traversing from one base to another, by definition of a round). Summing up over all r rounds we get the lower bound $N^2(2/9)ck!$, which is $\Omega(n^2)$, since c and k are constants.

3 Mobile Agents with More Than Two Tokens

Now that we have the complete picture for rendezvous with detection when each mobile agent has at most two tokens we look for the more general case whereby each mobile agent has more than two tokens.

3.1 Upper Bounds for Rendezvous with Detection

The first theorem provides a trade-off between number t of tokens being used and time required for $\mathcal{RD}$. We consider the case of at least three tokens per mobile agent, i.e., $t \geq 3$.

Theorem 6. *Consider a synchronous, bidirectional ring with n nodes and two mobile agents located at two of its nodes. Rendezvous with detection ($\mathcal{RD}$) is solvable for two mobile agents having $t \geq 3$ tokens and $O(\log t)$ bits of memory each in time $O(mn)$, where m is the smallest integer such that $\binom{m-1}{t-2} \geq n-1$.*

Proof. A basic idea of the algorithm is to implement a counter C_t, that can count up to n. The counter will be represented by a segment of nodes of the ring containing up to t tokens at its nodes, delimited by two tokens at a maximum distance m, say, from each other. The values that this counter takes are held within this segment of nodes of the ring; one of the two tokens delimiting it is located at the home base of the agent while the other is the last token that the mobile agent released at a node of the ring at distance m from its home base.

Assuming that such counter exists, we can proceed just like in the proof of Theorem 2. The basic idea for $\mathcal{RD}$ is to have an agent go from its home base to the home base of the other agent, while incrementing its counter by one once in each round. After the counter reaches its maximum, the agent continues to go from its home base to the base of the other agent but now decrementing its value by one once in each round. Notice that the counter can be incremented/decremented in time $O(m)$ per round. As with the algorithm of Theorem 2, when the counter reaches 0 before encountering its own home base, the mobile agent goes to the first base, and if the counter of the other mobile agent reaches exactly 0 at the second base as well, then the situation is symmetric and rendezvous is impossible. Otherwise this agent waits at the second base until the other agent comes there and the rendezvous is accomplished. Therefore the algorithm whose idea has just been described not only reaches rendezvous, whenever possible, but also detects when it is not possible within the same time bound. The running time of the algorithm presented will be $O(mn)$ since each round takes n steps which is the size of the ring.

Clearly, the running time of the algorithm is directly proportional to how compact the counter C_t can be, as the cost of moving is proportional to its size m, which is the distance between the two tokens delimiting the counter C_t. Therefore for the given number t of tokens it remains to determine m so that the mobile agent can implement a counter that can hold a maximum value n. By assumption, each mobile agent has t tokens. One token is being used to mark the mobile agent's home base thus leaving $t-1$ tokens that can be used to implement the counter. The technical part is how to implement the counter, with the remaining $t-1$ tokens. The counter will be delimited by two tokens, located in nodes A, B, at distance m apart. A token is located at the home base A, say, of an agent. This leaves $t-1$ tokens for marking positions at nodes of the network. Another token located at B increments the range of the counter. Since the agent can count internally to $t-1$ (since it has $t-1$ tokens), all possible

combinations of $t-3$ tokens between two fencing tokens at distance k can be tried, and afterward increment k and repeat until the home base of the other mobile agent is reached, where $k \leq m$.

It remains to investigate what the size of the counter should be so as to guarantee that it is able to count up to n. For given k, there are $\binom{k-2}{t-3}$ possibilities (assuming no two tokens can can be left at the same node, but appropriate combination numbers can be derived for that as well). Summing up over all $k \leq m$ until the home base of the other agent is reached results in at most $\sum_{k=2}^{m} \binom{k-2}{t-3} = \binom{m-1}{t-2}$ possibilities (see [4][page 56]). Since the position of the home base of the other agent is at most $n-1$ the counter C_t needs to count up to $n-1$. Therefore the value of m will never need to exceed the smallest m such that $\binom{m-1}{t-2} \geq n-1$. Further, notice that the two agents are required to have $O(\log t)$ bits of memory so that they can count internally up to t and thus distinguish the two delimiters of the counter C_t. This completes the proof of Theorem 6.

Corollary 1. *Rendezvous with detection ($\mathcal{RD}$) is solvable for two mobile agents having $t > 2$ tokens and memory $O(\log t)$ each, in time $O(n^{\frac{t-1}{t-2}} t)$ in a bidirectional ring. Moreover, if $t = \log n$ then the algorithm works in time $O(n \log n)$.*

3.2 Lower Bounds for Rendezvous in Unidirectional Rings

Very little is known concerning lower bounds for agents with more than two tokens. However, for the case of unidirectional rings we can improve on the result of Theorem 5 thus relating the rendezvous time with the number of tokens available to each agent.

Theorem 7. *The rendezvous problem ($\mathcal{RV}$) for two mobile agents having constant memory and t tokens each requires $\Omega(n^2/t)$ time in an unidirectional ring of size n. Moreover, there is an algorithm achieving this bound.*

Proof. This is similar to the proof of Theorem 5. As before, we take a ring of size $n = 2N + k!$, with the agents starting at distance N. Because the ring is unidirectional, $\delta_{2i} = 0$ and $\delta_{2i+1} = k!$, as in two consecutive rounds each agent traverses the entire ring. Moreover, the quasi-rounds correspond to the rounds from Lemma 1, as an agent cannot reverse direction and traverse and increase the size of the base it have just crossed. An agent having t tokens can increase the size of the base by at most $(t-1)k$: one token should remain to mark the beginning of the base, so an agent carries at any moment at most $t-1$ tokens. In the $(t-1)k+1$ steps after the agent left the right endpoint of the former base, at least two agent configurations (state, number of tokens held) repeat. By the definition of a base, the new base stops growing when the agent starts cycling, i.e. $x_{j+1} \leq x_j + (t-1)k$. That means that Lemma 1 holds for at least $N/(t-1)k$ rounds. As two consecutive rounds last together n time steps, we get that the algorithm cannot terminate before time $nN/(2(t-1)k \in O(n^2/t)$.

The matching upper bound is simple. Each agent goes around and every round it increases the base by t: skip first token of the base, then walk until the second

token is found, pick it up and keep picking up until you have $t-1$ tokens in hand (if you cannot count up to t, just have the tokens next to each other, the first empty space means end-of-base), and then just lay them down one after another. While leaving the tokens of your base, verify if you fall onto the other agent's base.

4 Conclusion and Open Problems

In this paper we studied the rendezvous problem $\mathcal{RV}$ and rendezvous with detection $\mathcal{RD}$ for two variants of a synchronous, anonymous ring: unidirectional and bidirectional. Several challenging problems remain. Some of them concern closing gaps remaining in the trade-offs derived in this paper. Generally, we are lacking general non-trivial lower bounds for $t \geq 3$ tokens. E.g., can we derive sharp upper and lower bounds for t tokens? Another problem is related to the case $t = 3$: are three tokens really more powerful than two tokens (see Theorem 6)? It would also be interesting to look at rendezvous with detection for more than two mobile agents, and also consider the case where no synchrony is assumed.

References

1. Alpern, S.: Rendezvous search: A personal perspective. Operations Research 50(5), 772–795 (2002)
2. Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous. Kluwer Academic Publishers, Norwell, Massachusetts (2003)
3. Dobrev, S., Flocchini, P., Prencipe, G., Santoro, N.: Multiple agents rendezvous in a ring in spite of a black hole. In: Papatriantafilou, M., Hunel, P. (eds.) OPODIS 2003. LNCS, vol. 3144, pp. 34–46. Springer, Heidelberg (2004)
4. Knuth, D.: The Art of Computer Programming, Fundamental Algorithms, 3rd edn., vol. 1. Addison-Wesley, Reading (1997)
5. Kranakis, E., Krizanc, D.: An algorithmic theory of mobile agents. In: Roddick, J.F., Hornsby, K. (eds.)TGC 2006. LNCS, vol. 4661, Springer, Heidelberg (2007)
6. Kranakis, E., Krizanc, D., Markou, E.: Mobile agent rendezvous in a synchronous torus. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, Springer, Heidelberg (2006)
7. Kranakis, E., Krizanc, D., Santoro, N., Sawchuk, C.: Mobile agent rendezvous search problem in the ring. In: ICDCS. International Conference on Distributed Computing Systems, pp. 592–599 (2003)
8. Sawchuk, C.: Mobile Agent Rendezvous in the Ring. PhD thesis, Carleton University, School of Computer Science, Ottawa, Canada (2004)
9. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Wiley, Chichester (2002)
10. Yu, X., Yung, M.: Agent rendezvous: A dynamic symmetry-breaking problem. In: Meyer auf der Heide, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 610–621. Springer, Heidelberg (1996)

How Much Information about the Future Is Needed?*

Stefan Dobrev[1], Rastislav Královič[2], and Dana Pardubská[2]

[1] Institute of Mathematics,
Slovak Academy of Sciences
Stefan.Dobrev@savba.sk
[2] Department of Computer Science,
Comenius University, Bratislava, Slovakia
{kralovic,pardubska}@dcs.fmph.uniba.sk

Abstract. We propose a new way of characterizing the complexity of online problems. Instead of measuring the degradation of output quality caused by the ignorance of the future we choose to quantify the amount of additional global information needed for an online algorithm to solve the problem optimally. In our model, the algorithm cooperates with an oracle that can see the whole input. We define the advice complexity of the problem to be the minimal number of bits (normalized per input request, and minimized over all algorithm-oracle pairs) communicated between the algorithm and the oracle in order to solve the problem optimally. Hence, the advice complexity measures the amount of problem-relevant information contained in the input.

We introduce two modes of communication between the algorithm and the oracle based on whether the oracle offers an advice spontaneously (helper) or on request (answerer). We analyze the Paging and DiffServ problems in terms of advice complexity and deliver tight bounds in both communication modes.

1 Introduction

The term "online" is used to describe algorithms that operate without the full knowledge of the input: a typical scenario would be a server that must continually process a sequence of requests in the order they arrive. More formally, an online algorithm processing an input sequence of requests $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$ produces an output sequence $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle$ in such a way that each y_i is computed as a function of the prefix $\langle x_1, x_2, \ldots, x_i \rangle$. On the other hand, an algorithm computing the whole output sequence $\boldsymbol{y}$ from the entire input sequence $\boldsymbol{x}$ is termed "offline". The systematic study of online problems began in the late sixties [12], and has received much attention over the years (see e.g. [1], [4]). The standard measure used for evaluating online algorithms is the competitive ratio [16], [20], i.e. the worst case ratio between the solution quality of the given online algorithm and that of the optimal offline algorithm. The competitive

* Supported by APVV-0433-06 and VEGA 1/3106/06.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 247–258, 2008.

complexity of an online problem is the best competitive ratio attainable by an online algorithm solving the problem. Intuitively, this measure describes the price, in terms of solution quality, that has to be paid for not knowing the whole input from the beginning.

In this paper we propose a new way of characterizing the complexity of online problems. The hardness incurred by the online setting comes from the fact that there is some information about the future input that is not available to the algorithm. In our approach we measure the amount of this hidden information. However, the input contains also information that is irrelevant to the problem at hand, and we have to find a way of distilling the problem-relevant information from the input.

Our approach to measure the relevant information is inspired by the communication complexity research. We consider, in addition to the algorithm itself, an oracle that sees the whole input and knows the algorithm. When computing the i-th output y_i, the algorithm not only sees the sequence $\langle x_1, x_2, \ldots, x_i \rangle$, but can also communicate with the oracle. We require that the algorithm always computes an optimal solution. The advice complexity of the algorithm is the number of bits communicated between the algorithm and the oracle, normalized per request. The advice complexity of an online problem is the minimum advice complexity over all oracle–algorithm pairs that together solve the problem.

Apart from its theoretical significance, this measure can be of use in some semi-online scenarios where the input is available, but has to be accessed sequentially by the algorithm. As a motivation example, consider the scenario where a simple device (e.g. a remote robot) is supposed to process a large amount of data (e.g. a series of orders) in an online fashion. The data are stored and sequentially fed to the device from a powerful entity (base station) over a (wireless) communication link. In order to guide the robot in the processing, the base station may pre-process the data and send some additional information together with each data item. However, since communication rapidly depletes the robots battery, the amount of this additional communication should be kept as small as possible.

We are primarily interested in the relationship between the competitive ratio and the advice complexity. If the competitive ratio measures the price paid for the lack of information about future, the advice complexity quantifies for how much information is this price paid.

Note that there are two ways to achieve trivial upper bounds on advice complexity: (1) the oracle can send, in some compressed way, the whole input to the algorithm, which then can proceed as an optimal offline algorithm, and (2) the oracle can tell the algorithm exactly what to do in each step. However, both these approaches can be far from optimum. In the first case all information about the future input is communicated, although it may not be relevant[1]. In the second case, the power of the online algorithm is completely ignored. Indeed, an online

[1] Consider, e.g. the PAGING problem. There may be a long incompressible sequence of requests that do not result in a page fault; the information about the actual requests in this sequence is useless for the algorithm.

algorithm may be able to process large parts of the input optimally without any advice, requiring only occasional help from the oracle.

In the paper, we define two modes of interaction with the oracle. In the *helper* mode, the algorithm itself cannot activate the oracle; instead, the oracle oversees the progress of the algorithm, and occasionally sends some pieces of advice. In the *answerer* mode the oracle remains passive, and the algorithm may, in any particular step, ask for advice.

To model the impact of the timing of the communication, let the algorithm work in a synchronous setting: in the i-th step, it receives the i-th input request x_i, and possibly some advice a_i, based on which it produces the output y_i. In a manner usual in the synchronous distributed algorithms (see e.g. [22] and references therein) we count the number of bits communicated between the oracle and the algorithm, relying upon the timing mechanism for delimiting both input and advice sequences [2]. We show that these two modes are different, but are related by $B_H(\mathcal{P}) \leq B_A(\mathcal{P}) \leq 0.92 + B_H(\mathcal{P})$ where $B_H(\mathcal{P})$ is the advice complexity of a problem $\mathcal{P}$ in the helper mode, $B_A(\mathcal{P})$ is the complexity in the answerer mode. Moreover, we analyze two well studied online problems from the point of view of advice complexity, obtaining the results shown in Figure 1. Due to space constraints some of the proofs have been omitted and can be found in the technical report [7].

	competitive ratio	helper	answerer
PAGING	K [24]	$(0.1775, 0.2056)$	$(0.4591, 0.5 + \varepsilon)$
DIFFSERV	≈ 1.281 [8]	$\frac{1}{K}$	$\left(\frac{\log K}{2K}, \frac{\log K}{K}\right)$

Fig. 1. Communication complexities of some online problems compared with competitive ratio (asymptotics for large K)

To conclude this section we note that there has been a significant amount of research devoted to developing alternative complexity measures for online problems. The competitive ratio has been criticized for not being able to distinguish algorithms with quite different behavior on practical instances, and giving too pessimistic bounds [13]. Hence, several modifications of competitive ratio have been proposed, either tailored to some particular problems (e.g. loose competitiveness [26]), or usable in a more general setting. Among the more general models, many forms of *resource augmentation* have been studied (e.g. [15],[21]). The common idea of these approaches is to counterbalance the lack of information about the input by granting more resources to the online algorithm (e.g. by comparing the optimal offline algorithm to an online algorithm that works k-times faster). Another approach was to use a *look-ahead* where the online algorithm is allowed to see some limited number of future requests [3],[15],[25]. The

[2] Alternatively, we might require that both the input requests, and the oracle advices come in a self-delimited form. This would alter our upper bounds by a factor of at most 4, as discussed in the appendix.

main problem with the look-ahead approach is that a look-ahead of constant size generally does not improve the worst case performance measured by the competitive ratio. Yet another approach is based on not comparing the online algorithms to offline ones, but to other online algorithms instead (e.g. Max/Max ratio [3], relative worst-order ratio [6]; see also [9]). Still another approach is to limit the power of the adversary as e.g. in the access graph model [5,14], statistical adversary model [23], diffuse adversary model [18], etc.

Finally, a somewhat similar approach of measuring the complexity of a problem by the amount of additional information needed to solve it has been recently pursued in a different setting by Fraigniaud, Gavoille, Ilcinkas, and Pelc [10,11].

2 Definitions and Preliminaries

An online algorithm receives the input incrementally, one piece at a time. In response to each input portion, the algorithm has to produce output, not knowing the future input. Formally, an online algorithm is modeled by a *request-answer* game [4]:

Definition 1. *Consider an input sequence* $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$. *An* online algorithm $\mathcal{A}$ *computes the output sequence* $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) = \langle y_1, y_2, \ldots, y_n \rangle$, *where* $y_i = f(x_1, \ldots, x_i)$. *The* cost *of the solution is given by a function* $C_{\mathcal{A}}(\boldsymbol{x}) = COST(\boldsymbol{y})$.

In the competitive analysis, the online algorithm $\mathcal{A}$ is compared with an optimal offline algorithm OPT, which knows the whole input in advance (i.e. $\boldsymbol{y} = f(\boldsymbol{x})$) and can process it optimally. The standard measure of an algorithm $\mathcal{A}$ is the competitive ratio:

Definition 2. *An online algorithm is* c-competitive, *if for each input sequence* $\boldsymbol{x}$, $C_{\mathcal{A}}(\boldsymbol{x}) \leq c \cdot C_{OPT}(\boldsymbol{x})$

Let us suppose that the algorithm $\mathcal{A}$ is equipped with an oracle $\mathcal{O}$, which knows $\mathcal{A}$, can see the whole input, and can communicate with $\mathcal{A}$. We shall study pairs $(\mathcal{A}, \mathcal{O})$ such that the algorithm (with the help of the oracle) solves the problem optimally. We are interested in the minimal amount of communication between $\mathcal{A}$ and $\mathcal{O}$, needed to achieve the optimality.

We distinguish two modes of communication: the *helper mode*, and the *answerer mode*. In the helper mode, the oracle (helper) sends in each step i a binary *advice* string $\boldsymbol{a_i}$ (possibly empty), thus incurring a communication cost of $|\boldsymbol{a_i}|$. $\mathcal{A}$ can use this advice, together with the input $x_1, \ldots, x_i$ to produce the output y_i.

Definition 3 (Online algorithm with a helper). *Consider an online algorithm* $\mathcal{A}$, *an input sequence* $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, *and a* helper sequence $\mathcal{O}(\boldsymbol{x}) = \langle \boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n} \rangle$ *of binary strings* $\boldsymbol{a_i}$. *The* online algorithm with helper $(\mathcal{A}, \mathcal{O})$ *computes the output sequence* $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle$, *where* $y_i = f(x_1, \ldots, x_i, \boldsymbol{a_1}, \ldots, \boldsymbol{a_i})$. *The* cost *of the solution is* $C_{(\mathcal{A},\mathcal{O})}(\boldsymbol{x}) = COST(\boldsymbol{y})$, *and the* advice (bit) complexity *is* $B^H_{(\mathcal{A},\mathcal{O})}(\boldsymbol{x}) = \sum_{i=1}^{n} |\boldsymbol{a_i}|$

In the answerer mode, on the other hand, the oracle is allowed to send an advice only when asked by the algorithm. However, this advice must be a non-empty string. For the ease of presentation we define the answerer oracle as a sequence of non-empty strings. However, only those strings requested by the algorithm are ever considered.

Definition 4 (Online algorithm with an answerer). *Consider an algorithm* $\mathcal{A}$*, an input sequence* $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$*, and an* answerer sequence $\mathcal{O}(\boldsymbol{x}) = \langle \boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n} \rangle$ *of non-empty binary strings* $\boldsymbol{a_i}$*. The* online algorithm with answerer $(\mathcal{A}, \mathcal{O})$ *computes the output sequence* $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle$ *as follows:*

1. *in each step* i*, a query* $r_i \in \{0, 1\}$ *is generated first as a function of previous inputs and advices, i.e.* $r_i = f_r(x_1, \ldots, x_i, r_1 \star \boldsymbol{a_1}, \ldots, r_{i-1} \star \boldsymbol{a_{i-1}})$[3]
2. *then, the output is computed as* $y_i = f(x_1, \ldots, x_i, r_1 \star \boldsymbol{a_1}, \ldots, r_i \star \boldsymbol{a_i})$
   The cost *of the solution is* $C_{(\mathcal{A},\mathcal{O})}(\boldsymbol{x}) = COST(\boldsymbol{y})$*, and the* advice (bit) complexity *is* $B^A_{(\mathcal{A},\mathcal{O})}(\boldsymbol{x}) = \sum_{i=1}^{n} |r_i \star \boldsymbol{a_i}|$

As already mentioned, we are interested in the minimal amount of information the algorithm must get from the oracle, in order to be optimal. For an algorithm $\mathcal{A}$ with an oracle (helper or answerer) $\mathcal{O}$, the communication cost is the worst case bit complexity, amortized per one step:

Definition 5. *Consider an online algorithm* $\mathcal{A}$ *with an oracle* $\mathcal{O}$ *using communication mode* $M \in \{H, A\}$[4]*. The* bit complexity of the algorithm *is*

$$B^M_{(\mathcal{A},\mathcal{O})} = \limsup_{n \mapsto \infty} \max_{|\boldsymbol{x}|=n} \frac{B^M_{(\mathcal{A},\mathcal{O})}(\boldsymbol{x})}{n}$$

The advice complexity of an online problem $\mathcal{P}$ is the minimum bit complexity of an optimal pair $(\mathcal{A}, \mathcal{O})$:

Definition 6. *Consider a problem* $\mathcal{P}$*. The advice* complexity of $\mathcal{P}$ *in communication mode* $M \in \{H, A\}$ *is* $B_M(\mathcal{P}) = \min_{(\mathcal{A},\mathcal{O})} B^M_{(\mathcal{A},\mathcal{O})}$ *where the minimum is taken over all* $(\mathcal{A}, \mathcal{O})$ *such that* $\forall \boldsymbol{x} : C_{(\mathcal{A},\mathcal{O})}(\boldsymbol{x}) = C_{OPT}(\boldsymbol{x})$

We start analyzing the advice complexity with an immediate observation that the answerer model is more restrictive in the following sense:

Claim 1. *For each problem* $\mathcal{P}$*,* $B_H(\mathcal{P}) \leq B_A(\mathcal{P}) \leq 0.92 + B_H(\mathcal{P})$*.*

In the lower bound arguments, we shall use the notion of a *communication pattern*. Informally, a communication pattern is the entire information that the algorithm receives from the oracle. Since the algorithms are deterministic, the number of different communication patterns gives the number of different behaviors of the algorithm on a given input.

[3] The function "$\star$" is defined $c \star \alpha = \begin{cases} \textit{empty string} & \text{if } c = 0 \\ \alpha & \text{otherwise} \end{cases}$.

[4] In the description of communication modes, H stands for helper and A for answerer.

Definition 7 (Communication pattern – helper). *Consider an algorithm with helper. The* communication pattern *is defined as the sequence of advices given at each particular step, i.e.* $\langle \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \rangle$*, where* $\boldsymbol{a}_i$ *is a, possibly empty, binary string.*

Obviously, the input and communication pattern completely determine the behavior of the algorithm.

Lemma 1. *Consider an algorithm with helper, and let the input sequence be of length* $n+1$*. For a fixed* s*, consider only communication patterns in which the helper sends in total at most* s *bits over all* $n+1$ *advices. The number* X *of distinct communication patterns with this property is at most* $\log X \leq s\left(\log(1+\alpha)+1+\frac{1}{\ln 2}\right)+\frac{1}{2}\left[\log\left(1+\frac{1}{\alpha}\right)+\log s\right]+c$ *where* $\alpha=\frac{n}{s}>1$*, and* c *is some constant.*

The situation in the answerer mode is slightly more complicated due to the fact that answers are delivered only when requested.

Definition 8 (Communication pattern – answerer). *For each execution of an algorithm with* q *queries to the answerer, the* communication pattern *is the sequence* $\langle \boldsymbol{a}_{i_1}, \ldots, \boldsymbol{a}_{i_q} \rangle$ *of non-empty answers, where* i_j *is the step in which the* j*'th question was asked.*

The behavior of the algorithm is clearly completely determined by the input, the communication pattern and a mapping that assigns for each $\boldsymbol{a}_{i_j}$ the step j in which the answer was delivered. However, this mapping bears no relevant information: for a given input and communication pattern, the algorithm always receives identical answers, and hence it also asks identical questions. Hence, the behavior of an algorithm with answerer is completely determined by its input and communication pattern.

Lemma 2. *Consider an algorithm with answerer. For a fixed* q*, and* $s \geq q$*, consider only communication patterns, in which the algorithm ask* q *questions, and* s *is the total number of bits in all answers. Then there are* $X=\frac{1}{3}\left(2^{2s+1}+1\right)$ *different communication patterns with this property*[5]*.*

In the rest of the paper we assume that the algorithm knows the length of the input. Indeed, it is always possible to alter the oracle in such a way that it sends the length of the input[6] in the first step. Since there are $O(\log n)$ additional bits sent, the normalized contribution to one request is $O(\log n/n)$ which is asymptotically zero.

3 Paging

Paging and its many variants belong to the classical online problems. The virtual memory of a computer is divided into logical pages. At any time K logical pages

[5] Note that the formula does not depend on q.

[6] In self-delimited form to distinguish it from the possible advice.

can reside in the physical memory. A paging algorithm is the part of the operating system responsible for maintaining the physical memory. If a program requests access to a logical page that is not currently in the physical memory, a *page fault* interrupt occurs and the paging algorithm has to transfer the requested page into physical memory, possibly replacing another one. Formally, we define the paging problem as follows:

Definition 9 (Paging Problem). *The input is a sequence of integers (logical pages)* $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$. *The algorithm maintains a buffer (physical memory)* $B = \{b_1, \ldots, b_K\}$ *of* K *integers. Upon receiving an input* x_i, *if* $x_i \in B$, $y_i = 0$. *If* $x_i \notin B$ *a* page fault *is generated, and the algorithm has to find some* victim b_j, *i.e.* $B := B \setminus \{b_j\} \cup \{x_i\}$, *and* $y_i = b_j$. *The cost of the solution is the number of faults, i.e.* $COST(\boldsymbol{y}) = |\{y_i : y_i > 0\}|$.

It is a well known fact [24] that there is a K-competitive paging algorithm, and that K is the best attainable competitive ratio by any deterministic online algorithm. The optimal offline algorithm is due to [2]. Let us consider the advice complexity of this problem for both helper and answerer modes. We prove that for the helper mode the complexity is between 0.1775 and 0.2056, and for the answerer mode the complexity is between 0.4591 and $0.5 + \varepsilon$ bits per request. Let us first analyze the helper mode. We start with a simple algorithm that uses one bit per request:

Lemma 3. *Consider the* Paging *problem. There is an algorithm* $\mathcal{A}$ *with a helper* $\mathcal{O}$, *such that* $\mathcal{O}$ *sends an advice of exactly one bit each step.*

Proof. Consider an input sequence $\boldsymbol{x}$, and an optimal offline algorithm OPT processing it. In each step of OPT, call a page currently in the buffer *active*, if it will be requested again, before OPT replaces it by some other page. We design $\mathcal{A}$ such that in each step i, the set of OPT's active pages will be in B, and $\mathcal{A}$ will maintain with each page an *active* flag identifying this subset. If $\mathcal{A}$ gets an input x_i that causes a page fault, some passive page is replaced by x_i. Moreover, $\mathcal{A}$ gets with each input also one bit from the helper telling whether x_i is active for OPT. Since the set of active pages is the same for OPT and $\mathcal{A}$, it is immediate that $\mathcal{A}$ generates the same sequence of page faults. □

Now we are going to further reduce the advice complexity. The algorithm will still receive the required one bit for every input, however, it is possible to encode the bits in a more efficient way using larger strings as advice:

Lemma 4. *For* r *large enough, the helper can communicate a binary string of length* αr *using* r *bits over a period of* αr *steps, where* $\alpha \approx 4.863876183$.

Theorem 1. $B_H(\text{Paging}(K)) \leq \frac{1}{\alpha}$, *where* $\alpha \approx 4.863876183$.

On the lower bound side, we can prove the following:

Theorem 2. *For every fixed* K, *there is a constant* $\alpha_K < 20.742$ *such that* $B_H(\text{Paging}(2K)) \geq \frac{1}{\alpha_K}$. *Moreover,* α_K *is a decreasing function in* K *and* $\lim_{K \mapsto \infty} \alpha_K \approx 5.632423693$

Sketch of the proof. We shall consider a particular subset of input sequences $\boldsymbol{x} = \{x_k\}_{k=1}^{K(2+3i)}$ for some i. Each input sequence consists of the sequence $S_0 = \langle 1, 2, \ldots, 2K \rangle$ followed by i *frames*, each of length $3K$, where the jth frame has the form $D_j \cdot \langle d_j \rangle \cdot S_j$. The first part of each frame, D_j is of length $K-1$ and contains unused pages that generate page faults, the next request d_j is again an unused page. The last part, S_j is a sequence of length $2K$ consisting of any subsequence of $S_{j-1} \cdot D_j$ of length $2K-1$, followed by d_j.

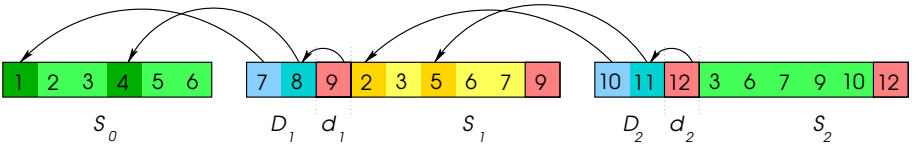


Fig. 2. An example of first two frames for $K = 3$, i.e. with buffer of size 6. The arrows indicate which pages are replaced during faults.

It is easy to see that no optimal algorithm can generate a page fault in S_j, which means that at the beginning of S_j, the content of the buffer of any optimal algorithm is uniquely determined.

Since any optimal algorithm needs a different communication pattern for each input, a simple calculation shows that there must be at least $Y = \left\lceil \frac{2}{3}\binom{3K}{K} \right\rceil^i$ different communication patterns. However, using Lemma 1, we get that there are at most X different communication patterns of length $n+1$ using at most s bits. Comparing these two numbers concludes after some calculations the proof. □

Let us proceed now with the analysis of the answerer mode. First, we give an upper bound by refining Lemma 3:

Theorem 3. *For each* $\varepsilon > 0$, $B_A(\textsc{Paging}(K)) \leq \frac{1}{2} + \varepsilon$

To conclude this section, the same technique as used in Theorem 2 can be employed to deliver the corresponding lower bound:

Theorem 4. $B_A(\textsc{Paging}(2K)) \geq 0.4591 - O\left(\frac{\log K}{K}\right)$

4 Diff-Serv

DiffServ is another problem widely studied using competitive analysis (see [8], [19] and references therein). The setting involves a server processing an incoming stream of packets of various values. If the processing speed of the server is slower than the arrival rate, some packets must be dropped, ideally those least valuable. For our purposes, following [19], the packets arrive in discrete time steps. In each step a number of packets can arrive, one packet can be processed, and at most K packets can be stored in a buffer. Moreover, it is required that the packets are processed in FIFO manner. The formal definition is as follows:

Definition 10 (DIFF-SERV problem). *Consider a sequence of items* $\langle p_1, \ldots, p_m \rangle$*, partitioned into a series of subsequences, called requests. The input is the sequence of requests* $\boldsymbol{x} = \langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \rangle$*, where each* $\boldsymbol{x}_i = \langle p_{j_{i-1}+1}, \ldots, p_{j_i} \rangle$ *is a (possibly empty) request. Each item* p_i *has a value* $v(p_i)$*. In each step* i*, the algorithm maintains an ordered buffer* $B_i = \langle b_1, \ldots, b_K \rangle$ *of* K *items. Upon receiving a request sequence* $\boldsymbol{x}_i$*, the algorithm* discards *some elements from the sequence* $B_i \cdot \boldsymbol{x}_i$*, keeping some subsequence* $B'_i \preceq B_i \cdot \boldsymbol{x}_i$ *of length at most* $K+1$*. The first item (if* B'_i *is nonempty) of* B'_i *is* submitted*, and the remainder of the sequence forms the new buffer, i.e.* $B'_i = y_i \cdot B_{i+1}$*. The process ends if there are no more requests*[7] *and the buffer is empty.*

The cost of the solution is the sum of the values of all submitted elements, i.e. $COST(\boldsymbol{y}) = \sum_{i>0} v(y_i)$.

For the remainder of this section we shall consider only the case of two distinct item values; we shall refer to them as heavy and light items. Without loss of generality we may assume that each request contains at most $K+1$ heavy items. Lotker and Patt-Shamir [19] presented an optimal greedy offline algorithm. We first present another optimal offline algorithm, and then show how to transform it to an online algorithm with a helper.

Let us start with a simple greedy algorithm that never discards more items than necessary (Algorithm 1 without line 4). This algorithm is not optimal in situations where it is favorable to discard leading light items even if the buffer would not be filled[8]. These situations, however, can easily be recognized:

Definition 11. *Consider a buffer* B *at time* t_0 *and the remainder* $\{\boldsymbol{x}_{t_0+i}\}_{i=1}^{n}$ *of the input sequence. Let* a_0 *be the number of heavy elements in* B *(before* $\boldsymbol{x}_{t_0+1}$ *has arrived), and* $a_i \leq K+1$ *be the number of heavy elements in* x_{t_0+i}*. The remainder of sequence* $\boldsymbol{x}$ *is called* critical *(w.r.t.* B*), if there exists* $t > 0$ *such that* $\sum_{i=0}^{t} a_i \geq K + t$*, and for each* t' *such that* $0 < t' \leq t$ *it holds* $\sum_{i=0}^{t'} a_i \geq t'$.

Informally, an input sequecne is critical w.r.t. an initial buffer if the buffer gradually fills with heavy items even if the algorithm submits a heavy item in each step. Our algorithm processes requests sequentially. Each request is processed as shown in Algorithm 1, and it can be proven that this algorithm is optimal.

Now we turn this offline algorithm into an online algorithm with helper. We are going to simulate Algorithm 1 with an algorithm and a helper. The only place where the algorithm needs information about the future is on line 4, where the algorithm tests the criticality of the input. Clearly, one bit per request (indicating whether the input is critical or not) is sufficient to achieve optimality. However,

[7] In this case some number of virtual empty requests is added until the buffer is emptied.

[8] Consider a situation with a buffer of size 3 containing one light and two heavy items. If there are no more requests, the best solution is to submit all three of them in the next three steps. However, if there is another request coming, containing two heavy items, the best solution is to discard the light one and submit heavy items in the next four steps.

Algorithm 1. Processing of a request $\boldsymbol{x}_i$ with a buffer B

1: $B' \leftarrow B \cdot \boldsymbol{x}_i$
2: starting from left, discard light items from B' until $|B'| = K + 1$ or there are no light items left.
3: **if** $|B'| > K + 1$ **then** discard last $|B'| - K - 1$ (heavy) items
4: **if** *the remainder of the input sequence is critical and there are some heavy items in* B' **then** discard leading light items from B'
5: submit the first item of B' (if exists)
6: $B \leftarrow$ remainder of B'

we show that situation in which a bit must be sent can occur at most once in every $K + 1$ steps.

Theorem 5. $B_H(\textsc{DiffServ}(K)) \leq \frac{1}{K+1}$

Using a technique similar to the proof of Theorem 2, we can show the following:

Theorem 6. *For* $K \geq 4$ *it holds* $B_H(\textsc{DiffServ}(K)) \geq \frac{1}{\gamma_K \cdot K}$, *where* $\gamma_K \leq 6.13$ *and* $\lim_{K \mapsto \infty} \gamma_K = 1$

In a similar fashion, the following results can be shown for the answerer mode:

Theorem 7. $B_A(\textsc{DiffServ}(K)) \leq \frac{1+\log(K+1)}{K+1}$

Theorem 8. *For each fixed* $K \geq 4$ *there exists a* $\gamma_K \leq 3.822$ *such that* $B_A(\textsc{DiffServ}(K)) \geq \frac{\log(K+2)}{\gamma_K(K+2)}$ *Moreover* $\lim_{K \mapsto \infty} \gamma_K = 2$.

5 Conclusion

We have proposed a new way to evaluate online problems, based on the communication complexity. While the competitive analysis is an algorithmic measure evaluating the output quality degradation incurred by the requirements to produce the output online, our measure is a structural one quantifying the amount of additional information about the input needed to produce optimal output in an online fashion. The study of the relation between those two measures can lead to a deeper understanding of the nature of online problems. We have shown that there are problems like Paging and DiffServ where the advice complexity (in the helper mode) is proportional to the competitive ratio. On the other hand, there are problems with simple structure like SkiRental [17], which has competitive ratio $2 - \varepsilon$, but a single bit of information is sufficient to solve the problem optimally (i.e. it has zero advice complexity).

Studying advice complexity of a problem can lead to exposure of the critical decisions to be made (like in Algorithm 1 for DiffServ) and subsequently to better understanding of the problem and possibly more efficient algorithms. Moreover, we expect that in certain situations involving cooperating devices of uneven computational power communicating over a costly medium (as e.g. in sensor networks), the advice complexity might be of practical interest.

The proposed topic presents a number of intriguing open questions. Is it, for example, possible to characterize a class of problems where the competitive ratio is proportional to the advice complexity? Another whole research area is to study the tradeoff between the amount of communicated information and the achieved competitive ratio.

There is also a number of variations of the model that could be investigated. One potential modification would be to limit the size of advice given in one step. In our model this size is unbounded, and this fact is heavily relied upon (sending the length of the input in one step). However, for modelling potentially infinite inputs it would be more appealing to limit the size of advice to be independent of the input size.

References

1. Albers, S.: Online algorithms: A survey. Mathematical Programming 97, 3–26 (2003)
2. Belady, L.A.: A study of replacement algorithms for virtual storage computers. IBM Systems Journal 5, 78–101 (1966)
3. Ben-David, S., Borodin, A.: A new measure for the study of on-line algorithms. Algorithmica 11(1), 73–91 (1994)
4. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
5. Borodin, A., Irani, S., Raghavan, P., Schieber, B.: Competitive paging with locality of reference. In: Proc. 23rd Annual ACM Symp. on Theory of Computing, pp. 249–259 (1991)
6. Boyar, J., Favrholdt, L.M.: The Relative Worst Order Ratio for Online Algorithms. In: Petreschi, R., Persiano, G., Silvestri, R. (eds.) CIAC 2003. LNCS, vol. 2653, pp. 58–69. Springer, Heidelberg (2003)
7. Dobrev, S., Královič, R., Pardubská, D.: How Much Information About the Future is Needed?, Technical report TR-2007-007, Faculty of Mathematics, Physics, and Informatics, Comenius University, Bratislava, http://kedrigern.dcs.fmph.uniba.sk/reports/display.php?id=22
8. Englert, M., Westermann, M.: Lower and Upper Bounds on FIFO Buffer Management in QoS Switches. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 352–363. Springer, Heidelberg (2006)
9. Fiat, A., Karp, R.M., Luby, M., McGeoch, L.A., Sleator, D.D., Young, N.E.: Competitive Paging Algorithms. J. Algorithms 12, 685–699 (1991)
10. Fraigniaud, P., Gavoille, C., Ilcinkas, D., Pelc, A.: Distributed computing with advice: information sensitivity of graph coloring. In: Arge, L., et al. (eds.) ICALP 2007. LNCS, vol. 4596, Springer, Heidelberg (2007)
11. Fraigniaud, P., Ilcinkas, D., Pelc, A.: Oracle size: a new measure of difficulty for communication problems. In: PODC 2006. Proc. 25th Ann. ACM Symposium on Principles of Distributed Computing, pp. 179–187 (2006)
12. Graham, R.L.: Bounds for Certain Multiprocessing Anomalies. Bell Systems Technical Journal 45, 1563–1581 (1966)
13. Irany, S., Karlin, A.R.: Online Computation. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-Hard Problems, pp. 521–564. PWS Publishing Company (1997)

14. Irani, S., Karlin, A.R., Phillips, S.: Strongly competitive algorithms for paging with locality of reference. In: Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms, pp. 228–236 (1992)
15. Kalyanasundaram, B., Pruhs, K.: Speed is as Powerful as Clairvoyance. In: IEEE Symposium on Foundations of Computer Science, pp. 214–221 (1995)
16. Karlin, A.R., Manasse, M.S., Rudolph, L., Sleator, D.D.: Competitive Snoopy Caching. Algorithmica 3, 79–119 (1988)
17. Karp, R.: On-line algorithms versus off-line algorithms: how much is it worth to know the future? In: Proc. IFIP 12th World Computer Congress, vol. 1, pp. 416–429 (1992)
18. Koutsoupias, E., Papadimitriou, C.H.: Beyond competitive analysis. In: Proc. 34th Annual Symp. on Foundations of Computer Science, pp. 394–400 (1994)
19. Lotker, Z., Patt-Shamir, B.: Nearly Optimal FIFO Buffer Management for DiffServ. In: PODC 2002, pp. 134–143 (2002)
20. Manasse, M.M., McGeoch, L.A., Sleator, D.D.: Competitive Algorithms for Online Problems. In: Proc. 20th Annual Symposium on the Theory of Computing, pp. 322–333 (1988)
21. Philips, C.A., Stein, C., Torng, E., Wein, J.: Optimal Time-Critical Scheduling via Resource Augmentation. In: Proc. 29th Annual ACM Symp on the Theory of Computing, pp. 140–149 (1997)
22. O'Reilly, U.M., Santoro, N.: The Expressiveness of Silence: Tight Bounds for Synchronous Communication of Information Using Bits and Silence. In: Mayr, E.W. (ed.) WG 1992. LNCS, vol. 657, pp. 321–332. Springer, Heidelberg (1993)
23. Raghavan, P.: A statistical adversary for on-line algorithms. In: On-Line Algorithms, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 79–83 (1991)
24. Sleator, D.D., Tarjan, R.E.: Amortized Efficiency of Update and Paging Rules. Comm. of the ACM 28(2), 202–208 (1985)
25. Torng, E.: A Unified Analysis of Paging and Caching. Algorithmica 20, 175–200 (1998)
26. Young, N.: The k-server dual and loose competitiveness for paging. Algorithmica 11, 525–541 (1994)

On Compiling Structured Interactive Programs with Registers and Voices*

Cezara Dragoi** and Gheorghe Stefanescu

Faculty of Mathematics and Computer Science, University of Bucharest
Str. Academiei 14, Bucharest, Romania 010014
{cdragoi,gheorghe}@funinf.cs.unibuc.ro

Abstract. A model (consisting of *rv-systems*), a core programming language (for developing *rv-programs*), several specification and analysis techniques appropriate for modeling, programming and reasoning about interactive computing systems have been recently introduced by Stefanescu using register machines and space-time duality, see [13]. In [3,4,5,6] the authors have have introduced and studied structured programming techniques for rv-systems.

The aim of the present paper is to define a scenario-based operational semantics for structured rv-programs and to offer a translation from structured rv-programs to rv-programs. The main technical result states that the translation is correct. This is part of an effort to get a running environment for structured rv-programs built up on top of rv-programs.

Keywords: interactive systems, structured rv-systems, programming languages, operational semantics, registers and voices, compiler correctness.

1 Introduction

Interactive computation has a long tradition and there are many successful approaches to deal with the intricate aspects of this type of computation, see [1,2,7,8,15], to mention just a few references from a very rich literature. However, a general simple and unifying model for interactive computation, extending the classical, popular imperative programming paradigm, is still to be find.

A model (consisting of *rv-systems*), a core programming language (for developing *rv-programs*), several specification and analysis techniques appropriate for modeling, programming and reasoning about interactive computing systems have been recently introduced by Stefanescu using register machines and space-time duality, see [13]. One of the key features of the model is the introduction of high-level temporal data structures. Actually, having high level temporal data on interaction interfaces is of crucial importance in getting a compositional model

* This research was partially supported by the Romanian Ministry of Education and Research (PNCDI-II Program 4, Project D1/1052/18.09.2007: *GlobalComp - Models, semantics, logics and technologies for global computing*).

** Current address: LIAFA, Universite Paris Diderot - Paris 7, France.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 259–270, 2008.

for interactive systems, a goal not always easy to achieve (recall the difficulties in getting a compositional semantics for data-flow networks).

In [3,4,5] the authors have introduced and studied structured programming techniques for rv-systems. In [6], a kernel programming language for interactive systems AGAPIA is introduced and its typing system is studied. See [14,10] for more information and results on rv-systems and their verification.

The aim of the present paper is to offer a translation from structured rv-programs to rv-programs. This is part of an effort to get a running environment for structured rv-programs built up on top of rv-programs. The main technical contribution of the paper is a proof of the translation correctness.

The paper is organized as follows. It start with a presentation of spatial and temporal data, of spatio-temporal relational specifications, and of scenarios and operations on scenarios. Next, after a brief recall of rv-programs, structured rv-programs are introduced. Then, the scenario-based operational semantics is presented. After that, the translation from structured rv-programs to rv-programs is defined and finally, the statement on translation correctness is included. (The proof of the translation correctness, developed in the long version of the paper, is rather tricky, based on a good understanding of rv-program transformations.)

2 Scenarios

In this section we briefly present temporal data, spatio-temporal specifications, grids, scenarios, and operations on scenarios.

2.1 Specifications and Scenarios

Spatio-temporal specifications. To handle spatial data, common data structures and their natural representations in memory are used. For the temporal data, we use streams: a *stream* is a sequence of data ordered in time and is denoted as $a_0 \frown a_1 \frown \ldots$, where $a_0, a_1, \ldots$ are its data at time $0, 1, \ldots$, respectively. Typically, a stream results by observing the data transmitted along a channel: it exhibits a datum (corresponding to the channel type) at each clock cycle.

A *voice* is defined as the time-dual of a register: *A voice is a temporal data structure that holds a natural number. It can be used ("heard") at various locations. At each location it displays a particular value.*

Voices may be implemented on top of a stream in a similar way registers are implemented on top of a Turing tape, for instance specifying their starting time and their length. Most of usual data structures have natural temporal representations. Examples include timed booleans, timed integers, timed arrays of timed integers, etc.

For an interactive system using no more complex data than registers and voices, a *spatio-temporal specification* $S : (m, p) \to (n, q)$ is a relation $S \subseteq (\mathbb{N}^m \times \mathbb{N}^p) \times (\mathbb{N}^n \times \mathbb{N}^q)$, where m (resp. p) is the number of input voices (resp. registers) and n (resp. q) is the number of output voices (resp. registers). It may be defined as a relation between tuples, written as $\langle v \mid r \rangle \mapsto \langle v' \mid r' \rangle$, where v, v' (resp. r, r') are tuples of voices (resp. registers).

Specifications may be composed horizontally and vertically, as long as their types agree; e.g., for two specifications $S_1 : (m_1, p_1) \to (n_1, q_1)$ and $S_2 : (m_2, p_2) \to (n_2, q_2)$ the *horizontal composition* $S_1 \triangleright S_2$ is defined only if $n_1 = m_2$ and the type of $S_1 \triangleright S_2$ is $(m_1, p_1 + p_2) \to (n_2, q_1 + q_2)$.

Grids and scenarios. A *grid* is a *rectangular* two-dimensional area filled in with letters of a given alphabet. An example of a grid is presented in Fig. 1(a). In our standard interpretation, the columns correspond to processes, the top-to-bottom order describing their progress in time. The left-to-right order corresponds to process interaction in a *nonblocking message passing discipline*: a process sends a message to the right, then it resumes its execution. (See [9] for related studies.)

A *scenario* is a grid enriched with data around each letter. The data may have various interpretation: they either represent control/interaction information, or current data of the variables, or both. Fig. 1 illustrates the first case, Fig. 1(c) the last case, and Fig. 1(d) the middle case. Notice that the scenario from Fig. 1(d) is similar to that in (c), but the control/interaction labels A,B,C,1,2,3 are omitted. The scenarios of a rv-program look like in (c), while those of structured rv-programs as in (d) - there are not labels in the latter case.

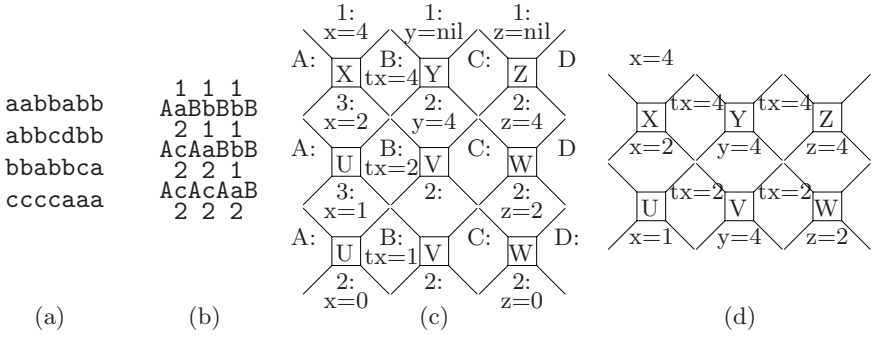


Fig. 1. A grid (a), an abstract scenario (b), and concrete scenarios (c,d)

The type of a scenario interface of type (d) is represented as $t_1; t_2; \ldots; t_k$, where each t_k is a tuple of simple types used in the scenario cells. An empty tuple is also written 0 or *nil* and can be freely inserted to or omitted form such descriptions. The type of a scenario f is specified by the notation $f : \langle w|n\rangle \to \langle e|s\rangle$, where $w/n/e/s$ represent the types of its west/north/east/south borders. For the example in Fig. 1(d), the type is $\langle nil; nil|sn; nil; nil\rangle \to \langle nil; nil|sn; sn; sn\rangle$, where sn denotes the spatial integer type.

2.2 Operations with Scenarios

We say two scenario interfaces $t = t_1; t_2; \ldots; t_k$ and $t' = t'_1; t'_2; \ldots; t'_{k'}$ are *equal* if $k = k'$ and the types and the values of each pair t_i, t'_i are equal. Two interfaces

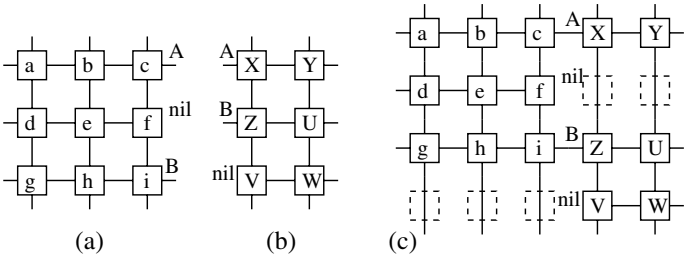


Fig. 2. Horizontal composition of scenarios

are *equal up to the insertion of nil elements*, written $t =_n t'$, if one can insert *nil* elements into these interfaces such that the resulting interfaces are equal.

We denote by $Id_{m,p} : \langle m|p\rangle \to \langle m|p\rangle$ the identity constant, i.e., the temporal/spatial output is equal to the temporal/spatial input, respectively.

Horizontal composition: Suppose we start with two scenarios $f_i : \langle w_i|n_i\rangle \to \langle e_i|s_i\rangle, i = 1, 2$. Their *horizontal composition* $f_1 \triangleright f_2$ is defined only if $e_1 =_n w_2$. For each inserted *nil* element in an interface, a dummy row is inserted in the corresponding scenario, resulting a scenario $\overline{f_i}$. After these transformations, the result is obtained putting $\overline{f_1}$ on left of $\overline{f_2}$. (Notice that $\overline{f_1} : \langle w_1|n_1\rangle \to \langle t|s_1\rangle$ and $\overline{f_2} : \langle t|n_2\rangle \to \langle e_2|s_2\rangle$, where t is the resulting common interface.) The result, $f_1 \triangleright f_2 : \langle w_1|n_1; n_2\rangle \to \langle e_2|s_1; s_2\rangle$, is unique up to insertion or deletion of dummy rows. See Fig. 2 and Fig. 3(b). Its identities are $Id_{m,0}$.

Vertical composition The definition of *vertical composition* $f_1 \cdot f_2$ is similar, but now $s_1 =_n n_2$. For each inserted *nil* element, a dummy column is inserted in the corresponding scenario, resulting a scenario $\overline{f_i}$. The result, $f_1 \cdot f_2 : \langle w_1; w_2|n_1\rangle \to \langle e_1; e_2|s_2\rangle$, is obtained putting $\overline{f_1}$ on top of $\overline{f_2}$. See Fig. 3(a). Its identities are $Id_{0,m}$.

Constants: Except for the already defined identities I, additional constants may be used. Some of them may be found in Fig. 3: A recorder R (2nd cell in the 1st row of (c)), a speaker S (1st cell in the 2nd row of (c)), an empty cell Λ (3rd cell in the 1st row of (c)), etc.

Diagonal composition: The *diagonal composition* $f_1 \bullet f_2$ is defined only if $e_1 =_n w_2$ and $s_1 =_n n_2$. It is a derived operation defined by

$$f_1 \bullet f_2 = (f_1 \triangleright R_1 \triangleright \Lambda_1) \cdot (S_2 \triangleright Id \triangleright R_2) \cdot (\Lambda_2 \triangleright S_1 \triangleright f_2)$$

for appropriate constants R, S, Id, Λ. See Fig. 3(c). In this case $R_1 : \langle t|\ \rangle \to \langle\ |t\rangle$, $S_1 : \langle\ |t\rangle \to \langle t|\ \rangle$, $Id : \langle u|t\rangle \to \langle u|t\rangle$, $R_2 : \langle u|\ \rangle \to \langle\ |u\rangle$, $S_2 : \langle\ |u\rangle \to \langle u|\ \rangle$, where

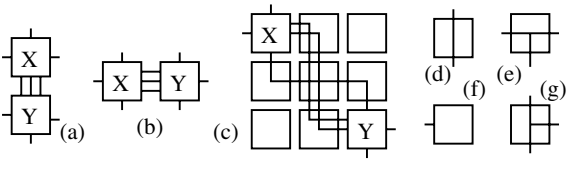


Fig. 3. Operations on scenarios

t (resp. u) is a common representation for e_1 and w_2 (resp. s_1 and n_2) obtained inserting *nil* elements. Its identities are $Id_{m,n}$.

We extend the definitions of the scenario compositions to set of scenarios. Given two sets of scenarios, A and B, we define the *horizontal composition*

$$A \triangleright B = \{f_a \triangleright f_b | \text{ for } f_a \in A \text{ and } f_b \in B\}.$$

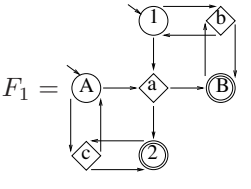
The *vertical composition* $A \cdot B$ and the *diagonal composition* $A \bullet B$ on set of scenarios A, B are similarly defined.

3 Rv-Programs

In this section we briefly describe rv-programs (interactive programs with registers and voices); see [13,14] for more details.

Finite interactive systems. A *finite interactive system* (shortly FIS) is a finite hyper-graph with two types of vertices and one type of (hyper) edges: the first type of vertices is for *states* (labeled by numbers), the second is for *classes* (labeled by capital letters) and the edges/transitions are labeled by letters denoting the atoms of the grids; each transition has two incoming arrows (one from a class and the other from a state), and two outgoing arrows (one to a class and the other to a state). Some classes/states may be *initial* (indicated by small incoming arrows) or *final* (indicated by double circles); see, e.g., [12,13]. An example is shown below.

For the *parsing procedure*, given a FIS F and a grid w, insert initial states/classes at the north/west border of w and parse the grid completing the scenario according to the FIS transitions; if the grid is fully parsed and the south/east border contains final states/classes only, then the grid w is recognized by F. The *language* of F is the set of its recognized grids. A FIS F_1 and a parsing accepting $\begin{smallmatrix}\texttt{abb}\\ \texttt{cab}\\ \texttt{cca}\end{smallmatrix}$ are shown below.




```
 1 1 1    1 1 1          1 1 1
Aa b b   AaBb b   ···  AaBbBbB
           2                2 1 1
Ac a b   Ac a b       AcAaBbB
                          2 2 1
Ac c a   Ac c a       AcAcAaB
                          2 2 2
```

Interactive programs with registers and voices. An *rv-system* (*interactive system with registers and voices*) is a FIS enriched with: (i) registers associated to its states and voices associated to its classes; and (ii) appropriate spatio-temporal transformations for actions.

We study programmable rv-systems specified using *rv-programs*. An example of rv-program is presented in Fig. 4. A computation is described by a scenario mixing control/interaction labels and data; see Fig. 1(c) for an example.

Syntax of rv-programs. A program is a collection of modules. A module has a name and 4 areas: (1) The top-left part contains a pair of labels specifying the interaction/control (class/state) coordinates where the module may be applied.

in: A,1; out: D,2

X::

(A,1)	x : sInt

	tx : tInt;
	tx = x;
	x = x/2;
	goto [B,3];

W::

(C,2)	z : sInt
tx : tInt	z = z - tx; goto [D,2];

U::

(A,3)	x : sInt

	tx : tInt;
	tx = x; x = x - 1;
	if (x > 0) {goto [B,3]}
	else {goto [B,2];}

Y::

(B,1)	y : sInt
tx : tInt	y = tx; goto [C,2];

V::

(B,2)	y : sInt

tx : tInt	if(y%tx != 0)
	tx = 0;
	goto [C,2];

Z::

(C,1)	z : sInt
tx : tInt	z = tx; goto [D,2];

Fig. 4. The rv-program **Perfect** (for perfect numbers)

(2)-(3) The top-right (resp. bottom-left) part specifies the spatial (resp. temporal) input variables. (4) The bottom-right part is the body of the module, including C-like code. The exit from the module is specified by a `goto` statement. A statement, say `goto [B,3]`, indicates that: (i) the data of the spatial variables in the current module will be used in a next module with control state `3`; and (ii) the data of the temporal variables in the current module will be used for the interaction interface of a new module with interaction label `B`.

Operational and denotational semantics of rv-programs. The *operational semantics* is given in terms of scenarios. Scenarios are built up with the following procedure, described using the scenario in Fig. 1(c) for the rv-program **Perfect**:

(1) Each cell has a module name as label.
(2) In the areas around a cell we show how variables are modified.
(3) In a current cell, the values of spatial variables are obtained going vertically up and collecting the last updated values.
(4) Similarly, the full information on temporal variables in a current cell is obtained collecting their last updated values going horizontally on left.
(5) The first column has input classes and particular values for their temporal variables; the first row has input states and particular values for their spatial variables.
(6) The computation in a cell α is done as follows: (i) Take a module β of the program bearing the class label of the left neighboring area of α and the state label of the top neighboring area of α. (ii) Follow the code in β using the spatial and the temporal variables of α with their current values. (iii) If the local execution of β is finished with a `goto` $[\Gamma, \gamma]$ statement, then the label of the right neighboring area of α is set to Γ and the label of the bottom neighboring area of α is set to γ. (iv) Insert the values of the temporal variables updated by β in the right neighboring area of α and the values of the spatial variables updated by β in the bottom neighboring area of α.
(7) A *partial scenario* (for an rv-program) is a scenario built up using the above rules; it is a *complete scenario* if the bottom row has only final states and the rightmost column has only final classes.

The scenario in Fig. 1(c) is a complete scenario for the rv-program **Perfect**.

The *input-output denotation* of an rv-program is the relation between the input data on the north/west borders and output data on the south/east borders of the program scenarios.

Notice that a global scoping rule is implicitly used here: once defined, a variable is always available. It is also possible to introduce rv-programs obeying a stronger typing discipline, where each module comes with an explicit type at each border. This option is actually used for structured programs to be introduced in the next section.

4 Structured rv-Programs

The rv-programs, briefly presented in the previous section, resemble flowcharts and assembly languages: one freely uses `goto` statements, with both temporal and spatial labels. The aim of this section is to introduce structured programming techniques on top of rv-programs. The resulting structured rv-programs may be described directly, from scratch. The lower level of rv-programs is used as a target language for compiling.

4.1 Syntax, Examples

The syntax of structured rv-programs. It is given by the BNF grammar

$$P ::= X \mid if(C)then\{P\}else\{P\} \mid P\%P \mid P\#P \mid P\$P$$
$$\mid while\_t(C)\{P\} \mid while\_s(C)\{P\} \mid while\_st(C)\{P\}$$
$$X ::= module\{listen\, t\_vars\}\{read\, s\_vars\}\{\, code;\, \}\{speak\, t\_vars\}\{write\, s\_vars\}$$

Structured rv-programs use modules X as their basic blocks. On top of them, larger programs are built up by "if" and composition and iteration constructs for the vertical (or temporal), the horizontal (or spatial), and the diagonal (or spatio-temporal) directions, i.e., $(\%, while\_t)/(\#, while\_s)/(\$, while\_st)$. These statements aim to capture at the program level the corresponding operations on scenarios.

Examples. We include a simple, but rather general example of structured rv-program to give a clue to the reader on the naturalness and the expressiveness of the language. More examples may be found, e.g., in [3,5,10].

A structured rv-program for a ring termination detection protocol is presented in [5]; except for the details on `I1,I2,R`, it has the following format

```
P :: [I1# for_s(tid=0;tid<tn;tid++){I2}#] $
     [while_st(!(token.col==white && token.pos==0)){
     for_s(tid=0;tid<tn;tid++){R}}]
```

It starts with an initialization step where processes are created and inserted into the ring. Next, a diagonal iteration takes places where, in each step, the processes do their jobs and interact horizontally by passing a message list from one process to the next, in the order from process 0 to process `tn-1`. At the

end of an iteration, if the guard condition is fulfilled, a new iteration takes place where the message list of process `tn-1` is passed to process 0 and all processes continue the execution from their last memory states.

Such a program is rather generic and may be used for many other problems like n-player games, 8-queen problem, implementations of OO-systems based on message passing communication, etc.

A dynamic case where processes may freely join or leave the ring may be easily specified in a slightly extended context (see, e.g., [10]).

4.2 Operational Semantics

The operational semantics

$$|\ \ | : \text{Structured rv-programs} \to \text{Scenarios}$$

associates to each program the set of its possible running scenarios.

The type of a program P, which is denoted by $P : \langle w(P)|n(P)\rangle \to \langle e(P)|s(P)\rangle$, indicates the types at its west, north, east, and south borders. On each side, the type may be quite complex, including sets of possible instantiations — see, e.g., the types for AGAPIA interfaces [6]. We use the convention to separate by "," the data from within a module and by ";" the data coming from different modules. These convention refers to both, spatial data (coming from different processes) and temporal data (coming from different transactions).

Two interface types *match* if they have a nonempty intersection.

Modules. The modules are the starting blocks for building structured rv-programs. The `listen (read)` instruction is used to get the temporal (spatial) input and the `speak (write)` instruction to return the temporal (spatial) output. The `code` consists in simple instructions as in the C code. No distinction between temporal and spatial variables is made within a module.

A scenario for a module consists of a unique cell, with particular data on the borders, and such that the output data are obtained from the input data applying the module code.

Composition. Due to their two dimensional structure, programs may be composed horizontally and vertically, as long as their types on the connecting interfaces agree. They can also be composed diagonally by mixing the horizontal and vertical compositions.

Suppose two programs $P_i : \langle w_i|n_i\rangle \to \langle e_i|s_i\rangle$, $i = 1, 2$ are given. We define the following composition operators.

Horizontal composition: $P_1\#P_2$ is defined if the interfaces e_1 and w_2 match. The type of the composite is $\langle w_1|n_1; n_2\rangle \to \langle e_2|s_1; s_2\rangle$. A scenario for $P_1\#P_2$ is a horizontal composition of a scenario in P_1 and a scenario in P_2, formally $|P1\#P2| = |P1| \triangleright |P2|$.

Vertical composition: Similarly, $|P_1\%P_2| = |P1| \cdot |P2|$.

Diagonal composition: $P_1\$P_2$ connects the east border of P_1 to the west border of P_2 and the south border of P_1 to the north border of P_2. It is defined if

each pair of interfaces e_1, w_2 and s_1, n_2 matches. The type of the composite is $\langle w_1|n_1\rangle \to \langle e_2|s_2\rangle$. A scenario for $P_1\$P_2$ is a diagonal composition of a scenario in P_1 and a scenario in P_2, formally $|P1\$P2| = |P1| \bullet |P2|$.

If. Given two programs $P_i : \langle w_i|n_i\rangle \to \langle e_i|s_i\rangle$, $i = 1, 2$, a new program $Q = if\ (C)\ then\ P_1\ else\ P_2$ is constructed, for a condition C involving both, the temporal variables in $w_1 \cap w_2$ and the spatial variables in $n_1 \cap n_2$. The type of the result is $Q : \langle w_1 \cup w_2|n_1 \cup n_2\rangle \to \langle e_1 \cup e_2|s_1 \cup s_2\rangle$.

A scenario for Q is a scenario of P_1 if the data on west and north borders of the scenario satisfy condition C, otherwise is a scenario of P_2.

While. We have introduced three types of while statements, each being the iteration of a corresponding composition operation.

Temporal while: For a program $P : \langle w|n\rangle \to \langle e|s\rangle$, the statement $while\_t\ (C)\{P\}$ is defined if the interfaces n and s match and C is a condition on the spatial variables in $n \cap s$. The type of the result is $\langle (w;)^*|n \cup s\rangle \to \langle (e;)^*|n \cup s\rangle$. (When the body program P of a temporal while has dummy temporal interfaces, the temporal while coincides with the while from imperative programming languages.)

A scenario for $while\_t\ (C)\{P\}$ is either an identity (if C is false), or a repeated vertical composition $f_1 \cdot f_2 \cdot \ldots \cdot f_k$ of scenarios for P, such that the north border of each f_i satisfies C, while the south border of f_k does not satisfy C.

Spatial while: $while\_s\ (C)\{P\}$ is similar.

Spatio-temporal while: If $P : \langle w|n\rangle \to \langle e|s\rangle$, the statement $while\_st\ (C)\{P\}$ is defined if each pair of interfaces w, e and n, s matches and C is a condition on the temporal variables in $w \cap e$ and the spatial variables in $n \cap s$. The type of the result is $\langle w \cup e|n \cup s\rangle \to \langle w \cup e|n \cup s\rangle$.

A scenario for $while\_st\ (C)\{P\}$ is either an identity (if C is false), or a repeated diagonal composition $f_1 \bullet f_2 \bullet \ldots \bullet f_k$ of scenarios for P, such that the west and north border of each f_i satisfies C, while the east and south border of f_k does not satisfy C.

5 The Translation

In this section we describe a translation from structured rv-programs to rv-programs. As we will show later, the translation is correct with respect to the input-output semantics. Moreover, it is weakly correct with respect to the operational semantics, i.e., under mild scenario transformations, the set of running scenarios is preserved.

A transformation on rv-programs

Lemma 1. *(i) For each rv-program P there is an equivalent rv-program P' where all initial/final states/classes only occur on one border of the scenarios, and never inside. (ii) Moreover, for each border, one can manage to have a unique state/class for each interface type of the scenario cells.*

The translation: general form. The translation Tr is done in three steps: $Tr(\bullet) = Tr3(Tr2(Tr1(\bullet)))$. First, we define a function $Tr1$ from structured rv-programs to rv-programs. The next two functions act on rv-programs transforming them into a canonical form. $Tr2$, based on Lemma 1(i), renames the labels such that the initial/final states/classes only occur on one border of the scenarios, and never inside. $Tr3$, based on Lemma 1(ii), further transforms the rv-programs into rv-programs with a unique state/class on the borders for each interface cell type. We will focus on the definition of $Tr1$, but one has to have in mind that, in the inductive definition, all rv-programs used as arguments in a step have the particular format resulting form Lemma 1(ii).

The translation of modules and of composite programs
Module: A module M is translated to an rv-program $Tr1(M)$ that consists of one module. Fresh control/interaction labels for the input/output state/class in $Tr1(M)$ are used. The input registers/voices of $Tr1(M)$ are the variables that appear in the `read`/`listen` instruction and the output registers/voices of $Tr1(M)$ are the variables that appear in the `speak`/`write` instruction. The code of $Tr1(M)$ is the code from M enriched with `goto` statements for termination.

Horizontal composition: The rv-program $Tr1(X_1 \# X_2)$ is obtained taking the translations $Tr(X_1)$, $Tr(X_2)$ and identifying each output class of $Tr(X_1)$ to each input class of $Tr(X_2)$, provided they have the same type. (We suppose the labels in $Tr(X_1)$ and $Tr(X_2)$ are different.) The initial/final states of $Tr(X_1 \# X_2)$ are the join initial/final states of $Tr(X_1)$ and $Tr(X_2)$. The initial classes are those in $Tr(X_1)$ and the final those in $Tr(X_2)$.

Vertical and diagonal composition: They are similarly defined.

Collecting data form various components. The translation of "if" and "while" programs is further complicated by the fact that data may be spread on various components and we have to collect and test them before starting a real computation. We use a preprocessing unit which, in terms of scenarios, acts as follows (see Fig. 5(a))

$Pre_{yes} = |collect_H| \cdot (|collect_V| \triangleright [(|Test_{yes}| \triangleright |init_H|) \cdot (|init_v| \triangleright |P1|)])$ and
$Pre_{no} = |collect_H| \cdot (|collect_V| \triangleright [(|Test_{no}| \triangleright |init_H|) \cdot (|init_v| \triangleright |P2|)]$.

This preprocessing unit is implemented by an rv-program which consists of the following components:

- Two *collect* components which use transformed recorders and speakers (presented in Fig. 3(e,g)) to collect the data from general interfaces into single components, still preserving the interface. The type of the horizontal collector is $collect_H : \langle nil | a_1; \ldots; a_m \rangle \to \langle nil | (a_1, \ldots, a_m); a_1; \ldots; a_m \rangle$ and of the vertical one is $collect_V : \langle b_1; \ldots; b_n | nil \rangle \to \langle (b_1, \ldots, b_n); b_1; \ldots; b_n | nil \rangle$.
- A *test* block which uses the collected spatial and temporal data $(a_1, \ldots, a_m)$, $(b_1, \ldots, b_n)$ to test the condition. The output state/class label of its **goto**'s denote either the "yes" or the "no" branch.
- Two *init* components (blocks), which under the "yes" trigger, generate the initial states/classes for the program $Tr(P1)$ (following the "yes" branch). Similarly for the "no" branch.

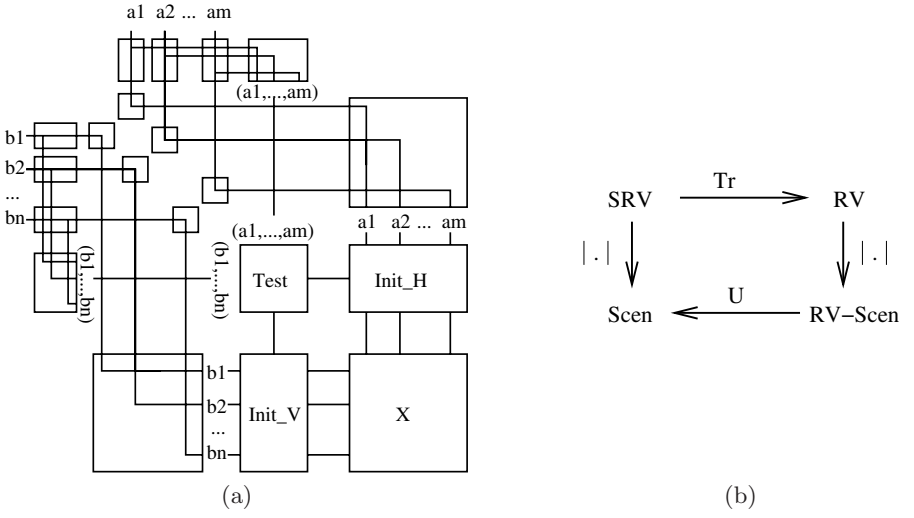


Fig. 5. Collecting data (a); translation correctness (b)

– Finally, the translated rv-programs $Tr(P1)$ and $Tr(P2)$, defined using the induction hypothesis.

The translation for if and while programs. For "if" and "while" programs, the translation acts as follows:

If–then–else: The translation $Tr1(if(C)\ then\ P_1\ else\ P_2)$ contains the translations of P_1 and P_2 and the above preprocessing unit Pre, where in the "yes" (resp. "no") case the *init* components generate initial states/classes from $Tr(P_1)$ (resp. $Tr(P_2)$). The initial states/classes are those of Pre, $Tr(P_1)$, and $Tr(P_2)$, while the final states/classes are those of $Tr(P_1)$ and $Tr(P_2)$.

Temporal while: The translation $Tr1(while\_t(C)\{P\})$ is the rv-program: (1) containing the translation of P; (2) containing a particular preprocessing unit Pre with dummy temporal data which passes the control either to P, or stops the execution passing the control to an exit block; (3) and such that the output state labels of P are identified to the input state labels of Pre (to repeat the procedure). The initial/final classes are those of P, while the initial states are those of Pre and $Tr(P)$, and the final states are those of the exit block.

Spatial while: The translation is similar.

Spatio-temporal while: The translation of the program $while\_sp(C)\{P\}$ is the rv-program which combine the translation of P, the preprocessing unit Pre which passes the control either to P, or stops the execution passing the control to an exit block, and additional components to connect the outputs of P to the inputs of Pre (similar to those used for diagonal composition). The initial states/classes are those of Pre and $Tr(P)$, while the final states/classes are those of the exit block.

6 The Translation Correctness

Actually, the translation correctness is reduced to the proof of the commutativity of the diagram in Fig. 5(b), where U is a forgetful transformation which strips the state/class labels from rv-scenarios.

Two scenarios are *weakly equivalent* $=_w$ if they have the same computation atoms, except for tests, or connections cells. For instance, for input data which pass the test `Test`, X and the scenario in Fig. 5(a) are weakly equivalent.

Theorem 1. *(i) The above translation Tr, from structured rv-programs to rv-programs, is correct with respect to the input-output semantics.*

(ii) Moreover, the translation weakly preserves the set of running scenarios. That is, up to mild scenario transformations regarding the use of tests and constants (recorders, speakers, etc.) the associated scenarios are the same.

References

1. Bruni, R.: Tile logic for synchronized rewriting of concurrent systems. PhD Thesis, Department of Computer Science, University of Pisa (1999)
2. Broy, M., Olderog, E.R.: Trace-oriented models of concurrency. In: Bergstra, J.A., et al. (eds.) Handbook of process algebra, pp. 101–196. North-Holland, Amsterdam (2001)
3. Dragoi, C., Stefanescu, G.: Structured programming for interactive rv-systems. IMAR Preprint 9/2006, Bucharest (2006)
4. Dragoi, C., Stefanescu, G.: Towards a Hoare-like logic for structured rv-programs. IMAR Preprint 10/2006, Bucharest (2006)
5. Dragoi, C., Stefanescu, G.: Implementation and verification of ring termination detection protocols using structured rv-programs. Annals of University of Bucharest, Mathematics-Informatics Series 55, 129–138 (2006)
6. Dragoi, C., Stefanescu, G.: AGAPIA v0.1: A programming language for interactive systems and its typing systems. In: Proc. FINCO/ETAPS (2007)
7. Gadducci, F., Montanari, U.: The tile model. In: Proof, language, and interaction: Essays in honor of Robin Milner, pp. 133–168. MIT Press, Cambridge (1999)
8. Goldin, D., Smolka, S., Wegner, P. (eds.): Interactive computation: The new paradigm. Springer, Heidelberg (2006)
9. Lindgren, K., Moore, C., Nordahl, M.: Complexity of two-dimensional patterns. Journal of Statistical Physics 91, 909–951 (1998)
10. Popa, A., Sofronia, A., Stefanescu, G.: High-level structured interactive programs with registers and voices. J. Universal Computer Science 13(11) (2007)
11. Stefanescu, G.: Network algebra. Springer, Heidelberg (2000)
12. Stefanescu, G.: Algebra of networks: modeling simple networks as well as complex interactive systems. In: Proof and System-Reliability, Proc. Marktoberdorf Summer School 2001, pp. 49–78. Kluwer, Dordrecht (2002)
13. Stefanescu, G.: Interactive systems with registers and voices. Fundamenta Informaticae 73, 285–306 (2006), (Early draft, School of Computing, National University of Singapore July 2004)
14. Stefanescu, G.: Towards a Floyd logic for interactive rv-systems. In: Letia, A.I. (ed.) Proc. 2nd IEEE Conference on Intelligent Computer Communication and Processing, Technical University of Cluj-Napoca, pp. 169–178 (September 1-2, 2006)
15. Wegner, P.: Interactive foundations of computing. Theoretical Computer Science 192, 315–351 (1998)

Optimal Orientation On-Line

Lech Duraj and Grzegorz Gutowski

Theoretical Computer Science Department, Jagiellonian University,
ul. Gronostajowa 3, 30-387 Kraków, Poland
lech.duraj@tcs.uj.edu.pl, grzegorz.gutowski@tcs.uj.edu.pl

Abstract. We consider the problem of graph orientation on-line. Orientation of a graph is an assignment of direction to every edge, resulting with a directed graph. The optimal orientation of a graph G is the one which maximizes the number of ordered pairs (u, v) of vertices of G for which there is a directed path from u to v in the resulting directed graph. Graph orientation on-line is a game in which one of the players constructs a graph by adding vertices one by one, so that the graph is connected at all times, and the second one assigns direction to the newly added edges. The goal of the second player is to maximize the number of connected pairs in the orientation, while the first player is trying to minimize it. We present asymptotically optimal strategies for both players and state that the game with n turns has a $\Theta\left(n\frac{\log n}{\log\log n}\right)$ outcome.

Keywords: On-line algorithms, Connectivity, Oriented graphs.

1 Introduction and Off-Line Results

The concept of the average connectivity, as an interesting measure of the reliability of a graph, was first introduced in [1] and further studied in [2]. Both papers focus on determining bounds on average connectivity of several classes of graphs. The results were extended for directed graphs in [3], where also a problem of finding an optimal orientation of a graph was introduced. We define optimal orientation slightly different, in order to simplify the presentation of obtained results. For a given graph G an *orientation* is an assignment of direction to every edge. In the resulting directed graph $\overrightarrow{G}$ we define the *connectivity measure* as the number of ordered pairs of different vertices (u, v) which are connected by a path from u to v. *Optimal orientation* of a graph is the one which maximizes the connectivity measure amongst all possible orientations.

The connectivity measure used in [3] was *average connectivity*, that is the average of number of internally disjoint paths from u to v over all ordered pairs (u, v) of distinct vertices. For trees both measures are the same, as there is at most one path connecting two vertices in a directed tree. An algorithm for constructing optimal orientation of a given tree was presented. It was also proven that on every tree with n vertices the number of connected pairs in an optimal orientation lies between $\frac{2}{9}n\,(n-1)$ and $\frac{1}{2}n\,(n-1)$.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 271–279, 2008.

2 On-Line Results

In an *on-line* version of the problem the algorithm works in turns. During each turn, the algorithm is given a single new vertex together with edges connecting it to previous vertices. The algorithm needs to decide the direction of each new edge before receiving next vertex, and the decisions are permanent. The algorithm's goal is to keep the total number of connected pairs as high as possible.

For a fixed algorithm A, this number is described by a function $s_A(G)$ dependent on the graph G together with on-line presentation. The function $s_A(n) = \min_{|G|=n} s_A(G)$ represents the score of the algorithm in the worst possible case. We are interested in determining the best possible score regardless of the graph given, i.e. determining the behavior of the function $s(n) = \max_A s_A(n)$.

Without putting any constraints on the presentation of a graph, we allow construction of a graph without edges. Each algorithm scores 0 in this game. Therefore to make the problem interesting, we require that the graph is connected in every turn. We will show that in this setting $s(n) = \Theta\left(n\frac{\log n}{\log\log n}\right)$. First, we show a specific greedy algorithm Gr, for which $s_{Gr}(n) = \Omega\left(n\frac{\log n}{\log\log n}\right)$. Then, we present a strategy of constructing an on-line graph which for every algorithm A assures $s_A(n) = O\left(n\frac{\log n}{\log\log n}\right)$.

It is common for analysis of on-line algorithms to determine the *competitivity ratio*, i.e., the ratio between the score of an on-line algorithm and the optimal off-line solution. In this problem, however, our results show that no finite ratio can be achieved, as optimal orientation of any tree (and therefore any connected graph) gives $\Omega(n^2)$ connected pairs. This is why we focus only on an on-line score without comparing it to the off-line one. Another feature of our approach is that in fact all interesting action appear already among graphs with lowest possible connectivity, that is, among trees.

The unusual $\frac{\log n}{\log\log n}$ component in our results comes from inverting the factorial, as stated in the following preliminary proposition:

Proposition 1. *For* $e_n := \max\{k : k! \quad n\}$ *we have* $e_n = \Theta\left(\frac{\log n}{\log\log n}\right)$.

Proof. It is well known that $c_1 k\log k \quad \log k! \quad c_2 k\log k$ for some $c_1, c_2 > 0$. By definition, $e_n! \quad n \quad (e_n+1)!$, which yields:

$$c_1 e_n \log e_n \quad \log n \tag{1}$$

$$c_2 (e_n+1)\log(e_n+1) \quad \log n$$

The left side of the second equation can be bounded from above by $c_2' e_n \log e_n$ for some $c_2' > 0$, so that

$$c_2' e_n \log e_n \quad \log n \tag{2}$$

From (1) and (2) we obtain, respectively:

$$\log c_1 + \log e_n + \log\log e_n \quad \log\log n$$

$$\log c_2' + \log e_n + \log\log e_n \quad \log\log n$$

which means that

$$\log e_n \quad d_1 \log\log n \tag{3}$$

$$\log e_n \quad d_2 \log\log n \tag{4}$$

for some $d_1, d_2 > 0$. Thus,

$$\frac{1}{c_2' d_1}\frac{\log n}{\log\log n} \quad e_n \quad \frac{1}{c_1 d_2}\frac{\log n}{\log\log n}$$

Where the first inequality follows from (1) and (4), while the second one follows from (3) and (2). This ends the proof. □

3 Lower Bound

The goal of this section is to describe an on-line algorithm which greedily chooses directions for edges to create as many connected pairs as possible at each step. We assume that the graph given is a tree – this is the worst case for the algorithm and the general case is shortly discussed at the end of the section. Notice that the only way of presenting a tree on-line is to give a leaf (a vertex connected to only one other vertex) in each turn – otherwise the graph would be disconnected or would contain a cycle.

For any vertex v, let $R_{\text{out}}^{(n)}(v)$ be the number of vertices reachable from v after n-th turn (we call it the *out-rank* of v), and let $R_{\text{in}}^{(n)}(v)$ be the number of vertices from which v can be reached (the *in-rank*). Given a new vertex t connected to an old vertex s, the greedy algorithm Gr compares the numbers $R_{\text{in}}^{(n)}(s)$ and $R_{\text{out}}^{(n)}(s)$. The algorithm Gr directs the edge (s,t) from s to t if and only if $R_{\text{in}}^{(n)}(s) \quad R_{\text{out}}^{(n)}(s)$. This way Gr greedily creates as many connected pairs as possible. Suppose that $R_{\text{in}}^{(n)}(s) \quad R_{\text{out}}^{(n)}(s)$, in this case it holds that $R_{\text{in}}^{(n+1)}(s) = R_{\text{in}}^{(n)}(s)$ and $R_{\text{out}}^{(n+1)}(s) = R_{\text{out}}^{(n)}(s) + 1$ (the set of reachable vertices is extended by t). Moreover, $R_{\text{in}}^{(n+1)}(t) = R_{\text{in}}^{(n)}(s) + 1$, since the vertices with a path to s, as well as s itself, have paths to t. The case of $R_{\text{in}}^{(n)}(s) < R_{\text{out}}^{(n)}(s)$ is similar: the bigger rank of the parent vertex stays the same, and the smaller one increases by 1. One of the ranks of t is zero, while the other is its parent's bigger rank increased by 1. Let us call this number the *order* of a vertex (precisely, the order $\sigma(v)$ of a vertex v is its only positive rank immediately after its creation). The order of a vertex, once assigned, does not change in the future.

Lemma 1. *At least one of the vertex ranks is greater or equal to its order.*

Proof. The ranks can never decrease (no paths are lost), and in the beginning one of them equals to the order. □

Lemma 2. *The order of a vertex is strictly greater then its parent's order.*

Proof. The order of a vertex is greater by 1 than the parent's bigger rank, which by Lemma 1 must be at least as big as its order. □

Lemma 3. *For any $k \in \mathbb{N}$, a vertex has at most k children of order k.*

Proof. Let u be a vertex. If a child of u inherits order k, then the bigger rank of u must be exactly $k-1$ at the moment. The smaller rank of u goes up by 1 for every child, therefore after connecting k children, one of the ranks must exceed $k-1$. Thus, there can be at most k children of order k. □

Lemma 4. *There are at most $(k+2)!$ vertices of order k.*

Proof. Let A_k denote the maximum number of vertices of order k. We prove by induction that $A_k \quad (k+2)!$. Only the very first vertex has order 0, thus $A_0 \quad 2$. For induction observe that a vertex of order $k+1$ is a child of a vertex of smaller order, and each of them has at most $k+1$ such children. Thus:

$$\begin{aligned} A_{k+1} \quad & (k+1)\left(A_k + A_{k-1} + \ldots + A_0\right) \\ & (k+1)\left((k+2)! + (k+1)! + \ldots + 2!\right) \\ & (k+1)\,(k+2)! + (k+1)\cdot 2\cdot (k+1)! \\ & (k+1)\,(k+2)! + 2\,(k+2)! = (k+3)! \end{aligned}$$

□

Lemma 5. *There are at most $\frac{(k+3)!}{2}$ vertices of order at most k.*

Proof. Obvious from Lemma 4, as $2! + \ldots + (k+2)! < \frac{(k+3)!}{2}$. □

Lemma 6. *In a tree with $n \quad 2$ vertices, at least $\frac{n}{2}$ of them have order $e_n - 2$ or greater.*

Proof. Suppose not. Then more than $\frac{n}{2}$ vertices have order less or equal $e_n - 3$. By Lemma 5, there are less than $\frac{e_n!}{2} \quad \frac{n}{2}$ such vertices, a contradiction. □

Theorem 1. *Let T be a tree with $|T| = n$. The total number $s_{Gr}(T)$ of connected pairs is $\Omega(ne_n)$.*

Proof.

$$\begin{aligned} s_{Gr}(T) = \frac{1}{2}\sum_{v\in G}\left(R^{(n)}_{\text{in}}(v) + R^{(n)}_{\text{out}}(v)\right) \quad & \frac{1}{2}\sum_{v\in G}\sigma(v) \\ \frac{1}{4}\sum_{v\in G}(e_n - 2) = \frac{1}{4}\left(ne_n - 2n\right) & = \Omega(ne_n) \end{aligned}$$

The first inequality follows by Lemma 1, and the second one follows by Lemma 6. □

Now we are ready to consider the general case. If the graph given is not a tree, the only difference is that the arriving vertex is possibly connected to many previous vertices. A modified version of greedy algorithm simply picks an arbitrary one of the incoming edges and forgets about the rest. With this modification, the algorithm obtains at least the same score as greedy algorithm Gr on the resulting spanning tree.

Corollary 1. $s(n) = \Omega(ne_n) = \Omega\left(n\frac{\log n}{\log\log n}\right)$.

4 Upper Bound

In this section we describe a strategy which does not allow any algorithm to exceed $O\left(n\frac{\log n}{\log\log n}\right)$ bound for the number of connected pairs. Again, the constructed graph is a tree. The situation can be viewed as a game between us and another player, the algorithm A: in each move we present to A a vertex and an edge which A has to direct. We keep the number of connected pairs after n turns bounded by $O\left(n\frac{\log n}{\log\log n}\right)$. Of course, our construction depends on how A has directed previous edges.

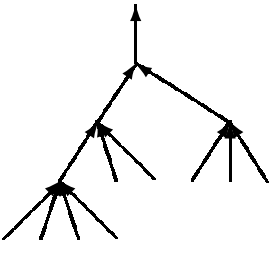


Fig. 1. Ascending 1-factorial tree

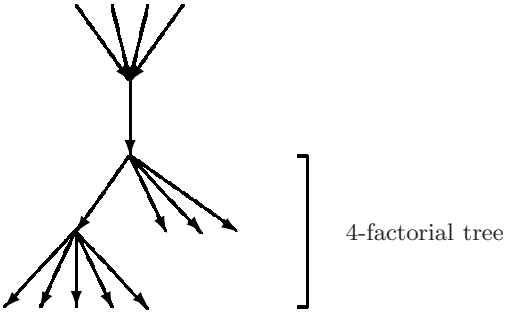


Fig. 2. Descending $4^{\star}$-factorial tree

By a *k-factorial tree* we mean a directed tree in which the following conditions hold:

- all edges are directed the same way, i.e. all towards the root, or all towards the leaves. Depending on this orientation, we call a factorial tree *ascending* or *descending* respectively.
- the root has up to k children, each of which has up to $k+1$ children, and so on.

The $k^\star$*-factorial tree* is a k-factorial tree, with additional parent and exactly k grandparents attached to the root. As in k-factorial tree all edges need to be directed the same way. By *factorial trees* we mean both k- and $k^\star$-factorial trees.

The key idea behind our strategy is to force A to orient the edges such that entire tree becomes an union of factorial trees. During the construction, we assign to every vertex v a number $\rho(v)$, called *rank*, and one additional bit of information, the *mark*. Our strategy attaches new vertices only to marked ones.

First we present a pseudo-code of our strategy, and after it we describe it in a more readable form.

```
1   rank[]; // For each vertex stores its rank.
2   mark[]; // For each vertex stores its mark.
3   dir[];  // For each vertex stores the direction
4           // of the edge connecting it to the parent.
5   // Two connected vertices v_0 and v_1 are presented to A.
6   rank[v_0] = 1;
7   mark[v_0] = false;
8   rank[v_1] = 2;
9   mark[v_1] = true;
10  // Here algorithm A needs to direct the first edge v_0v_1.
11  dir[v_1] = ask A;
12  while true do
13     u = select a marked vertex of lowest possible rank;
14     r = rank[u];
15     s = 0;
16     for i = 1 to r do
17        // New vertex a_i connected to u is presented to A.
18        rank[a_i] = r + 1;
19        dir[a_i] = ask A;
20        if dir[a_i] <> dir[u] then
21           s = i;
22           break;
23     mark[u] = false;
24     if s <> 0 then
25        // A has played reversal move and a_s is reversal vertex.
26        for i = 1 to s - 1 do
27           mark[a_i] = false;
28        rank[a_s] = s;
29        mark[a_s] = true;
30     else
31        // A has played blooming move.
32        for i = 1 to r do
33           mark[a_i] = true;
```

Our strategy starts with two connected vertices v_0 and v_1. It gives one of them rank 1 and unmarks it (prevents it to have more neighbors), whereas the other one receives rank 2 and it is marked. The strategy repeats the following move: selects the marked vertex u of lowest possible rank r, and tries to create, one by one, up to r children $a_1, \ldots, a_r$ of u. Assuming, without loss of generality, that the edge connecting u to its parent is descending (i.e. it is directed towards u), there are two possible ways A may react:

1. All new edges are directed from u to new vertices $a_1, \ldots, a_r$. In this case, strategy unmarks u, gives positive mark and assigns rank $r+1$ to all of the a_i's and continues the construction. As long as A chooses this response (called *blooming*), a factorial tree is created. Notice that algorithm Gr described in Section 3 always reacts this way.
2. One of the edges to a new vertex, say a_s, is directed by A towards u. If this happens, our strategy creates no more children of u, and starts a new $s^\star$-factorial tree: giving rank s to a_s and unmarking u as well as $a_1, \ldots, a_{s-1}$. The new $s^\star$-factorial tree will be ascending, thus we call this move *reversal*, and a_s is the *reversal vertex*. From now on the vertices $a_1, \ldots, a_{s-1}$ as unmarked are considered to be "dummy" vertices – although they are leaves, our strategy will not develop their subtrees anymore.

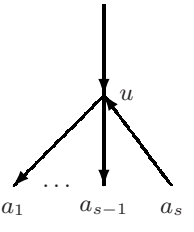


Fig. 3. Reversal move

Let T be a rooted, undirected tree. A set $I \subseteq T$ is called *independent* if for different $a, b \in I$, a does not lie on the path from b to the root of T. For $a \in T$ the set $\{x \in T : a \text{ lies on the path from } x \text{ to the root of } T\}$ is a *subtree* of a.

In the following, by a *dummy* vertex we mean simply an unmarked leaf. Note that a dummy vertex is always a result of a reversal move.

Lemma 7. *Let u be a non-dummy vertex with $\rho(u) = r$. Then in the subtree of u there is an independent set of r non-dummy vertices of rank $r+1$, or a marked vertex of rank at most r.*

Proof. If u is a non-dummy leaf, then it is itself marked and of rank r. If A makes a blooming move at u, then the children of u form an independent set of r non-dummy vertices of rank $r+1$. The reversal at u always creates a new marked vertex u' with $\rho(u')$ r. Such a vertex can be unmarked only at line 23

of our strategy. If rank of u' is lower than r or A makes a reversal move at u', a new marked vertex u'' of rank at most r is created.

We repeat the same argument with u' replaced by u'' and state that the only way for A to unmark a marked vertex of rank at most r and not to create another one is to make a blooming move at a vertex of rank exactly r. □

Lemma 8. *If there is a vertex of rank $r+1$, there must be at least $(r-1)!$ independent, non-dummy vertices of rank r.*

Proof. To induct on r note that there is nothing to show for $r = 1$. For $r \quad 2$, consider the moment, when a vertex of rank $r+1$ appears for the first time. The vertex needs to be a child of a vertex of rank r and, by induction, there is a set I of $(r-2)!$ independent, non-dummy vertices of rank $r-1$. Notice that there are no marked vertices of rank $r-1$ or lower – this is because in line 13 of our strategy a marked vertex with lowest possible rank is selected. Lemma 7 gives us, that in a subtree of each vertex in I there are at least $r-1$ independent vertices, each of rank r. Together all those vertices form a set of $(r-1)!$ independent vertices of rank r. □

Lemma 9. *Each vertex u of rank r forms at most r connected pairs with vertices created earlier in our construction.*

Proof. To induct on r note that when a vertex has rank 1, then it is either v_0 or a reversal vertex being the first (and only) child of its parent.

For $r \quad 2$, all paths connecting u with earlier vertices must go through its parent v. First suppose that $r \quad \rho(v)$ and observe that this may happen only if u is a reversal vertex. In this case u forms connected pairs with r of previous vertices, that is v and $r-1$ dummy children of v.

Now, suppose that $r = \rho(v)+1$. This means that u is not a reversal vertex and none of earlier children of v is connected by a path with u. Therefore all vertices connected with u are v and vertices earlier than v. By induction hypothesis it follows, that there are at most $\rho(v)+1 = r$ of them. □

Theorem 2. *Our strategy allows any algorithm to achieve at most $O(ne_n)$ connected pairs on trees with n vertices.*

Proof. Let m be maximum rank of vertices in a tree. By Lemma 8, it follows that $n \quad (m-2)!$ and $m \quad e_n+2$. To count the total number of connected pairs it is sufficient to count for each vertex u the number $c(u)$ of vertices forming a connected pair with u and created before u – this is because in each pair of vertices one is created earlier than the other. However, Lemma 9 tells us that $c(u) \quad \rho(u) \quad m$. Hence, the total number of pairs is $O(nm) = O(ne_n)$. □

Theorem 3. $s(n) = \Theta\left(n\frac{\log n}{\log\log n}\right)$.

Proof. Immediate from Corollary 1 and Theorem 2. □

References

1. Beineke, L.W., Oellermann, O.R., Pippert, R.E.: The average connectivity of a graph. Discrete Mathematics 252, 31–45 (2002)
2. Dankelmann, P., Oellermann, O.R.: Bounds on the average connectivity of a graph. Discrete Applied Mathematics 129, 305–318 (2003)
3. Henning, M.A., Oellermann, O.R.: The average connectivity of a digraph. Discrete Applied Mathematics 140, 143–153 (2004)

Some Tractable Instances of Interval Data Minmax Regret Problems: Bounded Distance from Triviality

Bruno Escoffier[1], Jérôme Monnot[1], and Olivier Spanjaard[2]

[1] LAMSADE-CNRS, Université Paris Dauphine, Place du M^{al} de Lattre de Tassigny, F-75775 Paris Cedex 16, France
{escoffier,monnot}@lamsade.dauphine.fr
[2] LIP6, Université Pierre et Marie Curie, 4 Place Jussieu, F-75252 Paris Cedex 05, France
olivier.spanjaard@lip6.fr

Abstract. This paper focuses on tractable instances of interval data minmax regret graph problems. More precisely, we provide polynomial and pseudopolynomial algorithms for sets of particular instances of the interval data minmax regret versions of the shortest path, minimum spanning tree and weighted (bipartite) perfect matching problems. These sets are defined using a parameter that measures the distance from well known solvable instances. Tractable cases occur when the parameter is bounded by a constant. Two kinds of parameters are investigated, measuring either the distance from special weight structures or the distance from special graph structures.

Keywords: Robust optimization, Interval data, Shortest path, Spanning tree, Bipartite perfect matching.

1 Introduction

In recent years there has been a growing interest in robust optimization problems [15]. Studies in this field concern problems where some parameters are ill-known due to uncertainty or imprecision. Usually, in valued graph optimization problems, the ill-known parameters are the valuations. In such a case, a set of scenarios is defined, with one scenario for each possible assignment of valuations to the graph. Two approaches can be distinguished according to the way the set of scenarios is defined: the *interval model* where each valuation is an interval and the set of scenarios is defined implicitly as the cartesian product of all the intervals; the *discrete scenario model* where each valuation is a vector, every component of which is a particular scenario. Intuitively, a *robust* solution is a solution that remains suitable whatever scenario finally occurs. Several criteria have been proposed to formalize this: the *minmax* criterion consists of evaluating a solution on the basis of its worst value over all scenarios, and the *minmax regret* criterion consists of evaluating a solution on the basis of its maximal deviation from the optimal value over all scenarios. We will mainly focus here on the

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 280–291, 2008.

robust shortest path problem (RSP for short), the *robust minimum spanning tree* problem (RST for short) and the *robust minimum weighted (bipartite) perfect matching* problem (R(B)PM for short), with the minmax regret criterion in the interval model.

Formally, an interval data minmax regret network optimization problem can be defined as follows. Let $G = (V, E)$ be a given directed or undirected graph with n vertices and m edges. A feasible solution is a subset $\pi \subseteq E$ satisfying a given property Π (for example, being a path, a tree or a matching). Each edge $e \in E$ is valued by an interval $I_e = [l_e; u_e]$ of possible weights. The set of scenarios is the cartesian product $\mathcal{S} = \prod_{e \in E} I_e$. In other words, a scenario $s \in \mathcal{S}$ consists in assigning a weight $w_s(e) \in I_e$ for every $e \in E$. For any feasible solution π and any scenario $s \in \mathcal{S}$ of an instance $\mathcal{I} = (G, I_E)$ where $I_E = \{I_e : e \in E\}$, the *value of* π *under scenario* s is $w_s(\pi) = \sum_{e \in \pi} w_s(e)$ and its *regret under scenario* s is $R_s(\pi) = |w_s(\pi) - opt(s)|$, where $opt(s)$ is the value of an optimal solution for the standard instance valued by w_s (rigorously, we should write $R_s(\mathcal{I}, \pi)$ but we omit to indicate $\mathcal{I}$ when no confusion is possible). The max regret of solution π is defined by $R(\pi) = \max_{s \in \mathcal{S}} R_s(\pi)$. The aim of a minmax regret optimization problem is, given an instance $\mathcal{I} = (G, I_E)$, to find a feasible solution π^* minimizing $R(\pi^*)$. Note that, for a minimization problem, $R(\pi) = R_{s(\pi)}(\pi)$, where $s(\pi)$, called *worst case scenario for* π, is defined by $w_{s(\pi)}(e) = u_e$ if $e \in \pi$ and $w_{s(\pi)}(e) = l_e$ otherwise [3].

In this paper, we consider tractable instances of RSP and RST, that have been proved strongly NP-hard [4] in the general case, as well as tractable instances of RBPM, the restriction of which to complete bipartite graphs (known as the interval data minmax regret assignment problem) has been proved NP-hard [12]. For this purpose, as suggested by Guo *et al.* [10], we introduce parameters that measure the distance from well known solvable instances. For example, if all the intervals of an instance reduce to a single point –*degenerate* intervals–, then the robust optimization problem reduces to a standard optimization problem, and is therefore polynomially solvable provided that the standard version is polynomial. One can define the distance from this easy case as the number k of non degenerate intervals. If this distance k is bounded by a constant, then the robust optimization problem is polynomially solvable by a brute force algorithm [4]. In this work, we focus on two kinds of parameters: the ones that measure the distance from special valuation structures (instances the minmax regret of which is zero, instances with linearly ordered valuations), and the ones that measure the distance from special graph structures (series-parallel graphs, trees). The paper is organized as follows. The first two sections deal with the first kind of parameters: we show that RSP and RBPM are polynomially solvable when the minmax regret is bounded by a constant k (Section 2), as well as RST when the number of intersecting intervals in the instance is bounded by a constant k (Section 3). More precisely, following parameterized complexity terminology [8], the first two problems are in XP (problems solvable in $O(n^{f(k)})$ for some function f) while the third one is in FPT (problems solvable in $O(f(k)n^c)$ for some constant c). The next sections deal with the second kind of parameters:

we show that RSP is pseudopolynomial for graphs which are close to be series-parallel (Section 4), and that RSP and RBPM are pseudopolynomial for graphs with bounded treewidth and bounded degree (Section 5). Due to lack of space, some proofs are omitted and can be found in [9].

2 Upper Bounded Minmax Regret

In this section, we investigate the hardness of solving an interval data minmax regret graph optimization problem when there exists a solution with bounded maximal regret. Note that studying instances where the optimum value is upper bounded is a classical way to understand the intrinsic difficulty of a combinatorial optimization problem (problems which become polynomially solvable in this case are called *simple*, see Paz and Moran [16]). Here, we first show that we can easily determine if there is a solution of maximal regret 0, *i.e.* a solution which is optimal under every possible scenario. Next, we show that for RSP and RBPM, we can extend this result to polynomially determine if there exists a solution of maximal regret at most k.

First, let us prove that the problem of the existence of a solution of maximal regret 0 can be easily solved for any interval data minmax regret graph optimization problem Π. We use a nice generic 2-approximation algorithm proposed by Kasperski and Zielinski [13]. For any instance $\mathcal{I}$ this algorithm outputs a solution π such that $R(\pi) \leq 2R(\pi^*)$ (where $R(\pi^*)$ is the minmax regret of $\mathcal{I}$). If $R(\pi^*) = 0$, then $R(\pi) = 0$, else since $R(\pi) \geq R(\pi^*)$, we have $R(\pi) > 0$. The expected result follows (Π being assumed to be polynomial). Now, by a reduction to the regret 0 case, we prove the following:

Proposition 1. *For RSP, the problem of determining if the minmax regret is at most k can be solved in time $O(n^2 m^k)$.*

Proof. Let $\mathcal{I} = (G, I_E)$ be an instance of RSP and denote by r its optimum regret. Let us remark that if there exists a degenerate interval $I_e = \{0\}$ in $\mathcal{I}$ with $e = (v_1, v_2)$, then one can merge nodes v_1 and v_2 and get an equivalent instance (possibly with multiedges). In particular, we can assume that $u_e > 0$ for any e. We construct m instances $\mathcal{I}_1, \ldots, \mathcal{I}_m$ of RSP as follows: $\mathcal{I}_i$ is the same instance as $\mathcal{I}$ up to the interval $[l_i, u_i]$ associated in $\mathcal{I}$ to e_i which is transformed into $[\max\{l_i - 1; 0\}, u_i - 1]$. We claim that:

(i) $r_i^* \geq r - 1$ where r_i^* denotes the optimum regret of $\mathcal{I}_i$;
(ii) if $r_i^* = r - 1$ then any optimum solution for $\mathcal{I}_i$ is optimum for $\mathcal{I}$;
(iii) there exists at least one i such that $r_i^* = r - 1$ (if $r > 0$).

If the claims are true, then by applying k times these procedures, $\mathcal{I}$ has an optimum regret at most k if and only if (at least) one of the final instances has optimum regret 0 (if at some point, we find an interval reduced to $\{0\}$, we can merge the corresponding nodes). We get m^k instances; the generic 2-approximation algorithm is in $O(n^2)$ for RSP, and the complexity follows. Claims (i) and (ii) hold since the regret of any path π satisfies $R_i(\pi) \geq R(\pi) - 1$ (under any scenario,

the value of any path has decreased by at most 1). For Claim (iii), consider an optimum solution $\pi^* = ((v_0, v_1), \cdots, (v_{p-1}, v_p))$ (where $v_0 = s$ and $v_p = t$) of $\mathcal{I}$, and its worst case scenario $s(\pi^*)$ in $\mathcal{I}$. We prove that there exists at least one edge $e_i \in \pi^*$ such that no shortest path in $s(\pi^*)$ contains this edge. Note that if this is true, then consider instance $\mathcal{I}_i$: in $s(\pi^*)$, the value of the shortest path is the same in $\mathcal{I}$ and in $\mathcal{I}_i$, hence the regret of π^* decreased by 1, and Claim (iii) is true. Then, assume that for any i, there exists a shortest path π^i (in $s(\pi^*)$) which contains (v_{i-1}, v_i). Let w_1^i be the value (in $s(\pi^*)$) of this path between s and v_{i-1} and w_2^i its value between v_i and t (hence $w_1^1 = w_2^p = 0$). Since π^* has regret r, we get ($s(\pi^*)$ is omitted for readability) that $w(\pi^i) = w_1^i + w_2^i + u_{(v_{i-1}, v_i)} = w(\pi^*) - r$. Summing up we obtain:

$$\sum_{i=1}^{p} (w_1^i + w_2^i) = pw(\pi^*) - pr - \sum_{i=1}^{p} u_{(v_{i-1}, v_i)} = (p-1)w(\pi^*) - pr \qquad (1)$$

But remark that for each $i \in \{2, \cdots, p\}$ we can build a path of value $w_1^i + w_2^{i-1}$ (composed of the initial part of π^i from s to v_{i-1} and the final part of π^{i-1} from v_{i-1} to t). Then, since each of these paths has value at least $w(\pi^*) - r$:

$$\sum_{i=2}^{p} (w_1^i + w_2^{i-1}) \geq (p-1)(w(\pi^*) - r) = (p-1)w(\pi^*) - pr + r \qquad (2)$$

But since $w_1^1 = w_2^p = 0$, Equations (1) and (2) are incompatible for $r > 0$. □

The central property, leading to Claim (iii), is that, in an optimum solution π^* for which $R(\pi^*) > 0$, there exists at least one edge that does not belong to any optimum solution in $s(\pi^*)$. Actually, one can show that this property is also true for the interval data minmax regret perfect matching problem in bipartite graphs. For any instance $\mathcal{I} = (G, I_E)$ of R(B)PM, we assume that G has a perfect matching (in particular, the number n of vertices of G is even).

Proposition 2. *For RBPM, the problem of determining if the minmax regret is at most k can be solved in time $O(n^2 m^k)$.*

Proof. The proof is almost identical to the one of Proposition 1. Let $\mathcal{I} = (G, I_E)$ be an instance of RBPM where $G = (V, E)$ is a bipartite graph which admits a perfect matching and denote by r its optimum regret. W.l.o.g., assume that $l_e \geq k$ for any e. Actually, by adding any constant $c > 0$ to each interval I_e, we obtain an equivalent instance since all the perfect matchings have the same size. As previously, we build m instances $\mathcal{I}_1, \ldots, \mathcal{I}_m$ of RBPM where $\mathcal{I}_i$ is the same instance as $\mathcal{I}$ up to the interval $[l_i, u_i]$ associated in $\mathcal{I}$ to e_i which is transformed to $[l_i - 1, u_i - 1]$. Using the same notation as in Proposition 1, we claim that: (i) $r_i^* = R(\mathcal{I}_i) \geq r - 1$; (ii) if $r_i^* = r - 1$ then any optimum solution for $\mathcal{I}_i$ is optimum for $\mathcal{I}$; (iii) there exists at least one i such that $r_i^* = r - 1$ (if $r > 0$).

The proof of Claims (i) and (ii) is identical to the proof of Proposition 1. So, we only prove Claim (iii). Consider an optimum solution $\pi^* = \{e_1, \cdots, e_{\frac{n}{2}}\}$ of $\mathcal{I}$, and its worst case scenario $s(\pi^*)$ in $\mathcal{I}$. As previously, we prove that there exists

at least one edge $e_i \in \pi^*$ such that no perfect matching with minimum weight in $s(\pi^*)$ contains this edge. Assume the reverse, and let π^i for $i = 1, \cdots, \frac{n}{2}$ be a perfect matching with minimum weight $w(\pi^*) - r$ which contains edge e_i in scenario $s(\pi^*)$ (note that possibly some π^i are identical). Then, in scenario $s(\pi^*)$ we have:

$$\sum_{i=1}^{\frac{n}{2}} w(\pi^i \setminus e_i) = \frac{n-2}{2} w(\pi^*) - \frac{n}{2} r \tag{3}$$

On the other hand, the graph G' induced by $\cup_{i=1}^{\frac{n}{2}} \left(\pi^i \setminus e_i\right)$ is $(\frac{n}{2} - 1)$-regular (G' is considered as a multigraph, that is if an edge (x, y) appears p times in $\cup_{i=1}^{\frac{n}{2}} \left(\pi^i \setminus e_i\right)$, then there are p parallel edges between x and y in G'). Since G' is bipartite and $(\frac{n}{2} - 1)$-regular, G' can be decomposed into $(\frac{n}{2} - 1)$ matchings π'^i for $i = 1, \ldots, \frac{n}{2} - 1$. These matchings π'^i are perfect in G and if π' is a matching of minimum weight in scenario $s(\pi^*)$ among the matchings π'^i for $i = 1, \ldots, \frac{n}{2} - 1$, then the value of π' satisfies:

$$\frac{n-2}{2} w(\pi') \leq \sum_{i=1}^{\frac{n}{2}} w(\pi^i \setminus e_i) \tag{4}$$

Using equality (3) and inequality (4) we obtain $w(\pi') \leq w(\pi^*) - (1 + \frac{2}{n})r$, which is impossible for $r > 0$ since $w(\pi') \geq w(\pi^*) - r$.

By applying k times this method, we build m^k instances such that $\mathcal{I}$ has an optimum regret at most k iff (at least) one of the final instances has optimum regret 0. Since we supposed that $\forall e \in E, l_e \geq k$ for the initial instance, all the interval lower bounds in the final instances are non-negative. □

Our method seems to be quite general and may be fruitfully applied to other problems, but however not to all of them. Indeed, the property leading to Claim (*iii*) is no more true for some problems such as RST or RPM (in arbitrary graphs), and for them the question whether they are simple (according to the definition of [16]) or not remains open.

3 Upper Bounded Number of Interval Intersections

As previously mentioned, RST and RSP are fixed parameter tractable (FPT) when the parameter is the number of non degenerate intervals (with a brute force algorithm). Minimum spanning trees have special properties that leads to another easy cost structure: when all intervals are disjoint ($I_e \cap I_f = \emptyset$ for any edges e and f), any minimum spanning tree under any scenario is an optimum solution for RST [1]. Indeed, Kruskal's algorithm leads then to the same tree, independently of the scenario. This tree is optimal, and its regret is 0. Note that, on the other hand, even if all intervals are $[0, 1]$, RST is NP-hard [1,4]. Here, we show that considering as parameter the number of intervals that intersect at least one other interval, RST is FPT. Although using brute force, the optimality of the algorithm is not obvious.

Proposition 3. *RST can be solved in time* $O(2^k\ m \log m)$*, where* k *is the number of intervals that intersect at least one other interval.*

Proof. Let $\mathcal{I} = (G, I_E)$ be an instance of RST where $G = (V, E)$ and $I_e = [l_e, u_e]$ for any $e \in E$. We define $J = \{I_{e_1} : \exists e_2 \neq e_1, I_{e_1} \cap I_{e_2} \neq \emptyset\}$, and we set $k = |J|$. Let $J' \subseteq J$. We want to compute the best (in terms of regret) spanning tree π such that $\pi \cap E_J = E_{J'}$ (where E_J denotes the set of edges corresponding to intervals in J). If $E_{J'}$ contains a cycle, there is no such tree. If not, we proceed as follows: we remove from E the set $E_{J \setminus J'}$ and, considering $E_{J'}$ as part of the spanning tree, we complete it by applying Kruskal's algorithm to the remaining graph (choosing any valuation $w(e) \in [l_e, u_e]$ since the output does not depend on the value of an edge $e \notin J$). Let $\pi_{J'}$ be the obtained solution.

Now, let π be a spanning tree such that $\pi \cap E_J = E_{J'}$. We want to prove that $R(\pi_{J'}) \leq R(\pi)$. First, note that $\pi_{J'}$ and π agree on E_J. Then, under any scenario where $w(e) = u_e$ for $e \in E_{J'}$ and $w(e) = l_e$ for $e \in E_{J \setminus J'}$, Kruskal's algorithm will produce the same optimum solution π^*. In particular π^* is optimal *both* in $s(\pi)$ and $s(\pi_{J'})$. However, π^* has not the same value in these two scenarios. Then:

$$R(\pi_{J'}) - R(\pi) = w_{s(\pi_{J'})}(\pi_{J'}) - w_{s(\pi_{J'})}(\pi^*) - \Big(w_{s(\pi)}(\pi) - w_{s(\pi)}(\pi^*)\Big)$$

We upper bound this by considering each edge of the graph. If $\pi_{J'}$ and π agree on an edge e (either take it or not), then the difference is 0 for this edge, since this edge has the same value in $s(\pi)$ and $s(\pi_{J'})$, and since we refer to the same tree π^*. Note that this includes all edges in E_J. If $\pi_{J'}$ and π disagree on e:

- either e is in $\pi_{J'} \setminus \pi$. If e is not in π^*, then in the regret it counts u_e for $\pi_{J'}$ (u_e for $\pi_{J'}$ and 0 for π^*) and 0 for π (0 for π and 0 for π^*). If e is in π^*, it counts 0 for $\pi_{J'}$ and $-l_e$ for π. The loss (in terms of regret) from $\pi_{J'}$ with respect to π is therefore at most u_e;
- or e is in $\pi \setminus \pi_{J'}$. If e is not in π^*, then it counts 0 for $\pi_{J'}$ and u_e for π. If e is in π^*, it counts $-l_e$ for $\pi_{J'}$ and 0 for π. Then, with respect to π, $\pi_{J'}$ "wins" at least l_e.

Summing up these inequalities for all edges leads to:

$$R(\pi_{J'}) - R(\pi) \leq \sum_{e \in \pi_{J'} \setminus \pi} u_e - \sum_{e \in \pi \setminus \pi_{J'}} l_e \tag{5}$$

Now, recall that π and $\pi_{J'}$ agree on J, and that the intervals not in J do not intersect. Hence, whatever the value of edges not in J, $\pi_{J'}$ will have a better value than π. This is true in particular when the weight of each $e \notin J$ is fixed to u_e if e is in $\pi_{J'}$ and to l_e otherwise. This means that

$$\sum_{e \in \pi_{J'} \setminus \pi} u_e \leq \sum_{e \in \pi \setminus \pi_{J'}} l_e \tag{6}$$

Equations (5) and (6) lead to the result that $\pi_{J'}$ is the best tree π such that $\pi \cap J = J'$.

To conclude, we only have to consider each possible $J' \subseteq J$, and take the best solution so computed. The global complexity is hence $2^k O(m \log m)$. □

Note that for RSP, making assumptions on interval intersections does not simplify the problem.

Proposition 4. *RSP is NP-hard even if there are no intersections between intervals.*

4 Upper Bounded Reduction Complexity

We now consider a particular class of directed acyclic graphs (DAGs), namely *series-parallel graphs*. This class can be defined using the following kinds of reductions in a DAG: (1) a *series reduction* at v is possible when $e_1 = (u, v)$ is the unique edge into v and $e_2 = (v, w)$ is the unique edge out of v: then e_1 and e_2 are replaced by $e = (u, w)$; (2) a *parallel reduction* at u, w replaces two edges e_1, e_2 joining u to w by a single edge $e = (u, w)$. Two nodes s and t are distinguished as the source and the sink (st-DAG). A graph is said to be *edge series-parallel* (ESP) if it can be reduced to a single edge (s, t) by using such reductions. Kasperski and Zielinski have recently shown that RSP is NP-hard in ESP graphs, but admits a pseudopolynomial algorithm in this case [14]. In this section, we extend this result to graphs close to be ESP. For the convenience of the reader, we first describe the basic principles of the pseudopolynomial algorithm for ESP graphs. It operates by applying a sequence of series and parallel reductions from the input graph $G = (V, E)$ to a single edge (s, t). This sequence is given by an algorithm in $O(m)$ to recognize ESP graphs [17], where $m = |E|$. In a reduced graph, a subset $E_i \subseteq E$ is associated with every edge e_i. These subsets are defined recursively: the set $\{e\}$ is associated with every $e \in E$; let e_1, e_2 denote the edges involved in a reduction, then the set $E_1 \cup E_2$ is associated with the new edge. For every edge e_i, the subgraph of G *induced by* E_i is denoted G_{e_i}. Let u_π and $R(\pi)$ denote respectively the worst value and the max regret of a path π in an induced subgraph G_e. The principle of the algorithm is, for each reduction yielding a new edge $e = (v, w)$, to keep only a minimal subset P_e of non-dominated paths from v to w, where π dominates σ if $u_\pi \leq u_\sigma$ and $R(\pi) \leq R(\sigma)$ with at least a strict inequality. Indeed, those paths are potential subpaths of a minmax regret path from s to t in G. Initially, $P_e = \{e\}$ for every edge e. Then, for any new edge e obtained by a reduction involving e_1 and e_2, set P_e is computed from $P_{e_1} \cup P_{e_2}$ in a parallel reduction, and from $P_{e_1} \times P_{e_2}$ (concatenated paths) in a series reduction. When the sequence of reductions terminates, there is only a single edge (s, t), and path $\pi^* = \arg\min_{\pi \in P_{(s,t)}} R(\pi)$ is a minmax regret path from s to t in G. Noticing that $|P_e|$ is upper bounded by $L_{\max}$, where $L_{\max}$ is the value of the longest path from s to t in G over all scenarios, the authors, thanks to a recursive computation of u and R (avoiding shortest path computations from scratch when computing $R(\pi)$ for $\pi \in P_{e_1} \cup P_{e_2}$ or $P_{e_1} \times P_{e_2}$), establish that the running time is $O(mL^2_{\max})$, and therefore pseudopolynomial.

We now extend this result to graphs close to be ESP. We first need to measure how far a graph is from being ESP. For that purpose, the notion of *reduction*

complexity has been introduced [5]. It uses a third kind of reduction, called *node reduction*. Such a reduction can be performed at a node v when v has in-degree or out-degree 1: suppose v has out-degree 1, let $e_1 = (u_1, v), \ldots, e_\delta = (u_\delta, v)$ be the edges into v and $e_{\delta+1} = (v, w)$ be the edge out of v, then $\{e_1, \ldots, e_{\delta+1}\}$ is replaced by $\{e'_1, \ldots, e'_\delta\}$, where $e'_i = (u_i, w)$ (the case where v has in-degree 1 is symmetric). Note that every st-DAG can be reduced to a single edge (s, t) by iterating the three types of reductions. The reduction complexity of a graph G is defined as the minimum number of node reductions sufficient –along with series and parallel reductions– to reduce G to (s, t). There exists an $O(n^{2.5})$ algorithm to compute an optimal reduction sequence [5] (*i.e.*, involving a minimum number of node reductions), and hence to determine reduction complexity. Thanks to this, the result of Kasperski and Zielinski [14] can be extended:

Proposition 5. *RSP can be solved in time* $O(2^k m^2 \ L^2_{\max})$ *in st-DAGs of reduction complexity* k.

5 Upper Bounded Treewidth and Max Degree

The *treewidth* of a graph can be seen as a measure of how far it is from being a tree (the treewidth of a tree is 1). It is well-known that the treewidth of an (undirected) ESP graph is at most 2. A natural extension of the previous result is therefore to investigate the complexity of RSP in graphs of bounded treewidth (more precisely, in graphs whose corresponding undirected simple graph has a bounded treewidth). Clearly, RSP is polynomially solvable in a graph G the treewidth of which is $k = 1$ (G is a tree), or the max degree of which is $\Delta \leq 2$ (G is a set of cycles and/or chains). However, it is NP-hard when $k = 2$ and $\Delta = 3$ (since there is a polynomial reduction from the partition problem involving an ESP graph -without multiedges- of max degree 3 [14]). We show here its pseudopolynomiality for bounded k and Δ.

Proposition 6. *RSP can be solved in time* $O((n+m)2^{\Delta(k+1)}((n-1)u_{\max})^{k+1})$ *in graphs of treewidth* k *and max degree* Δ, *where* $u_{\max} = \max_{(i,j)\in A} u_{ij}$.

Proof. Let $G = (V, A)$ denote a directed graph with a source node s and a sink node t, and let $G' = (V, E)$ denote the simple undirected graph obtained from G by removing orientation of edges and by simplifying multiedges. Solving RSP in G amounts to solve the following integer linear program (ILP) [11]:

$$\min \sum_{(i,j)\in A} u_{ij} y_{ij} - x_t \tag{7}$$

$$\text{s.t. } x_j \leq x_i + l_{ij} + (u_{ij} - l_{ij}) y_{ij} \quad \forall (i,j) \in A, \tag{8}$$

$$\sum_{(j,k)\in A} y_{jk} - \sum_{(i,j)\in A} y_{ij} = \begin{cases} 1 & \text{if } j = s \\ -1 & \text{if } j = t \\ 0 & \text{if not} \end{cases} \quad \forall j \in V, \tag{9}$$

$$x_s = 0, \ y_{ij} \in \{0,1\} \ \forall (i,j) \in A, \ x_j \in \mathbb{N} \ \forall j \in V. \tag{10}$$

The *interaction graph* of an ILP includes a vertex for each variable of the program and an edge between two vertices if both corresponding variables appear in the same constraint. We now show that the program is solvable in pseudopolynomial time by applying a dynamic programming technique on a *tree decomposition* of the interaction graph $IG = (I, U)$, *i.e.* a labeled tree (T, L) such that (a) every node t of T is labeled by a non-empty subset $L(t)$ of V s.t. $\cup_{t \in T} L(t) = V$, (b) for every edge $\{i, j\} \in U$ there is a node t of T whose label $L(t)$ contains both i and j, (c) for every vertex $i \in I$ the nodes of T whose labels include i form a connected subtree of T. The *width* of a tree decomposition is $\max_{t \in T} |L(t)| - 1$. The *treewidth* of IG is the smallest k for which IG has a tree decomposition of width k. If the treewidth of a graph is bounded by a constant k, then a tree decomposition of treewidth at most k can be constructed in linear time (in the number of nodes) [7]. This tree decomposition can itself be converted in linear time in a *nice tree decomposition* of the same width, *i.e.* a rooted tree decomposition such that each node has at most two children, with four types of nodes t: *leaf nodes* with $|L(t)| = 1$, *join nodes* with two children t', t'' s.t. $L(t) = L(t') = L(t'')$, *introduce nodes* with one child t' s.t. $L(t') = L(t) \cup \{v\}$ for some $v \in V$, *forget nodes* with one child t' s.t. $L(t) = L(t') - \{v\}$ for some $v \in V$. The proof of pseudopolynomiality of the approach is in three steps: (i) we show that if the max degree of G and the treewidth of G' are bounded by some constant, then the treewidth of IG is bounded by some constant; (ii) we show how to solve by dynamic programming an ILP whose IG has a bounded treewidth; (iii) we show that the previous approach is pseudopolynomial since variables x_j are upper bounded by $(n-1)u_{\max}$, where $u_{\max} = \max_{(i,j) \in A} u_{ij}$.

Proof of (i). Assume that G' has treewidth k and G has max degree Δ. Note that IG restricted to constraints (9) is the *line graph* of G, *i.e.*, the graph where each vertex represents an edge of G and any two vertices are adjacent iff their corresponding edges are incident. It can be shown that the treewidth of the line graph is at most $\Delta(k+1) - 1$ [2]. Assuming (T, L) is a tree decomposition of width k of G', the idea is to consider the labeled tree (T, L') where $L'(t)$ is the set of edges of G incident to some node in $L(t)$. Indeed, one can show that (T, L') is then a tree decomposition of the line graph [2]. We now show that $(T, L \cup L')$ is a tree decomposition of IG (where we identify a vertex or an edge of G with the corresponding variable in the ILP). For this purpose, one can consider the following partitions of I and U: $I = X \cup Y$, where $X = \{x_j : j \in V\}$ and $Y = \{y_{ij} : (i,j) \in A\}$, and $U = U_X \cup U_Y \cup U_{XY}$, where $U_X = \{[x_i, x_j] : (i,j) \in A\}$, $U_Y = \{[y_{jk}, y_{ij}] : (i,j) \in A, (j,k) \in A\}$ and $U_{XY} = \{[x_i, y_{ij}], [x_j, y_{ij}] : (i,j) \in A\}$. Condition (a) holds since $\cup_{t \in T} L(t) = X$ and $\cup_{t \in T} L'(t) = Y$. Conditions (b) and (c) hold for edges of U_X and for vertices in X since (T, L) is a tree decomposition of G'. They also hold for edges of U_Y and for vertices in Y since (T, L') is a tree decomposition of the line graph. Besides, condition (b) holds for edges of U_{XY} by construction of L'. Hence, $(T, L \cup L')$ is a tree decomposition of IG. Furthermore, the treewidth of IG is upper bounded by $\max_{t \in T} L(t) + \max_{t \in T} L'(t) - 1 = k + \Delta(k+1)$.

Proof of (ii). By using a method related to non-serial dynamic programming [6], we now show how to solve an ILP in the following general form:

$$(P)\begin{cases}\min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \,\mathcal{R}_i\, b_i \text{ where } \mathcal{R}_i \in \{\leq, =, \geq\} \quad \forall i \leq m \\ x_j \in D_j \quad \forall j \leq n\end{cases}$$

For this purpose, let us introduce the notion of subprogram of an ILP. For each node t of T, $P(t)$ denotes the subprogram of P restricted to the variables whose indices belong to $D(t) = \bigcup_{t'} L(t')$ for $t' = t$ or t' a descendant of t:

$$(P(t))\begin{cases}\min \sum_{j\in D(t)} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \,\mathcal{R}_i\, b_i \quad \forall i : [\forall j,\ (a_{ij} \neq 0 \Rightarrow j \in D(t))] \\ x_j \in D_j,\ \forall j \in D(t)\end{cases}$$

Given $t \in T$ and $\sigma : L(t) \rightarrow \Pi_{j\in L(t)} D_j$ an assignment of values to variables of $L(t)$, we denote by $R_t(\sigma)$ the minimum value of a feasible solution x of $P(t)$ under the constraint $x_j = \sigma(j)\ \forall j \in L(t)$. One sets $R_t(\sigma) = +\infty$ if no feasible solution of $P(t)$ is compatible with σ. The dynamic programming algorithm consists of traversing the nice tree decomposition in a bottom up manner, and computing recursively the tables R_t for each $t \in T$, where table R_t has an entry $R_t(\sigma)$ for each possible assignment σ: let t be a leaf node, say $L(t) = \{j\}$, then $R_t(\sigma) = c_j\sigma(j)$; let t be a join node with two children t' and t'', then $R_t(\sigma) = R_{t'}(\sigma) + R_{t''}(\sigma) - \sum_{j\in L(t)} c_j\sigma(j)$; let t be an introduce node, say $L(t) = L(t') \cup \{j\}$, then $R_t(\sigma) = +\infty$ if σ violates a constraint of $P(t)$, otherwise $R_t(\sigma) = R_{t'}(\sigma_{t'}) + c_j\sigma(j)$ where $\sigma_{t'}$ denotes assignment σ restricted to the variables in $L(t')$; let t be a forget node, say $L(t) = L(t') - \{j\}$, then $R_t(\sigma) = \min_{d_j\in D_j}\{R_{t'}(\sigma') : \sigma'(k) = \sigma(k)\ \forall k \neq j \text{ and } \sigma'(j) = d_j\}$. The optimum is $\min_\sigma R_r(\sigma)$ at the root node r of the nice tree decomposition.

Proof of (iii). We have $|I| = n + m$ since there are n x_i's and m y_{ij}'s in the ILP formulation of RSP. There are therefore $O(n + m)$ nodes in the nice tree decomposition. Noticing that a table R_t can be computed in time $O(2^{\Delta(k+1)}((n-1)u_{\max})^{k+1})$ since there are at most $\Delta(k+1)$ boolean variables and $k+1$ integer variables in $L(t)$, the result follows. □

This approach based on properties of the interaction graph of an ILP formulation is quite general, and can be also fruitfully applied to RBPM. As in Section 2, for any instance of RBPM, we assume that there exists a perfect matching.

Proposition 7. *RBPM can be solved in time* $O((n+m)2^{\Delta(k+1)}((n+1)u_{\max})^{k+1})$ *in graphs of treewidth* k *and max degree* Δ*, where* $u_{\max} = \max_{(i,j)\in E} u_{ij}$.

6 Concluding Remarks

Several results given in this paper deserve to our opinion further research. For instance, we conjecture that RSP, as well as other problems, can be pseudopoly-

nomially solved in graphs with bounded treewidth (without any degree restriction). Alternatively, devising a general method for solving in polynomial time any problem with bounded minmax regret could be very appealing, but the existence of such a method seems quite hypothetical to us.

Besides, the issue we considered here can also be investigated in the discrete scenario model. In that model, each edge e is valued by $(s_1^e, \cdots, s_b^e)$. For example, the robust shortest path and spanning tree problems can be trivially solved under the minmax criterion when the set of valuations is comonotone, i.e. $s_i^e \leq s_j^e \Rightarrow s_i^f \leq s_j^f$ for any i, j and e, f. Indeed, the value of every solution is maximized under the same scenario. Then, one can measure the distance from comonotony as the minimum number of edges the removal of which leads to a comonotone instance. Interestingly enough, it can be shown that, even if the distance from comonotony is 1, and even if there are only 2 scenarios, the robust shortest path and minimum spanning tree problems are NP-hard.

References

1. Aron, I.D., Van Hentenryck, P.: On the complexity of the robust spanning tree problem with interval data. Operations Research Letters 32, 36–40 (2004)
2. Atserias, A.: On digraph coloring problems and treewidth duality (2006), www.lsi.upc.es/~atserias/
3. Averbakh, I.: On the complexity of a class of combinatorial optimization problems with uncertainty. Mathematical Programming Ser. A 90, 263–272 (2001)
4. Averbakh, I., Lebedev, V.: Interval data minmax regret network optimization problems. Discrete Applied Mathematics 138, 289–301 (2004)
5. Bein, W.W., Kamburowski, J., Stallmann, M.F.M.: Optimal reduction of two-terminal directed acyclic graphs. SIAM J. on Computing 21(6), 1112–1129 (1992)
6. Bertele, U., Brioschi, F.: Nonserial Dynamic Programming. Academic Press, London (1972)
7. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. on Computing 25(6), 1305–1317 (1996)
8. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
9. Escoffier, B., Monnot, J., Spanjaard, O.: Some tractable instances of interval data minmax regret problems: bounded distance from triviality. Technical Report 265, Cahiers de recherche, LAMSADE (2007), http://www.lamsade.dauphine.fr/cahiers/PDF/cahierLamsade265.pdf
10. Guo, J., Hüffner, F., Niedermeier, R.: A structural view on parameterizing problems: Distance from triviality. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 162–173. Springer, Heidelberg (2004)
11. Karasan, O.E., Pinar, M.C., Yaman, H.: The robust shortest path problem with interval data. Technical report, Bilkent Univ., Dpt. of Industrial Engineering (2001)
12. Kasperski, A., Zielinski, P.: Minimizing maximal regret in the linear assignment problems with interval costs. Technical Report 007, Instytut Matematyki Wroclaw (2004)
13. Kasperski, A., Zielinski, P.: An approximation algorithm for interval data minmax regret combinatorial optimization problems. Information Processing Letters 97, 177–180 (2006)

14. Kasperski, A., Zielinski, P.: The robust shortest path problem in series-parallel multidigraphs with interval data. Operations Research Letters 34, 69–76 (2006)
15. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, Dordrecht (1997)
16. Paz, A., Moran, S.: Non deterministic polynomial optimisation problems and their approximation. Theoretical Computer Science 95, 251–277 (1981)
17. Valdes, J., Tarjan, R., Lawler, E.: The recognition of series-parallel digraphs. SIAM J. on Computing 11(2), 298–313 (1982)

Assisted Problem Solving and Decompositions of Finite Automata

Peter Gaži and Branislav Rovan*

Department of Computer Science, Comenius University
Mlynská dolina, 842 48, Bratislava, Slovakia
{gazi,rovan}@dcs.fmph.uniba.sk

Abstract. A study of assisted problem solving formalized via decompositions of deterministic finite automata is initiated. The landscape of new types of decompositions of finite automata this study uncovered is presented. Languages with various degrees of decomposability between undecomposable and perfectly decomposable are shown to exist.

1 Introduction

In the present paper we initiate the study of *assisted problem solving*. We intend to model and study situations, where solution to the problem can be sought based on some additional a priori information about the inputs. One can expect to obtain simpler solution in such case. There are similar approaches known in the literature, most notably the notions of advice functions [1], where the additional information is based on the length of the input word and the notion of promise problems [2], where the set of inputs is separated into three classes – those with "yes" answer, those with "no" answer and those where we do not care about the outcome. By considering the simplest case where the "problem solving" machinery is the deterministic finite automaton (DFA) we obtain a new motivation for studying new types of finite automata decompositions.

In this paper we shall thus consider the case where solving a problem shall mean constructing an automaton for a given language L. The "assistance" shall be given by additional information about the input, e.g., that we can assume the inputs shall be restricted to words from a particular regular language L'. Thus, instead of looking for an automaton A such that $L = L(A)$ we can look for a (possibly simpler) automaton B such that $L = L(B) \cap L'$. We can then say that B accepts L with the assistance of L'. We shall call L' (or the corresponding automaton A' such that $L' = L(A')$) an *advisor* to B. In this case the advisor A' provides assistance to the solver B by guaranteeing that A' accepts the given input word. We shall also study a case where the assistance provides more detailed information about the outcome of the computation of A' on the input word (e.g., the state reached). Clearly the advisor can be considered useful only if it enables B to be simpler than A and at the same time A' is not more

* This work was supported in part by the grant VEGA 1/3106/06.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 292–303, 2008.

complicated than A. The measure of complexity we shall consider is the number of states of the deterministic finite automaton. This measure of complexity was used quite often recently due to renewed interest in finite automata prompted by applications such as model checking (see e.g. [3] for a recent survey). (Note that results complementary to ours, namely results on complexity of automata for the intersection of regular sets were studied in [4].)

The contribution of our paper is twofold. First, we can interpret the 'solver' and the 'advisor' as two parallel processes each performing a *different* task and jointly solving a problem. Since our approach lends itself to a generalization to k advisors it may stimulate new parallel solutions to problems (the traditional ones usually using parallel processes to perform essentially the same task). Second, the choice of finite automata as the simplest problem solving machinery brought about new types of decompositions motivated by the information the 'advisor' can provide to the 'solver'. Our results provide a complete picture of the landscape of these decompositions.

The problem within this scenario we shall address in this paper is the existence of a useful advisor for a given automaton A. We shall compare the power of several types of advisors, and investigate the effect of the advisor on the complexity of the assisted solver B. We can formulate this also as a problem of decomposition of deterministic finite state automata – given DFA A find DFA A_1 (a solver) and A_2 (an advisor) such that $w \in L(A)$ can be determined from the computations of A_1 and A_2. We shall study several new types of decompositions of DFA, one of them is analogous to the state behavior decomposition of finite state transducers studied in [5]. In Sect. 3 we prove relations among these decompositions. For each type of decomposition there are automata which are undecomposable and automata for which there is a decomposition that is the best possible. In Sect. 4 we consider the space between these extreme points and study the degree of decomposability.

2 Definitions and Notation

We shall use standard notions of the theory of formal languages (see e.g. [6]). Our notation shall be as follows. Σ^* denotes the set of all words over the alphabet Σ, the length of a word w is denoted by $|w|$, ε denotes the empty word, and for a language L we shall denote by Σ_L the minimal alphabet such that $L \subseteq \Sigma_L^*$. The number of occurrences of a given letter a in a word w is denoted by $\#_a(w)$. Throughout this paper we shall consider deterministic finite automata only.

A *deterministic finite automaton* (DFA) is a quintuple $(K, \Sigma, \delta, q_0, F)$, such that K is a finite set of states, Σ is a finite input alphabet, $q_0 \in K$ is the initial state, $F \subseteq K$ is the set of accepting states and $\delta\colon K \times \Sigma \to K$ is a transition function. As usual, we shall denote by δ also the standard extension of δ to words, i.e., $\delta\colon K \times \Sigma^* \to K$. We shall denote by $|K|$ the number of states in K.

Formalizing the notions of assisted problem solving from the Introduction we shall now define several types of decompositions of DFA A into two (simpler) DFAs A_1 and A_2 (a solver and an advisor) so that the membership of an input

word w in $L(A)$ can be determined based on the information on the computations of A_1 and A_2 on w.[1]

We first introduce an *acceptance-identifying* decomposition of deterministic finite automata.

Definition 1. *A pair of DFAs* (A_1, A_2)*, where* $A_1 = (K_1, \Sigma, \delta_1, q_1, F_1)$ *and* $A_2 = (K_2, \Sigma, \delta_2, q_2, F_2)$*, forms an* acceptance-identifying decomposition *(AI-decomposition) of a DFA* $A = (K, \Sigma, \delta, q_0, F)$*, if* $L(A) = L(A_1) \cap L(A_2)$*. This decomposition is* nontrivial *if* $|K_1| < |K|$ *and* $|K_2| < |K|$*.*

By decomposing A in this manner, one of the decomposed automata (say A_2) can act as an advisor and narrow down the set of input words for the other one (say A_1), whose task to recognize the words of $L(A)$ may become easier.

Another requirement we could pose on a decomposition is to identify the final state of any computation of the original automaton by only knowing the final states of both corresponding computations of the automata forming the decomposition. This requirement can be formalized as follows.

Definition 2. *A pair of DFAs* (A_1, A_2)*, where* $A_1 = (K_1, \Sigma, \delta_1, q_1, F_1)$ *and* $A_2 = (K_2, \Sigma, \delta_2, q_2, F_2)$*, forms a* state-identifying decomposition *(SI-decomposition) of a DFA* $A = (K, \Sigma, \delta, q_0, F)$*, if there exists a mapping* $\beta\colon K_1 \times K_2 \to K$*, such that it holds* $\beta(\delta_1(q_1, w), \delta_2(q_2, w)) = \delta(q_0, w)$ *for all* $w \in \Sigma^*$*. This decomposition is* nontrivial *if* $|K_1| < |K|$ *and* $|K_2| < |K|$*.*

The third – and the weakest – requirement we pose on a decomposition of a DFA is to require that there must exist a way to determine whether the original automaton would accept some given input word based on knowing the states in which the computations of both decomposition automata have finished.

Definition 3. *A pair of DFAs* (A_1, A_2)*, where* $A_1 = (K_1, \Sigma, \delta_1, q_1, F_1)$ *and* $A_2 = (K_2, \Sigma, \delta_2, q_2, F_2)$*, forms a weak* acceptance-identifying decomposition *(wAI-decomposition) of a DFA* $A = (K, \Sigma, \delta, q_0, F)$*, if there exists a relation* $R \subseteq K_1 \times K_2$ *such that it holds* $R(\delta_1(q_1, w), \delta_2(q_2, w)) \Leftrightarrow w \in L(A)$ *for all* $w \in \Sigma^*$*. This decomposition is* nontrivial *if* $|K_1| < |K|$ *and* $|K_2| < |K|$*.*

Note that in the last two definitions, the sets of accepting states of A_1 and A_2 are irrelevant.

By a decomposability of a regular language L in some way, we shall mean the decomposability of the corresponding minimal automaton over Σ_L.

To be able to compare these new types of decomposition to the *parallel decompositions of state behavior* introduced for sequential machines in [5], we shall redefine them for DFAs.

[1] We keep our terminology close to the original one of [5] since we find it more intuitive for a nonspecialist than the current terminology of automata morphisms and congruences used in the algebraic automata theory. It should facilitate the use of assisted problem solving in different setting while specialists in algebraic automata theory should have no difficulty to follow the decomposition results obtained.

Definition 4. *A DFA* $A' = (K', \Sigma, \delta', q_0', F')$ *is said to* realize the state behavior *of a DFA* $A = (K, \Sigma, \delta, q_0, F)$ *if there exists an injective mapping* $\alpha\colon K \to K'$ *such that*

(i) $(\forall a \in \Sigma)(\forall q \in K); \delta'(\alpha(q), a) = \alpha(\delta(q, a))$,
(ii) $\alpha(q_0) = q_0'$.

Moreover, A' *is said to* realize the state and acceptance behavior *of* A*, if in addition the following property holds:*

(iii) $(\forall q \in K); \alpha(q) \in F' \Leftrightarrow q \in F$.

Definition 5. *The* parallel connection *of two DFA* $A_1 = (K_1, \Sigma, \delta_1, q_1, F_1)$ *and* $A_2 = (K_2, \Sigma, \delta_2, q_2, F_2)$ *is the DFA* $A = A_1||A_2 = (K_1 \times K_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$ *such that* $\delta((p_1, p_2), a) = (\delta_1(p_1, a), \delta_2(p_2, a))$.

Definition 6. *A pair of DFAs* (A_1, A_2) *is a* state behavior (SB-) decomposition *of a DFA* A *if* $A_1||A_2$ *realizes the state behavior of* A*. The pair* (A_1, A_2) *is an* acceptance and state behavior (ASB-) decomposition *of* A *if* $A_1||A_2$ *realizes the state and acceptance behavior of* A*. This decomposition is* nontrivial *if both* A_1 *and* A_2 *have fewer states than* A.

We have modified the definitions to fit the formalism and purpose of deterministic finite automata (i.e., to accept formal languages) without loosing the connection to the strongly related and useful concept of *S.P.partitions*, exhibited below.

We shall use the following notation and properties of S.P. partitions from [5]. A partition π on a set of states of a DFA $A = (K, \Sigma, \delta, q_0, F)$ has *substitution property* (S.P.), if it holds $\forall p, q \in K;\ \ p \equiv_\pi q \Rightarrow (\forall a \in \Sigma; \delta(p, a) \equiv_\pi \delta(q, a))$. If π_1 and π_2 are partitions on a given set M, then

(i) $\pi_1 \cdot \pi_2$ is a partition on M such that $a \equiv_{\pi_1 \cdot \pi_2} b \Leftrightarrow a \equiv_{\pi_1} b \wedge a \equiv_{\pi_2} b$,
(ii) $\pi_1 + \pi_2$ is a partition on M such that $a \equiv_{\pi_1 + \pi_2} b$ iff there exists a sequence $a = a_0, a_1, a_2, \ldots, a_n = b$, such that $a_i \equiv_{\pi_1} a_{i+1} \vee a_i \equiv_{\pi_2} a_{i+1}$ for all $i \in \{0, \ldots, n-1\}$,
(iii) $\pi_1 \preceq \pi_2$ if it holds $(\forall x, y \in M);\ \ x \equiv_{\pi_1} y \Rightarrow x \equiv_{\pi_2} y$.

The set of all partitions on a given set (with the partial order $\preceq$, join realized by + and meet realized by .) forms a lattice. The set of all S.P. partitions on the set of states of a given DFA forms a sublattice of the lattice of all partitions on this set. The trivial partitions $\{\{q_0\}, \{q_1\}, \ldots, \{q_n\}\}$ and $\{\{q_0, q_1, \ldots, q_n\}\}$ shall be denoted by symbols 0 and 1, respectively. The block of a partition π containing the state q shall be denoted by $[q]_\pi$. In addition, we shall use the following separation notion.

Definition 7. *The partitions* $\pi_1 = \{R_1, \ldots, R_k\}$ *and* $\pi_2 = \{S_1, \ldots, S_l\}$ *on a set of states of a DFA* $A = (K, \Sigma, \delta, q_0, F)$ *are said to* separate the final states *of* A *if there exist indices* $i_1, \ldots, i_r$ *and* $j_1, \ldots, j_s$ *such that it holds* $(R_{i_1} \cup \ldots \cup R_{i_r}) \cap (S_{j_1} \cup \ldots \cup S_{j_s}) = F$.

3 Relations between Types of Decompositions

The concept of partitions separating the final states allows us to derive a necessary and sufficient condition for the existence of SB- and ASB-decompositions similar to the one stated in [5].

Theorem 1. *A DFA $A = (K, \Sigma, \delta, q_0, F)$ has a nontrivial SB-decomposition iff there exist two nontrivial S.P. partitions π_1 and π_2 on the set of states of A such that $\pi_1 \cdot \pi_2 = 0$. This decomposition is an ASB-decomposition if and only if these partitions separate the final states of A.*

Proof. The proof is analogous to that in [5] but had to be extended for the ASB-decomposition. We omit it due to space constraints. □

For the other decompositions, we can derive the following sufficient conditions that exploit the concept of S.P. partitions.

Theorem 2. *Let $A = (K, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton, let π_1 and π_2 be nontrivial S.P. partitions on the set of states of A, such that they separate the final states of A. Then A has a nontrivial AI-decomposition.*

Proof. Since π_1 and π_2 separate the final states of A, there exist blocks $B_1, \ldots, B_k$ and $C_1, \ldots, C_l$ of the partitions π_1 and π_2 respectively, such that $(B_1 \cup \ldots \cup B_k) \cap (C_1 \cup \ldots \cup C_l) = F$. We shall construct two automata A_1 and A_2 having states corresponding to blocks of these partitions and show that (A_1, A_2) is a nontrivial AI-decomposition of A. Let $A_1 = (\pi_1, \Sigma, \delta_1, [q_0]_{\pi_1}, \{B_1, \ldots, B_k\})$ and $A_2 = (\pi_2, \Sigma, \delta_2, [q_0]_{\pi_2}, \{C_1, \ldots, C_l\})$ be DFAs with δ_i defined by $\delta_i([q]_{\pi_i}, a) = [\delta(q, a)]_{\pi_i}$, $i \in \{1, 2\}$ (this definition does not depend on the choice of q since π_i is an S.P. partition). We now need to prove that $L(A) = L(A_1) \cap L(A_2)$.

Let $w \in L(A)$. Suppose that the computation of A on the word w ends in some accepting state $q_f \in F$. Then, from the construction of A_1 and A_2 it follows that the computation of A_i on the word w ends in the state corresponding to the block $[q_f]_{\pi_i}$ of the partition π_i. Since $q_f \in F$, it must hold $[q_f]_{\pi_1} \in \{B_1, \ldots, B_k\}$ and $[q_f]_{\pi_2} \in \{C_1, \ldots, C_l\}$, hence from the construction of A_i, these blocks correspond to the accepting states in the respective automata. Thus $w \in L(A_i)$ for $i \in \{1, 2\}$, therefore $L(A) \subseteq L(A_1) \cap L(A_2)$.

Now suppose $w \in L(A_1) \cap L(A_2)$, Thus the computation of A_1 on w ends in one of the states $B_1, \ldots, B_k$, which means that the computation of A on w would end in a state from the union of blocks $B_1 \cup \ldots \cup B_k$. Using the same argument for A_2, we get that the computation of A on w would end in a state from $C_1 \cup \ldots \cup C_l$. Since $(B_1 \cup \ldots \cup B_k) \cap (C_1 \cup \ldots \cup C_l) = F$ we obtain that the computation of A ends in an accepting state, hence $w \in L(A)$ and $L(A_1) \cap L(A_2) \subseteq L(A)$.

Since both partitions are nontrivial, so is the AI-decomposition obtained. □

Theorem 3. *Let $A = (K, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton, let π_1 and π_2 be nontrivial S.P. partitions on the set of states of A, such that $\pi_1 \cdot \pi_2 \preceq \{F, K - F\}$. Then A has a nontrivial wAI-decomposition.*

Proof. We shall construct A_1 and A_2 corresponding to the S.P. partitions π_1 and π_2 as follows: $A_i = (\pi_i, \Sigma, \delta_i, [q_0]_{\pi_i}, \emptyset)$, where $\delta_i([q]_{\pi_i}, a) = [\delta(q, a)]_{\pi_i}$ and $i \in \{1, 2\}$. To show that (A_1, A_2) is a wAI-decomposition of A, we define the relation $R \subseteq \pi_1 \times \pi_2$ by the equivalence $R(D_1, D_2) \Leftrightarrow (D_1 \cap D_2 \subseteq F)$,where D_i is some block of the partition π_i. Now we need to prove that $\forall w \in \Sigma^*$; $w \in L(A) \Leftrightarrow R(\delta_1([q_0]_{\pi_1}, w), \delta_2([q_0]_{\pi_2}, w))$.

Let the computation of A on w end in some state $p \in K$. It follows that the computation of A_i on the word w ends in the state corresponding to the block $[p]_{\pi_i}$, $i \in \{1, 2\}$. Thus $R(\delta_1([q_0]_{\pi_1}, w), \delta_2([q_0]_{\pi_2}, w)) \Leftrightarrow R([p]_{\pi_1}, [p]_{\pi_2})$ and by the definition of R, we have $R(\delta_1([q_0]_{\pi_1}, w), \delta_2([q_0]_{\pi_2}, w)) \Leftrightarrow [p]_{\pi_1} \cap [p]_{\pi_2} \subseteq F$. Since $p \in [p]_{\pi_1} \cap [p]_{\pi_2}$, $[p]_{\pi_1} \cap [p]_{\pi_2}$ is a block of the partition $\pi_1 \cdot \pi_2$ and $\pi_1 \cdot \pi_2 \preceq \{F, K - F\}$, it must hold that either $[p]_{\pi_1} \cap [p]_{\pi_2} \subseteq F$ or $[p]_{\pi_1} \cap [p]_{\pi_2} \subseteq K - F$. Therefore $R(\delta_1([q_0]_{\pi_1}, w), \delta_2([q_0]_{\pi_2}, w)) \Leftrightarrow p \in F$ and the proof is complete. □

It follows directly from the definitions, that each SI-decomposition is also a wAI-decomposition, and so is each AI-decomposition. Also, each ASB-decomposition is an AI-decomposition, which is a consequence of the definition of acceptance and state behavior realization. For minimal automata, a relationship between AI- and SI-decompositions can be obtained.

Theorem 4. *Let $A = (K, \Sigma, \delta, q_0, F)$ be a minimal DFA, let (A_1, A_2) be its AI-decomposition. Then (A_1, A_2) is also an SI-decomposition of A.*

Proof. Since (A_1, A_2) is an AI-decomposition of A, $L(A) = L(A_1) \cap L(A_2)$. Therefore if we use the well-known Cartesian product construction, we obtain the automaton $A_1||A_2$ such that $L(A_1||A_2) = L(A)$. Since A is the minimal automaton accepting the language $L(A)$, there exists a mapping $\beta\colon K' \to K$ such that it holds $(\forall w \in \Sigma^*)$; $\beta(\delta'(q_0', w)) = \delta(\beta(q_0'), w)$, where δ' is the transition function of $A_1||A_2$, K' is its set of states and q_0' is its initial state. Since $A_1||A_2$ is a parallel connection (i.e., $K' = K_1 \times K_2$, q_0' is the pair of initial states of A_1 and A_2), it is easy to see that β is in fact exactly the mapping required by the definition of the SI-decomposition. □

The ASB-decomposition is a combination of the SB-decomposition and the AI-decomposition, as the next theorem shows.

Theorem 5. *Let A be a DFA without unreachable states. (A_1, A_2) is an ASB-decomposition of A iff (A_1, A_2) is both an SB-decomposition and an AI-decomposition of A.*

Proof. The first implication clearly follows from the definitions, Theorem 1 and Theorem 2. Now let (A_1, A_2) be an SB- and AI-decomposition of $A = (K, \Sigma, \delta, q_0, F)$. Let α be the mapping given by the definition of SB-decomposition. We need to prove that for all states q of A, $q \in F$ iff $\alpha(q) \in F_1 \times F_2$, where F_i is the set of accepting states of A_i, $i \in \{1, 2\}$. Let $q \in K$ and let w be a word such that $\delta(q_0, w) = q$. Then $q \in F \Leftrightarrow w \in L(A) \Leftrightarrow w \in L(A_1) \cap L(A_2) \Leftrightarrow \alpha(q) \in F_1 \times F_2$, where the first equivalence is implied by the choice of w, the second holds because (A_1, A_2) is an AI-decomposition and the third is a consequence of the properties of α guaranteed by the SB-decomposition definition. □

There is also a relationship between SB- and SI-decompositions, in fact SB- is a stronger version of the state-identifying decomposition, as the following two theorems show. We need the notion of reachability on pairs of states.

Definition 8. *Let* $A_1 = (K_1, \Sigma, \delta_1, p_1, F_1)$ *and* $A_2 = (K_2, \Sigma, \delta_2, p_2, F_2)$ *be DFAs. We shall call a pair of states* $(q, r) \in K_1 \times K_2$ reachable*, if there exists a word* $w \in \Sigma^*$ *such that* $\delta_1(p_1, w) = q$ *and* $\delta_2(p_2, w) = r$.

Theorem 6. *Let* $A = (K, \Sigma, \delta, q_0, F)$ *be a DFA and let* (A_1, A_2) *be its SB-decomposition. Then* (A_1, A_2) *also forms an SI-decomposition of* A.

Proof. Let $A_i = (K_i, \Sigma, \delta_i, q_i, F_i), i \in \{1, 2\}$. Since (A_1, A_2) is an SB-decomposition of A, there exists an injective mapping $\alpha\colon K \to K_1 \times K_2$ such that it holds $\alpha(q_0) = (q_1, q_2)$ and $(\forall a \in \Sigma)(\forall p \in K); \alpha(\delta(p, a)) = (\delta_1(p_1, a), \delta_2(p_2, a))$, where $\alpha(p) = (p_1, p_2)$. Let us define a new mapping $\beta\colon K_1 \times K_2 \to K$ by

$$\beta(p_1, p_2) = \begin{cases} p & \text{if } \exists p \in K, \alpha(p) = (p_1, p_2) \\ q_0 & \text{otherwise.} \end{cases} \tag{1}$$

Since α is injective, there exists at most one such p and this definition is correct.

We now need to prove that β satisfies the condition from the definition of SI-decomposition, i.e., that $(\forall w \in \Sigma^*);\ \beta(\delta_1(q_1, w), \delta_2(q_2, w)) = \delta(q_0, w)$. Since $\alpha(q_0) = (q_1, q_2)$ and all the pairs of states we encounter in the computation of $A_1||A_2$ are thus reachable, this follows from the definition of α and (1) by an easy induction. □

Lemma 1. *Let* A *be a DFA without unreachable states and let* (A_1, A_2) *be its SI-decomposition, with* β *being the corresponding mapping. Then* (A_1, A_2) *is an SB-decomposition of* A *if and only if* β *is injective on all reachable pairs of states.*

Proof. Let (A_1, A_2) be an SB-decomposition of A. It clearly follows from Definition 2, that the corresponding β satisfies the equation (1) in the proof of Theorem 6 on all reachable pairs of states. Since the mapping α is a bijection between the set of states of A and the set of all reachable pairs of states of A_1 and A_2, β defined as its inverse on the set of reachable pairs of states will be injective on this set.

For the other implication, let (A_1, A_2) be an SI-decomposition of A and let β be injective on the set of reachable pairs of states, let β_r denote the mapping β restricted onto the set of all reachable pairs of states of A_1, A_2. Since A has no unreachable states, β_r is also surjective, thus we can define a new mapping $\alpha\colon K \to K_1 \times K_2$ by the equation $\alpha(q) = \beta_r^{-1}(q)$. Since β maps the initial state onto the initial state, so does α, and since β satisfies the condition from the Definition 2, it implies that also α satisfies the condition (i) from the definition of realization of state behavior. Therefore (A_1, A_2) is an SB-decomposition of A, with the corresponding mapping α. □

The converse of Theorem 6 does not hold. The minimal automaton for the language $L = \{a^{4k}b^{4l} | k \geq 0, l \geq 1\}$ gives a counterexample. Inspecting its S.P.

partitions shows that it has no nontrivial SB-decomposition, but it can be AI-decomposed into minimal automata for languages $L_1 = \{a^{4k}b^l | k \geq 0, l \geq 1\}$ and $L_2 = \{w | \#_b(w) = 4l; l \geq 0\}$. According to Theorem 4, this AI-decomposition is also state-identifying.

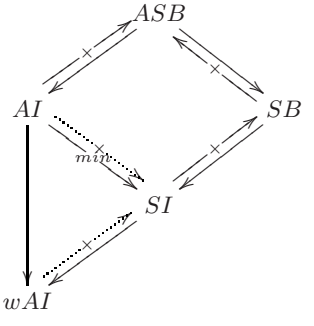
Each ASB-decomposition is obviously also an SB-decomposition. On the other hand, there exist SB-decomposable automata, that are ASB-undecomposable. For example, the minimal automaton for the language

$$\begin{aligned} L_1 = \{w \in \{a,b,c\}^* | \#_a(w) \mod 3 = 0 \wedge \#_b(w) \mod 5 = 0\} \\ \cup \{w \in \{a,b,c\}^* | \#_a(w) \mod 3 = 2 \wedge \#_b(w) \mod 5 = 4\} \end{aligned}$$

has this property, because the corresponding S.P. partitions on the set of its states do not separate the final states in the sense of Definition 7.

It is also not so difficult to see that for any non-minimal automaton A without unreachable states, there exists a nontrivial AI- and wAI-decomposition (A_1, A_2) such that A_1 is the minimal automaton equivalent to A and A_2 has only one state. This decomposition is obviously not state-identifying.

Figure 1 summarizes all the relationships among the decomposition types that we have shown so far.



Description:

- $A \longrightarrow B$: every A-decomposition is also a B-decomposition
- $A \xrightarrow{min} B$: every A-decomposition of a minimal DFA is also a B-decomposition
- $A \dashrightarrow B$: not every A-decomposition is also a B-decomposition
- $A \xrightarrow{\times} B$: there exists a DFA that has a nontrivial A-decomposition but does not have a nontrivial B-decomposition

Fig. 1. Relationships between decomposition types of DFA

Now we show that for the case of so-called perfect decompositions, some of the types of decomposition mentioned coincide.

Definition 9. *Let t be a type of decomposition, $t \in \{ASB, SB, AI, SI, wAI\}$. Let A be a DFA having n states, let A_1 and A_2 be DFAs having k and l states, respectively. We shall call the pair (A_1, A_2) a* perfect t-decomposition *of A, if it forms a t-decomposition of A and $n = k \cdot l$.*

Theorem 7. *Let A be a DFA with no unreachable states and let (A_1, A_2) be a pair of DFAs. Then (A_1, A_2) forms a perfect SI-decomposition of A iff (A_1, A_2) forms a perfect SB-decomposition of A.*

Proof. One of the implications is a consequence of Theorem 6. As to the second one, since (A_1, A_2) forms a perfect SI-decomposition of A, each of the pairs of states of A_1 and A_2 is reachable and each pair has to correspond to a different state of A in the mapping β, therefore β is bijective and the theorem follows from Lemma 1. □

Corollary 1. *Let A be a minimal DFA and let (A_1, A_2) be a pair of DFAs. Then (A_1, A_2) forms a perfect AI-decomposition of A iff (A_1, A_2) forms a perfect ASB-decomposition of A.*

Proof. The claim follows from Theorem 5, Theorem 4 and Theorem 7. □

As a consequence of these facts, we can use the necessary and sufficient conditions stated in Theorem 1 to look for perfect AI- and SI-decompositions.

Now, let us inspect the relationship between decompositions of an automaton and the decompositions of the corresponding minimal automaton.

Theorem 8. *Let $A = (K, \Sigma, \delta, q_0, F)$ be a DFA and let $A_{\min}$ be a minimal DFA such that $L(A) = L(A_{\min})$. Let (A_1, A_2) be an SI-decomposition (AI-decomposition, wAI-decomposition) of A, then (A_1, A_2) also forms a decomposition of $A_{\min}$ of the same type.*

Proof. First, note that this theorem does not state that any of the decompositions is nontrivial. To prove the statement for SI-decompositions, suppose that (A_1, A_2) is an SI-decomposition of A, thus there exists a mapping $\alpha\colon K_1 \times K_2 \to K$ such that it holds $(\forall w \in \Sigma^*)$; $\alpha(\delta_1(q_1, w), \delta_2(q_2, w)) = \delta(q_0, w)$, where δ_i and q_i are the transition function and the initial state of the automaton A_i. Since $A_{\min}$ is the minimal automaton corresponding to A, there exists some mapping $\beta\colon K \to K_{\min}$ such that $(\forall w \in \Sigma^*)$; $\beta(\delta(q_0, w)) = \delta_{\min}(\beta(q_0), w)$, where $\delta_{\min}$ is the transition function of $A_{\min}$ and $K_{\min}$ is the set of states of $A_{\min}$. By the composition of these mappings we obtain the mapping $\beta \circ \alpha\colon K_1 \times K_2 \to K_{\min}$, which combines A_1 and A_2 into $A_{\min}$ in the way that the definition of SI-decomposition requires. For both the AI- and the wAI-decomposition, this statement is trivial, since $L(A) = L(A_{\min})$. □

Based on the above theorem it thus suffices to inspect the SI- (AI-, wAI-) decomposability of the minimal automaton accepting a given language, and if we show its undecomposability, we know that the recognition of this language cannot be simplified using an advisor of the respective type. However, this does not hold for SB- and ASB-decompositions, this can be exhibited by the following example. Let us consider the language $L = \{a^{2k}b^{2l} | k \geq 0, l \geq 1\}$. The minimal automaton $A_{\min} = (K, \Sigma_L, \delta, a_0, \{a_0, b_0\})$ has its transition function defined by the first transition diagram in Fig. 2. We can easily show that this automaton does not have any nontrivial SB- (and thus neither ASB-) decomposition by enumerating its S.P. partitions.

Now let us examine the automaton $A' = (K', \Sigma_L, \delta', a_0, \{a_0, b_0\})$ with the transition function δ' defined by the second transition diagram in Fig. 2. Clearly, $L(A') = L(A_{\min})$, but by inspecting the lattice of S.P. partitions of A', we can find

Fig. 2. Transition functions of $A_{\min}$ and A'

the pair $\pi_1 = \{\{a_0\}, \{a_1\}, \{b_0, b_1\}, \{R_0, R_1\}\}$ and $\pi_2 = \{\{a_0, a_1, b_0, R_0\}, \{b_1, R_1\}\}$ such that $\pi_1 \cdot \pi_2 = 0$ and they separate the final states of A'. By Theorem 1 we can use these partitions to construct a nontrivial ASB- (and thus also SB-) decomposition of A' formed by the automata A_1 and A_2 having two and four states, respectively. Note that both A_1 and A_2 have less states than $A_{\min}$.

In the following theorem (inspired by a similar theorem in [5]) we state a condition, under which the situation from the last example cannot occur, i.e., under which any SB-decomposition of a DFA implies a (maybe simpler) SB-decomposition of the equivalent minimal DFA.

Theorem 9. *Let $A = (K, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton and let $A_{\min} = (K_{\min}, \Sigma, \delta_{\min}, q_{\min}, F_{\min})$ be the minimal DFA such that $L(A) = L(A_{\min})$. Let (A_1, A_2) be a nontrivial SB-decomposition of A consisting of automata having k and l states. If the lattice of S.P. partitions of A is distributive, then there exists an SB-decomposition of $A_{\min}$ consisting of automata having k' and l' states, such that $k' \leq k$ and $l' \leq l$.*

Proof. Since $A_{\min}$ is the minimal DFA such that $L(A) = L(A_{\min})$, there exists a mapping $f\colon K \to K_{\min}$ such that $(\forall w \in \Sigma^*)$; $f(\delta(q_0, w)) = \delta_{\min}(q_{\min}, w)$. Using the mapping f, let us define a partition ρ on the set of states of A by $p \equiv_\rho q \Leftrightarrow f(p) = f(q)$. Clearly, ρ is an S.P. partition.

Since (A_1, A_2) is a nontrivial SB-decomposition of A, we can use it to obtain S.P. partitions π_1 and π_2 on the set of states of A such that $\pi_1 \cdot \pi_2 = 0$. Let us define new partitions π_1' and π_2' on the set of states of $A_{\min}$ by $f(p) \equiv_{\pi_i'} f(q) \Leftrightarrow p \equiv_{\rho+\pi_i} q$. Since it holds that $\rho \preceq \rho + \pi_i$, this definition does not depend on the choice of the states p and q. It holds that $|\pi_i'| = |\rho + \pi_i| \leq |\pi_i|$, therefore if we prove that π_1' and π_2' are S.P. partitions and $\pi_1' \cdot \pi_2' = 0$, we can use them to construct the desired decomposition.

The fact that π_i' is an S.P. partition on the set of states of $A_{\min}$ is a trivial consequence of the fact that $\rho + \pi_i$ is an S.P. partition on the set of states of A. We need to prove that $\pi_1' \cdot \pi_2' = 0$. Let us assume that p' and q' are states of $A_{\min}$ such that $p' \equiv_{\pi_1' \cdot \pi_2'} q'$ and p, q are some states of A such that $f(p) = p'$ and $f(q) = q'$. Then $p' \equiv_{\pi_1'} q'$ and $p' \equiv_{\pi_2'} q'$, and by definition of π_i' we get $p \equiv_{\rho+\pi_1} q$ and $p \equiv_{\rho+\pi_2} q$, which is equivalent to $p \equiv_{(\rho+\pi_1)\cdot(\rho+\pi_2)} q$. Since the lattice of all S.P. partitions of A is distributive, we have $(\rho+\pi_1)\cdot(\rho+\pi_2) = \rho+(\pi_1 \cdot \pi_2) = \rho + 0 = \rho$, therefore $p \equiv_\rho q$, which by definition of ρ implies that $f(p) = f(q)$, in other words $p' = q'$. Hence $\pi_1' \cdot \pi_2' = 0$. □

4 Degrees of Decomposability

It is easy to see that for each type of decomposition, there exist undecomposable regular languages (e.g. $L^{(n)} = \{a^k | k \geq n-1\}$ is wAI-undecomposable for each $n \in \mathbb{N}$). There also exist regular languages, that are perfectly decomposable in each way (e.g. $L^{(k,l)} = \{w \in \{a,b\}^* | \#_a(w) \bmod k = 0 \wedge \#_b(w) \bmod l = 0\}$ has a perfect ASB-decomposition for all $k, l \geq 2$). We shall now investigate whether all values between these two limits can be achieved.

Definition 10. *Let A be a DFA, let (A_1, A_2) be its nontrivial SB- (ASB-) decomposition with the corresponding S.P. partitions π_1 and π_2. We shall call this decomposition* redundant*, if there exist S.P. partitions $\pi_1' \succeq \pi_1$ and $\pi_2' \succeq \pi_2$ such that at least one of these inequalities is strict, but it still holds $\pi_1' \cdot \pi_2' = 0$ (and π_1' and π_2' separate the final states of A).*

Lemma 2. *For each $r, s \in \mathbb{N}$, $r, s \geq 2$, there exists a minimal DFA A consisting of $r \cdot s$ states and having only one nontrivial nonredundant SB-decomposition (ASB-decomposition) up to the order of automata, consisting of automata having r and s states.*

Proof (sketch). By a careful analysis of the S.P. partitions of the minimal automaton accepting the regular language $L = \{w \in \{a,b\}^* | \#_a(w) \geq r-1 \wedge \#_b(w) \geq s-1\}$, it can be shown that it has the desired property. □

Definition 11. *Let $A = (K, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton, let $K \cap \{p_0, p_1, \ldots, p_{k-1}\} = \emptyset$ and let c be a new symbol not included in Σ. We shall define a k-*extension *A' of the automaton A by the following construction: $A' = (K \cup \{p_0, p_1, \ldots, p_{k-1}\}, \Sigma \cup \{c\}, \delta', p_0, F)$, where the transition function δ' is defined as follows:*

$$\begin{aligned}
(\forall q \in K)\ (\forall a \in \Sigma); &\quad \delta'(q,a) = \delta(q,a) \\
(\forall q \in K); &\quad \delta'(q,c) = q \\
(\forall p \in \{p_0, p_1, \ldots, p_{k-1}\})\ (\forall a \in \Sigma); &\quad \delta'(p,a) = p \\
(\forall i \in \{0,1,\ldots,k-2\}); &\quad \delta'(p_i,c) = p_{i+1} \\
&\quad \delta'(p_{k-1},c) = q_0.
\end{aligned}$$

Note that a k-extension of a minimal DFA is again a minimal DFA.

Lemma 3. *Let A be a DFA consisting of n states, all of which are reachable. Let A' be its k-extension. Then A has a nontrivial nonredundant SB-decomposition (ASB-decomposition) consisting of automata having r and s states iff A' has a nontrivial nonredundant decomposition of the same type, consisting of automata having $k+r$ and $k+s$ states.*

Proof (sketch). We shall analyze the S.P. partitions of the extended automaton A' and relate them to the S.P. partitions of the original automaton A. It can be shown that any S.P. partition of A' that does not contain a special one-state

block for each of the new extension states cannot distinguish between the old states of A at all, i.e. they all belong to the same block. Therefore, for each pair of partitions inducing a decomposition of A' there must exist a pair of partitions of A that can be obtained by removing all these one-state blocks, and these partitions induce a decomposition of A. On the other hand, each pair of partitions inducing a decomposition of A can be extended by adding new blocks for each of the extension states, obtaining a pair of partitions of A', and thus a decomposition. □

We omit the full proofs of both lemmas due to space limitations, they will appear in the full version of the paper.

We can combine the lemmas to obtain the following theorem.

Theorem 10. *Let $n \in \mathbb{N}$ be such that $n = k + r \cdot s$, where $r, s, k \in \mathbb{N}$, $r, s \geq 2$. Then there exists a minimal DFA A consisting of n states, such that it has only one nontrivial nonredundant SB-decomposition (ASB-decomposition) up to the order of the automata in the decomposition, and this decomposition consists of automata with $k + r$ and $k + s$ states.*

References

1. Balcazar, J.L., Diaz, J., Gabarro, J.: Structural Complexity I. Springer, New York (1988)
2. Even, S., Selman, A.L., Yacobi, Y.: The Complexity of Promise Problems with Applications to Public-Key Cryptography. Information and Control 61(2), 159–173 (1984)
3. Yu, S.: State Complexity: Recent Results and Open Problems. Fundamenta Informaticae 64, 471–480 (2005)
4. Birget, J.C.: Intersection and Union of Regular Languages and State Complexity. Information Processing Letters 43, 185–190 (1992)
5. Hartmanis, J., Stearns, R.E.: Algebraic Structure Theory of Sequential Machines. Prentice-Hall, Englewood Cliffs (1966)
6. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)

Energy-Efficient Windows Scheduling

Christian Gunia*

Dept. of Computer Science, Freiburg University
Georges-Köhler-Allee 79, 79110 Freiburg, Germany
gunia@informatik.uni-freiburg.de

Abstract. A server repeatedly transmits data items (pages) possibly with different speeds on a set of channels. The objective is to minimize energy consumption of the schedule. We adopt the common model that sending at speed s for t time units consumes $t \cdot s^\alpha$ energy for a constant $\alpha \geq 2$. An individual window length is associated with each page. This length is a strict upper bound on the time between two consecutive broadcasts for that page. We present an easy to implement algorithm for the single channel case that obtains an approximation ratio of $3 \cdot 4^\alpha$. For the multi-channel case with identical channels an extension of this algorithm computes an 8^α-approximation. Both algorithms run in low-order polynomial time. As our main tool for the analysis, we show that it suffices to consider periodic schedules as their energy density (total energy consumption per time unit) differs from the one of general schedules at most by $(1 + \varepsilon)$ for an arbitrary constant $\varepsilon > 0$.

1 Introduction

Recent years have witnessed a growing interest in solving tasks in an energy-efficient manner. This development can be observed in a variety of research areas ranging from CPU architecture to job scheduling to information transmission in wireless networks. Classical objectives in computer science mostly focus on achieving a solution that is best possible in terms of convenience for the serviced clients. However, they ignore the cost incurred by realizing this solution. While this approach is reasonable in many classical research areas, taking this realization cost into account has significantly helped establishing new application areas. In most of these applications energy is a critical resource and must not be neglected. Thus, a model has to reflect the underlying computing machinery appropriately. We briefly introduce two of these applications.

Multi-agent systems. In the growing field of multi-agent systems self-sustaining mobile agents cooperate to solve a common task [1]. In many applications those mobile agents wish to access information another agent has gathered during its operation. This information is typically transmitted by wireless broadcasts. Due to characteristics of their operation areas these agents tend to be small and their memory storage is limited. Consequently, not every agent stores all received

* Supported by DFG research training program No 1103 'Embedded Microsystems'.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 304–315, 2008.

data and relies on a transmission of the required information at the right time. It is reasonable to assume that this information does not have to be delivered instantly but the agent's waiting time is bounded from above by some reasonable but predetermined limit. This limit must not be exceeded in order to guarantee a smooth cooperation. As the agents also carry a limited power supply they use specialized hardware like low-power CPUs to reduce their energy consumption. For this reason the energy spent in establishing the communication link is far from being negligible. With decreasing transmission speed the signal-to-noise ratio needed to receive the transmission decreases as well [2]. We propose to adjust the transmission speed in order to save energy. The agents can reduce their energy consumption by keeping the transmission speed low.

Smart Sensor Nodes. A problem quite similar to the information distribution problem in multi-agent systems arises from signal pre-processing on smart sensor nodes. A smart sensor node pre-processes the sensor signal before it is made available for other agents [3]. These nodes often carry a limited power supply and reduce the processor speed in order to save energy. On the other hand most applications enforce a fixed sampling rate, i.e., the period between two samples must not exceed some given limit. Therefore, the node has to adjust its computational power to guarantee this property. As the sensor signal has to be consistent, preemption during the pre-processing is not allowed. In highly integrated sensor nodes many different sensor signals are pre-processed on one single processor to keep hardware expenses as low as possible.

1.1 Model and Notation

Inspired by these applications we investigate the following problem. A server stores *data items* $1, 2, \ldots, n$ and broadcasts them on *channels* $1, 2, \ldots, m$. These data items are called *pages* for the remainder of this work. The channels are *speed-controlled*, i.e., the server can choose the speed of each transmission independently. Let $\alpha \geq 2$ be a constant. We follow the standard model [4] assuming that a broadcast at speed s for t time units consumes $t \cdot s^\alpha$ energy. All pages are of unit size, i.e., each one can be transmitted in t time units at speed $1/t$ while consuming $(1/t)^{\alpha-1}$ energy. Due to the convexity of the energy function, it is not difficult to see that this is the minimal amount of energy needed to deliver the whole page within t time units. A *broadcast schedule* is a sequence of broadcasts for each channel. Each page p is associated with an individual *window length* w_p that has to be met. For a given broadcast schedule we say that the window length of page p is met if within every time window of length w_p page p is broadcasted at least once *uninterruptedly*. Without loss of generality we assume the window lengths to be ordered non-decreasingly for the rest of this work, i.e., $w_1 \leq w_2 \leq \cdots \leq w_n$. If all window lengths are met we call the schedule feasible. Note that every feasible schedule is an *infinite* sequence of broadcasts for each channel. Its energy consumption is the sum of the broadcasts' energy consumptions and, therefore, also infinite. We observe that the overall energy consumption is no appropriate measure for the quality of a schedule. Let $E_S(t)$

denote the energy consumed by schedule S up to time t. The *energy density* δ_S of a schedule S is defined by $\delta_S := \limsup_{t\to\infty} E_S(t)/t$. Although this measure seems a bit artifical at first glance, it reduces to a very natural one for an important kind of schedules as we will see in the next paragraph. By minimizing the energy density we maximize the period of time a server can operate with a given amount of energy for most natural schedules. In this context by most natural we mean that the server does not spend all its energy at the beginning of that schedule. In this paper we consider the problem of finding a feasible schedule with minimal energy density.

Although a feasible schedule is implicitly infinite due to the problem definition, it needs to be stored in finite space. To this end we compute a finite representation of that schedule. A representation R of a schedule S is a sequence of broadcasts for each channel such that S corresponds to repeatedly concatenating representation R. We assume that a representation starts at time 0 and define its length to be the ending point of its last broadcast. A schedule is called cyclic if there exists a representation of finite length. Otherwise, it is called acyclic. The period of a schedule is the length of its shortest representation; it is defined to be ∞ for acyclic schedules. We point out that that since $E_S(T)$ is finite the definition of energy density reduces to $\delta_S = E_S(T)/T$ for cyclic schedules where T denotes the period of schedule S. Thus, the energy density denotes the average energy consumption per time unit within one period for cyclic schedules.

We introduce some notation used throughout this work. A broadcast that starts at time t_s, ends at time t_e and broadcasts page p at constant speed is denoted by the triple (t_s, t_e, p). Its length is defined as $t_e - t_s$. For two disjoint broadcasts (t_s, t_e, p) and (t'_s, t'_e, p') with $t_e \leq t'_s$ their distance is defined by $t'_e - t_s$. As the complete input instance is known in advance and the energy function is convex, we assume without loss of generality that all schedules considered in this work do not change the transmission speed during a broadcast. For any page p let w'_p denote the windows length w_p rounded down to the next power of two.

As mentioned above a feasible schedule has to meet the window length of *every* page. Since time windows can start at any point of time—particularly right after the beginning of a broadcast for a page—it is rather easy to see that every feasible schedule has to fulfill the following property: For each page p the distance of each pair of consecutive broadcasts for this page is at most w_p. On the other hand, this property is sufficient to yield the feasibility of a schedule.

1.2 Related Work

The problem considered in this paper is closely related to the windows scheduling problem, where the goal is to find the minimal number of slotted channels needed to feasibly schedule pages for a given set of window lengths. One can think of the original windows scheduling as uniformly fixing the transmission speed for all broadcasts in advance. This results in slots of fixed size and the problem is to assign the pages to these slots while respecting their windows lengths. Since the channels are partitioned into slots of fixed length in the problem definition, feasible schedules are assumed to be cyclic and energy consumption is irrelevant

in classical windows scheduling. Bar-Noy and Ladner [5] present lower bounds on the number of channels and give an algorithm that computes the optimal schedule for harmonic windows scheduling. In a subsequent work Bar-Noy et al. [6] generalize this problem by considering different page sizes. They present an 8-approximation algorithm and a greedy algorithm based upon a tree representation of schedules. The authors also show that this generalized window scheduling problem is NP-hard even if all windows are powers of two.

A related problem that has received a lot of attention also from the energy perspective is job scheduling with deadlines [7]. Here, a sequence of jobs, each with an individual release time, deadline and a certain workload needs to be scheduled on a single speed-controlled processor, such that all jobs are finished in time and the overall energy consumption is minimized. Yao et al. [7] present a polynomial time offline algorithm and propose different online strategies. Bansal et al. [4] present a $(2 \cdot (e\alpha/(\alpha-1))^\alpha)$-competitive online algorithm and show that this is asymptotically best possible. Another work by Briest et al. [2] study pull-based broadcasting on a single speed-controlled channel. They present a conceptionally simple online algorithm that is $((\alpha/(\alpha-1))^2 \cdot 2^\alpha)$-competitive with respect to energy consumption for the case where only one page exists and each request defines a strict deadline. Furthermore, they show that this is best possible and present an online algorithm for the multi-page case. All cited works on energy preservation using speed control mechanisms use the common model that the maximum speed is not upper bounded. Most recently, Chan et al. [8] focused on fixing the maximum speed in job scheduling in advance and presented an algorithm that is 14-competitive in terms of throughput and $(\alpha^\alpha + \alpha^2 4^\alpha)$-competitive on energy usage.

1.3 Contributions

To the author's best knowledge this is the first analysis directly addressed to the minimization of energy consumption for windows scheduling by speed-scaling. To keep assumptions on the model as low as possible, we allow continuous selection of the transmission speed. As a consequence some schedules might not have finite representation, i.e., they are not cyclic. In Section 2 we show that an optimal cyclic schedule differs from an optimal (acyclic) schedule by at most a factor $(1+\varepsilon)$ for an arbitrary constant $\varepsilon > 0$. To this end we first show that the shortest broadcast in any optimal schedule does not shrink over time. Afterwards, we introduce what we call a configuration of a channel, and use a kind of combinatorial argument to show the claim. Section 3 is dedicated to the restricted version of the problem where only one channel is available for transmission. We present a simple algorithm that computes a cyclic schedule with a period of at most w'_n and an approximation ratio of at most $\min\{3 \cdot 4^\alpha, (2 \cdot (1+3/k))^\alpha\}$ for $k = \sum_{p=1}^{n} w'_1/w'_p \geq 1$. At the end of that section we discuss how to implement this algorithm to run in time $\mathcal{O}(n^2)$. Section 4 deals with the more general case of $m \geq 1$ identical channels. Here we obtain an 8^α-approximation by extending the algorithm of the preceding section. We consider this the main contribution of this work. To show this result we first round down

each window length to the next power of two, bound the approximation ratio obtained on this input instance and, finally, transfer this bound on the original window lengths. This extension of the algorithm presented in Section 3 runs in time $\mathcal{O}(m \cdot n^2)$ and obtains a schedule with period $w_n/2$. Although arbitrary speed changes are allowed in our model, the speed used on an arbitrary channel varies at most by a factor of 2 over time.

2 Cyclic and Acyclic Schedules

The definition of energy density allows us to compare the quality of two schedules. This holds for cyclic schedules as well as for acyclic ones. Nevertheless, acyclic schedules are harder to handle; e.g., they might not be stored on finite space. We focus on cyclic schedules and show how to compute such broadcast schedules in the next sections. In this section we give reasons for this choice. We show that for each acyclic schedule one can find a cyclic schedule that has the same energy density up to a factor $(1 + \varepsilon)$ for an arbitrary constant $\varepsilon > 0$. For the remainder of this section let S denote an arbitrary schedule.

Lemma 1. *There exists a feasible schedule whose broadcasts are at least of length $w_1/(64n)$ and whose energy density at most δ_S.*

The proof is omitted due to space limitations. The following lemma will become useful later in this section. Its proof is omitted due to space limitations and can be found in [9].

Lemma 2. *Given two vectors $(a_1, a_2, \ldots, a_n) > 0$ and $(b_1, b_2, \ldots, b_n) > 0$ such that $a_1/b_1 \leq a_2/b_2 \leq \cdots \leq a_n/b_n$. It holds*

$$\frac{a_1}{b_1} \leq \frac{a_1 + a_2}{b_1 + b_2} \leq \cdots \leq \frac{a_1 + a_2 + \cdots + a_n}{b_1 + b_2 + \cdots + b_n}.$$

With this lemma at hand we are able to show that cyclic schedules are not significantly worse than general schedules. The proof of the following theorem is mostly technical and only sketched due to space limitations.

Theorem 1. *For any arbitrary constant $\varepsilon > 0$ there exists a feasible cyclic schedule whose energy density is at most $(1 + \varepsilon) \cdot \delta_S$.*

Proof (Sketch). We fix an optimal schedule S and convert it into a cyclic schedule S'. During this conversion the energy density of schedule S increases by at most a factor $1+\varepsilon$. Choose $1/2 \geq \varepsilon' > 0$ such that $1/(1-\varepsilon')^\alpha \leq 1+\varepsilon$ holds. Without loss of generality we assume the smallest broadcast in schedule S to be bounded from below by $\ell > 0$. We raster the time line equally into slots of length $\sigma := \ell\varepsilon'/2$ and align all broadcast to this raster. This is done by moving each starting point of a broadcast to the next and each finishing point to the preceding raster point. It is not difficult to observe that the schedule's energy density increases at most by a factor $1/(1 - \varepsilon')^{\alpha-1}$ while its feasibility is preserved.

Next we introduce the notation of a *configuration* and say that schedule is in configuration $c = (p, t_1, \ldots, t_n)$ in a particular σ-slot if and only if

1. a broadcast of page p is finished at the end of this slot, and
2. for all $j \in \{1, \ldots, n\}$ the starting point of the last broadcast of page j is exactly t_j slots ago.

Consider a sequence of broadcasts that starts and ends in a slot with some configuration c and meets all constraints on the window lengths. Due to the second set of conditions in the definition of a configuration, it is guaranteed that repeating this sequence repeatedly results in a feasible schedule. On the other hand, at least every $\lceil w_1/\sigma \rceil$ slots a broadcast finishes. Thus, a configuration is reached at least every $\lceil w_1/\sigma \rceil$ slots. Schedule S induces a infinite sequence of configurations. As the number of configurations is finite, at least one configuration has to be reached infinitely often. It is possible to show that there has to exist a configuration c^* and a sequence of broadcast in schedule S which starts and ends in c^* and whose (infinite) repetition yields a schedule S' whose energy consumption is at most a factor $1/(1-\varepsilon')^\alpha$ higher than the one of schedule S.

Let δ_{opt} denote the energy density achieved by an optimal schedule. The last theorem guarantees that for each constant $\varepsilon > 0$ there exists a schedule with finite period whose energy density is at most $(1+\varepsilon) \cdot \delta_{\text{opt}}$. Thus, we focus on them for the remainder of this work. We point out that the period of this cyclic schedule might be large and depends on ε. Hence, the existence of an optimal *cyclic* schedule cannot be guaranteed. We will create schedules with acceptable period and performance in the next sections and start by dealing with the single channel case in the next section.

3 Broadcasts on a Single Channel

In this section we present algorithm SingleApprox and show that it obtains a $\min\{(2\cdot(1+3/k))^\alpha, 3\cdot 4^\alpha\}$-approximation for $k = \sum_{p=1}^{n} w'_1/w'_p$. Before presenting it formally we describe its basic structure. The algorithm rounds each window length down to the nearest power of 2 resulting in window lengths w'_i and partitions the channel equally into slots of length $w'_1(k+1)/(k+3)$. At any point of time each slot has a set of pages assigned to it. The algorithm processes all pages sequentially and assigns each page to the slot that has currently the fewest pages assigned to. Finally, the assignments are transformed into broadcasts by performing them sequentially in order of (non-decreasing) window length. The speed is chosen independently for each slot such that the whole slot is used for broadcasts. The canonical implementation of the algorithm shown below might have a pseudo-polynomial running time. So we discuss an implementation that runs in time $\mathcal{O}(n^2)$ at the end of this section. Before turning to a formal description of algorithm SingleApprox, we state an easy observation. Consider a schedule S that is feasible for a set of window lengths $w'_1, w'_2, \ldots, w'_n$. Modify all window lengths by scaling them by the same factor $c > 0$. Schedule S might be infeasible for this new set of parameters. Nevertheless, converting each broadcast (s, t, p) of schedule S into broadcast $(c \cdot s, c \cdot t, p)$, i.e., scaling it by factor c,

1 Round each page's window length down to the nearest power of 2, i.e., $w'_i := 2^{k_i}$
2 Partition the timeline equally into slots $1, 2, \ldots$ of length $t_0 \leftarrow (k+1)/(k+3)$ where $k = \sum_{p=1}^{n} w'_1/w'_p \in \mathbb{N}$
3 **foreach** *page* p **do**
4 Let s_p denote the currently first emptiest slot.
5 Assign page p to each slot $s_p, s_p + w'_p, s_p + 2 \cdot w'_p, \ldots$
end
6 Choose transmission speed of each slot such that all assigned broadcasts are performed and the whole slot is used.

Algorithm 1. SingleApprox

again results in a feasible schedule. Without loss of generality we assume $w_1 = 1$; otherwise scale all broadcasts by w_1^{-1} before rounding. Observe that the optimal energy density with respect to window lengths w_i and the optimal energy density with respect to w'_i differ at most by a factor $2^{\alpha-1}$. In all proofs we can assume all original window lengths to be powers of two and compensate for this with a factor $2^{\alpha-1}$. We call all slots with fewest pages assigned to it *emptiest* slots. The proofs of this section are omitted due to space limitations.

Lemma 3. *The approximation ratio of algorithm* SingleApprox *is at most* $(2 \cdot (1 + 3/k))^\alpha$ *for* $k = \sum_{p=1}^{n} w'_1/w'_p$.

The proof is based upon the fact that for each page the distance between its broadcasts is close to its window length. Thus, the frequency of broadcasts within schedule S and any optimal schedule is also close to each other. As the broadcast speed in schedule S does not vary too much over time, the claim follows straightforwardly. Obviously, the obtained schedule has a period of w'_n/w'_1 slots. As page p is assigned to every (w'_p/w'_1)-th slot, it is assigned to w'_n/w'_p slots within one period. This sums up to $\sum_{p=1}^{n} w'_n/w'_p$ which results in an average number of $\sum_{p=1}^{n} w'_1/w'_p$ pages per slot. As the number of assigned pages per slot differs at most by one, this yields that parameter k describes the number of pages assigned to the emptiest slot at the end of the algorithm. We point out that the more broadcasts per slot are performed, i.e., the more energy is used per time unit, the better the bound on the approximation ratio gets. However, last lemma only yields an upper bound of 8^α on the approximation ratio due to small k. We improve upon this bound by looking at small $k \in \{1, 2\}$ more carefully. If k is small, the most frequent page 1 dominates the energy consumption. This can be exploited to obtain a better lower bound and yields the next lemma.

Lemma 4. *The approximation ratio of* SingleApprox *is at most* $3 \cdot 4^\alpha$.

A canonical implementation of algorithm SingleApprox has a running time of $\Omega(w_n)$ as the number of slots is bounded below by this term and the emptiest one has to be found in Line 5. On the other hand, the running time can easily be bounded above by $\mathcal{O}(n \cdot w_n)$ which is pseudo-polynomial. Note that it is

impossible for any implementation of algorithm SingleApprox to output S *explicitly slot by slot* in polynomial time since its period is $\Omega(w_n)$. Nevertheless, it is sufficient to know t_0 and all page offsets s_p in order to specify schedule S completely. Moreover, with this information at hand one can construct schedule S on-the-fly over time with low computational effort. Due to this we say that schedule S is computed as soon as these parameters are known. Next lemma states that they can be computed in time $\mathcal{O}(n^2)$.

Lemma 5. *Algorithm* SingleApprox *can be implemented to run in time* $\mathcal{O}(n^2)$.

The proof of Lemma 5 is omitted. The key observation is the regular structure of the assignment: After page p has been processed the assignment is periodic with period $w'_p/2$. With this observation at hand it is not hard to observe that the first emptiest slot in Line 4 can be computed by dynamic programming in polynomial time. We obtain the following theorem.

Theorem 2. *Algorithm* SingleApprox *runs in time* $\mathcal{O}(n^2)$ *and has an approximation ratio of at most* $\min\{3 \cdot 4^\alpha, (2 \cdot (1 + 3/k))^\alpha\}$ *for* $k = \sum_{p=1}^{n} w'_1/w'_p$.

4 Broadcasts on Multiple Channels

In the following we extend the scenario to multiple but identical channels. Each page will be broadcasted on exactly one channel. Essentially, we have to find an assignment of the pages to channels. To this end we extend algorithm SingleApprox to algorithm MultiApprox. Again, first all window lengths are rounded down to the nearest power of 2. Recall that the window lengths are assumed to be ordered non-decreasingly. The first m pages are distributed among the m channels by assigning page i to channel i: Channel i is partitioned into slots of length $w'_i/2$ and page i is assigned to each one of them. Denote the j-th slot on channel i by s^i_j and let a^i_j be the number of pages currently assigned to it. As the slots on different channels might differ, the number of assigned pages is no longer an appropriate measure. Instead we use the slot density. Define the density of slot s^i_j by $d^i_j := 2a^i_j/w'_i$. The remaining pages $m+1, m+2, \ldots, n$ are processed sequentially with decreasing window length. Each page is assigned to the channel currently holding the slot with lowest density. This assignment is done in the same manner as in algorithm SingleApprox. Algorithm 2 shows algorithm MultiApprox in more detail. Here is our main result.

Theorem 3. *Algorithm* MultiApprox *can be implemented to run in time* $\mathcal{O}(n^2 \cdot m)$ *and has an approximation ratio of at most* $8^\alpha + 2^{\alpha-1}$.

The rest of this section is dedicated to proving this theorem. We show the claim on the running time first. Lines 1 and 2 can obviously be performed in linear time. Each iteration of the loop in Lines 3–6 requires to identify the slot with the lowest density. If we computed this slot in a trivial manner, the running time of each iteration would be bounded below by the number of slots. This is at least w'_n which is pseudo-polynomial. Nevertheless, the regular structure

```
1 Round each page's window length down to the nearest power of 2, i.e., w'_i := 2^{k_i}.
2 Partition channel i equally into slots s^i_j of length w'_i/2 and assign page i to each
  one.
3 foreach page p > m do
4 |   Let i be the channel containing slot s^i_j with the lowest density.
5 |   Assign page p to slots s^i_j, s^i_j + w'_p/w'_i, s^i_j + 2w'_p/w'_i, ...
6 |   Recompute densities d^i_j
  end
7 Choose transmission speed of each slot such that all assigned broadcasts are
  performed and the whole slot is used.
```

Algorithm 2. MultiApprox

of the assignment can be exploited to keep the running time polynomial. First, we observe that it is sufficient to know the slot with the lowest density on each channel; afterwards, we take their minimum over all channels. At any point of time the slot with lowest density on channel i is the slot on this channel with the least pages assigned to. Consider subset $\mathcal{P}_i$ of pages that are assigned to channel i. Note that the page assignment on this channel is identical to the one obtained by algorithm SingleApprox if that algorithm is applied to pageset $\mathcal{P}_i$. We use one instance of that algorithm for each channel to keep track of the lowest density of the channels. This yields the claim on the running time.

Let S denote the schedule obtained by algorithm MultiApprox. Before dealing with its approximation ratio we discuss its feasibility. Focus on page p and assume it is assigned to channel i. Since the page assignment on this channel is identical to the one obtained by algorithm SingleApprox on pageset $\mathcal{P}_i$, page p is assigned to one of the first w'_p/w'_i slots. Moreover, beginning with this slot it is assigned to each (w'_p/w'_i)-th slot and the distance between two consecutive broadcasts of page p is bounded from above by $(1 + w'_p/w'_i) \cdot w'_i/2 \leq w'_p$. As this holds for any page p the schedule's feasibility follows.

Let us turn to its approximation ratio. We assume for the rest of this proof that each window length w_i is already a power of 2, no rounding was done in Line 1 and show an approximation ratio of $4^{\alpha-1} + 1$ for this case. An additional factor $2^{\alpha-1}$ compensates for this assumption and establishes the claim. For an arbitrary constant $\varepsilon > 0$ fix a feasible cyclic schedule S^* whose energy density is at most $(1 + \varepsilon)\delta_{\text{opt}}$. Let T be the least common multiple of the periods of schedules S and S^*.

Lemma 6. *The energy consumption induced by page p in schedule S^* within time $[0, T]$ is bounded below by $T \cdot (2/w_p)^\alpha$.*

The proof is completely straightforward and omitted. If all pages were assigned due to Line 2, the number of pages is bounded from above by the number of channels. In this case schedule S broadcasts page i on channel i by broadcasts of length $w_i/2$. The schedule consumes $(2T/w_i) \sum_{i=1}^{n} (2/w_i)^{\alpha-1}$ energy up to time T. Due to Lemma 6 this energy consumption is best possible and the claim on

the approximation ratio of algorithm MultiApprox follows for this case. For the rest of this proof we assume that the number of pages exceed the number of available channels. Consider the moment page $p > m$ is processed and is about to be assigned to channel i. The current assignment on that channel is periodic with period $(w_p/w_i) \cdot (w_i/2) = w_p/2$, i.e., the assignment of any slot s is identical to the assignment of slot $s + w_p/w_i$. This follows from the fact that the assignment on that channel is done in the same manner as in algorithm SingleApprox. Consider the assignment of page n. All slots that are modified by this assignment have density d before and $d' \leq 2d$ after the modification. With respect to this threshold d we define two groups of channels. If *all* slot densities on a channel are in interval $[d, 2d]$, we add this channel to group $\mathcal{C}_=$. If at least one slot density exceeds $2d$ it is put in group $\mathcal{C}_>$.

Lemma 7. *The slot density of any slot of schedule S is at least d. The slot densities on all channels that broadcast at least two distinct pages are in $[d, 2d]$.*

Proof. Have a look at the following game. Given are a number of bins. At the beginning of the game each bin contains a particular load—these loads are not necessarily identical. In each round of the game the bin currently containing the least load is selected and its load increases but is at most doubled. At any point of time pick two bins that were modified during the game. It is not difficult to observe that the load of these two bins differs at most by factor 2. We transfer this observation to schedule S. The slots correspond to the bins. A bin is initiated with some load due to Line 2. Each time a page is assigned to some slot due to Line 5 this slot was the one with lowest density and its density is at most doubled; the corresponding bin was picked and its load at most doubled. Consequently, all densities of slots that were chosen at least once due to Line 5 differ at most by a factor 2. These are all slots that have at least two pages assigned to it. As this holds at any time this holds right *before* page n is processed. Recall that the assignment of page n increases the density of some slots from d to d'. Since the slot with lowest density among all slots (on all channels) is chosen in Line 4, all slots have at least density d. This shows the first part of the claim. As $d' \leq 2d$ holds the densities of all slots on channels $\mathcal{C}_=$ differ at most by factor 2.

Due to this lemma we observe that each channel in $\mathcal{C}_>$ broadcasts at most one page. All slot densities on each of these channels are identical and, therefore, each channel is either in $\mathcal{C}_=$ or $\mathcal{C}_>$. We conclude that these two groups form a partition of the channels. Since each page is broadcasted on exactly one channel they induce a partition into sets $\mathcal{P}_=$ and $\mathcal{P}_>$ on the pages as well.

First, we focus on channels $\mathcal{C}_>$. The slot length on a channel of $\mathcal{C}_>$ depends linearly on window length of the page due to Line 2. For each $p \in \mathcal{P}_>$ we successively half its window length and update the corresponding slot length in schedule S until its density falls in $[d, 2d]$. Note that we do not run algorithm MultiApprox for the new set of window lengths, but merely adjust schedule S. The resulting schedule is denoted by S' and the corresponding set $\mathcal{C}_>$ is empty. Let δ'_{opt} denote the optimal energy density achievable for this modified set of

window lengths and fix an arbitrary cyclic schedule $\widehat{S}$ whose energy density is at most $(1+\varepsilon)\delta'_{\text{opt}}$.

Let T' be the least common multiple of the periods of schedules S, S', S^* and $\widehat{S}$. Since cT' for an arbitrary natural number c is a multiple of the periods of these four schedules, equality $E_S(cT')/E_{S^*}(cT') = \delta_S/\delta_{S^*}$ holds as well as $E_{S'}(cT')/E_{\widehat{S}}(cT') = \delta_{S'}/\delta_{\widehat{S}}$. The following lemma shows that these two ratios are closely related.

Lemma 8. $E_S(cT')/E_{S^*}(cT') \leq E_{S'}(cT')/E_{\widehat{S}}(cT') + 1$.

Proof. As all arguments about energy consumptions used in this proof refer to energy consumptions up to time cT', we omit the term (cT') for clarity. Let $E_S^{>}$ and $E_S^{=}$ denote the energy consumption on channels $\mathcal{C}_>$ and $\mathcal{C}_=$, respectively, in schedule S. The corresponding values for schedule S' are identified by $E_{S'}^{>}$ and $E_{S'}^{=}$. Observe that due to Lemma 6 pages $\mathcal{P}_>$ induce in any cyclic schedule at least the cost the incur in schedule S and S', respectively. Hence, $E_{S^*} \geq E_S^{>}$ and $E_{\widehat{S}} \geq E_{S'}^{>}$ hold. Furthermore, the energy spent on channels $\mathcal{C}_=$ remains unchanged during the modification, i.e., $E_S^{=} = E_{S'}^{=}$. We abbreviate the ratio $E_{S'}/E_{\widehat{S}} = (E_{S'}^{>} + E_{S'}^{=})/E_{\widehat{S}}$ by $\varrho > 1$. The claim follows if we show that $E_S/E_{S^*} = (E_S^{>} + E_S^{=})/E_{S^*}$ is bounded from above by $\varrho + 1$. To this end we do a case inspection on $E_S^{>}$. If $E_S^{>} \geq E_S^{=}/\varrho$ holds, ratio E_S/E_{S^*} is bounded above by $(E_S^{>} + E_S^{=})/E_{S^*} \leq (1+\varrho)E_S^{>}/E_{S^*} \leq 1 + \varrho$. In the remaining case we bound the energy consumption implied by schedule S by $(1 + 1/\varrho) \cdot E_S^{=} \leq (1 + 1/\varrho) \cdot E_{S'}^{=}$. The energy density of an optimal schedule is anti-monotonic in the page window lengths, i.e., increasing some window length does not increase the optimal energy density. $E_{\widehat{S}} \leq E_S$ is fulfilled and the claim holds for this case as well.

According to this lemma in order to show the claimed approximation ratio it is sufficient to bound $E_{S'}(cT')/E_{\widehat{S}}(cT')$ from above for some appropriate c. Let N denote the number of broadcasts done in schedule S' up to time T'. As the distance between two broadcasts of page p is at least $w_p/2$ in schedule S' and this distance is bounded above by w_p in *any* schedule, it is not difficult to verify that at least $cN/2 - n$ broadcasts are performed in schedule $\widehat{S}$ up to that time. For sufficiently large c this is at least $cN(1-\varepsilon)/2$. As cmT time is available to perform these broadcasts energy consumption of $\widehat{S}$ is bounded below by $(1-\varepsilon)N/2 \cdot (2mT'/((1-\varepsilon)N))^{-\alpha+1}$. Recall that the density of any two slots differs by at most a factor 2 in schedule S'. Due to the definition of slot density we conclude that the length of any broadcasts in schedule S' differs at most this factor. Note that on each channel cT' time units are used for broadcasts in schedule S'. Hence, cmT' time units are used to perform cN broadcasts, yielding that the length of the longest one is at least mT'/N time units. As the length of the longest and the shortest broadcasts differs by at most a factor 2, the shortest broadcast is bounded from below by $mT'/(2N)$. This shows that the energy consumption of schedule S' is at most $N \cdot (mT'/(2N))^{-\alpha+1}$. Since the lower bound on the energy consumption of schedule $\widehat{S}$ holds for any $\varepsilon > 0$ and schedule S is not influenced by factor c, we conclude that $E_{S'}(cT')/E_{\widehat{S}}(cT') \leq 2 \cdot 4^{\alpha-1}$ is fulfilled. The inequality $E_S(cT')/E_{S^*}(cT') \leq 2 \cdot 4^{\alpha-1} + 1$ follows from Lemma 8.

5 Conclusions and Open Problems

We investigated the problem of finding energy efficient broadcast schedules for speed-controlled channels. Each page has an individual window length and a feasible schedule has to fulfill the constraint that for each page the distance between two broadcasts for that page has to be bounded above by this length. We introduced the energy density to measure the quality of a broadcast schedule and saw that for each $\varepsilon > 0$ cyclic schedules can approximate general schedules up to factor $1 + \varepsilon$. The length of their period might increase with decreasing ε.

For the single channel version of the problem we presented an easy to implement algorithm and showed that it obtains a $\min\{(2 \cdot (1 + 3/k))^\alpha, 3 \cdot 4^\alpha\}$-approximation for $k = \sum_{p=1}^{n} w'_1/w'_p$. We extended this algorithm to the multi-channel version and proved an approximation ratio of 8^α. Both algorithms run in low-order polynomial time and obtained schedules whose period is bounded above by w_n. This period is reasonable short for practical applications. Furthermore, the speed on each channel varies at most by a factor of 2 over time.

Further research on this topic seems to be promising. The most interesting question might be whether the problem is NP-hard at all. Up to now it remains open if algorithm MultiApprox can be adapted to work on non-identical channels: each channel has an individual power coefficient c and sending at speed s for $1/s$ time units consumes $(cs)^\alpha/s$ energy units. The power coefficients can be used to model technological differences between the channels.

References

1. Shen, W., Norrie, D.H.: Agent-based systems for intelligent manufacturing: A state-of-the-art survey. Knowledge and Information Systems 1(2), 129–156 (1999)
2. Briest, P., Gunia, C.: Energy-efficient broadcast scheduling for speed-controlled transmission channels. In: Proc. of Symp. on Algorithms and Computation (2006)
3. Krco, S.: Implementation solutions and issues in building a personal sensor network for health care monitoring. In: Proc. of the Special Topic Conference on Information Technology Applications in Biomedicine (2003)
4. Bansal, N., Kimbrel, T., Pruhs, K.: Dynamic speed scaling to manage energy and temperature. In: Proc. of the Symp. on Foundations of Computer Science (2004)
5. Bar-Noy, A., Ladner, R.E.: Windows scheduling problems for broadcast systems. SICOMP: SIAM Journal on Computing 32, 1091–1113 (2003)
6. Bar-Noy, A., Ladner, R.E., Tamir, T., VanDeGrift, T.: Windows scheduling of arbitrary length jobs on parallel machines. In: Proc. of Symp. on Parallelism in Algorithms and Architectures (2005)
7. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced cpu energy. In: Proc. of the Symp. on Foundations of Computer Science (1995)
8. Chan, H., Chan, W., Lam, T., Lee, L., Mak, K., Wong, P.: Energy efficient online deadline scheduling. In: Proc. of the Symp. on Discrete Algorithms (2007)
9. Panagiotou, K., Souza, A.: On adequate performance measures for paging. In: Proc. of the Symp. on Theory of Computing (2006)

A New Model to Solve the Swap Matching Problem and Efficient Algorithms for Short Patterns

Costas S. Iliopoulos* and M. Sohel Rahman**,***

Algorithm Design Group
Department of Computer Science, King's College London,
Strand, London WC2R 2LS, England
{csi,sohel}@dcs.kcl.ac.uk
http://www.dcs.kcl.ac.uk/adg

Abstract. In this paper, we revisit the much studied problem of Pattern matching with Swaps (Swap Matching problem, for short). We first present a new graph-theoretic approach to model the problem, which opens a new and so far unexplored avenue to solve the problem. Then, using the model, we devise an efficient algorithm to solve the swap matching problem. The resulting algorithm is an adaptation of the classic shift-or algorithm. For patterns having length similar to the word-size of the target machine, the algorithm runs in $O(n \log m)$ time, where n and m are the length of the text and the pattern respectively.

1 Introduction

The classical pattern matching problem is to find all the occurrences of a given pattern $\mathcal{P}$ of length m in a text $\mathcal{T}$ of length n, both being sequences of characters drawn from a finite character set Σ. This problem is interesting as a fundamental computer science problem and is a basic need of many practical applications such as text retrieval, music retrieval, computational biology, data mining, network security, among many others. In this paper, we revisit the Pattern Matching with Swaps problem (the Swap Matching problem, for short), which is a well-studied variant of the classic pattern matching problem. The pattern P is said to match the text T at a given location i, if adjacent pattern characters can be swapped, if necessary, so as to make the pattern identical to the substring of the text ending (or equivalently, starting) at location i. All the swaps are constrained to be disjoint, i.e., each character is involved in at most one swap. Amir et al. [1] obtained the first non-trivial results for this problem. They showed how to solve the problem in time $O(nm^{1/3} \log m \log \sigma)$, where $\sigma = \min(|\Sigma|, m)$. Amir et al. [3] also studied certain special cases for which $O(n \log^2 m)$ time can be obtained. However, these cases are rather restrictive. Finally, Amir et al. [2] solved the Swap Matching problem

* Supported by EPSRC and Royal Society grants.
** Supported by the Commonwealth Scholarship Commission in the UK under the Commonwealth Scholarship and Fellowship Plan (CSFP).
*** On Leave from Department of CSE, BUET, Dhaka-1000, Bangladesh.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 316–327, 2008.

in time $O(n \log m \log \sigma)$. We remark that all the above solutions to swap matching depend on the fast fourier transform (FFT) technique. It may be noted here that approximate swapped matching [4] and swap matching in weighted sequences [7] have also been studied in the literature.

The contribution of this paper is as follows. We first present a new graph-theoretic approach to model the problem which opens a new and so far unexplored avenue to solve the problem. Then, using the model, we devise an efficient algorithm to solve the swap matching problem. The resulting algorithm is an adaptation of the classic shift-or algorithm and runs in $O(n \log m)$ if the pattern is similar in size to the size of word in the target machine. This seems to be the first attempt to provide an efficient solution to the swap matching problem without using the FFT techniques.

The rest of the paper is organized as follows. In Section 2, we present some preliminary definitions. Section 3 presents our new model to solve the swap matching problem. In Section 4, we present the algorithm to solve the swap matching problem. Finally, we briefly conclude in Section 5.

2 Preliminaries

A *string* is a sequence of zero or more symbols from an alphabet Σ. A string X of length n is denoted by $X[1..n] = X_1X_2 \dots X_n$, where $X_i \in \Sigma$ for $1 \leq i \leq n$. The *length* of X is denoted by $|X| = n$. A string w is called a *factor* of X if $X = uwv$ for $u, v \in \Sigma^*$; in this case, the string w occurs at position $|u| + 1$ in X. The factor w is denoted by $X[|u| + 1..|u| + |w|]$. A *k-factor* is a factor of length k. A *prefix (or suffix)* of X is a factor $X[x..y]$ such that $x = 1$ ($y = n$), $1 \leq y \leq n$ ($1 \leq x \leq n$). We define i-th prefix to be the prefix ending at position i i.e. $X[1..i], 1 \leq i \leq n$. On the other hand, i-th suffix is the suffix starting at position i i.e. $X[i..n], 1 \leq i \leq n$.

Definition 1. *A* swap permutation *for X is a permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ such that:*

1. *if $\pi(i) = j$ then $\pi(j) = i$ (characters are swapped).*
2. *for all $i, \pi(i) \in \{i - 1, i, i + 1\}$ (only adjacent characters are swapped).*
3. *if $\pi(i) \neq i$ then $X_{\pi(i)} \neq X_i$ (identical characters are not swapped).*

For a given string X and a swap permutation π for X, we use $\pi(X)$ to denote the *swapped version* of X, where $\pi(X) = X_{\pi(1)}X_{\pi(2)} \dots X_{\pi(n)}$.

Definition 2. *Given a text $T = T_1T_2 \dots T_n$ and a pattern $P = P_1P_2 \dots P_m$, P is said to* swap match *at location i of T if there exists a swapped version P' of P that matches T at location*[1] *i, i.e. $P'_j = T_{i-m+j}$ for $j \in [1..m]$.*

Problem "SM" (Pattern Matching with Swaps). *Given a text $T = T_1T_2 \dots T_n$ and a pattern $P = P_1P_2 \dots P_m$, we want to find each location $i \in [1..n]$ such that P swap matches with T at location i.*

[1] Note that, we are using the end position of the match to identify it.

Definition 3. *A string X is said to be degenerate, if it is built over the potential $2^{|\Sigma|}-1$ non-empty sets of letters belonging to Σ.*

Example 1. Suppose we are considering DNA alphabet i.e. $\Sigma = \Sigma_{DNA} = \{A, C, T, G\}$. Then we have 15 non-empty sets of letters belonging to Σ_{DNA}. In what follows, the set containing A and T will be denoted by $[AT]$ and the singleton $[C]$ will be simply denoted by C for ease of reading.

Definition 4. *Given two degenerate strings X and Y each of length n, we say $X[i]$* matches *$Y[j], 1 \leq i, j \leq n$ if, and only if, $X[i] \cap Y[j] \neq \emptyset$.*

Example 2. Suppose we have degenerate strings $X = AC[CTG]TG[AC]C$ and $Y = TC[AT][AT]TTC$. Here $X[3]$ matches $Y[3]$ because $X[3] = [CTG] \cap Y[3] = [AT] = T \neq \emptyset$.

3 A Graph-Theoretic Model for Swap Matching

In this section, we present a new model to solve the swap matching problem. In our model, we view the text and the pattern as two separate graphs. We start with the following definitions.

Definition 5. *Given a text $T = T_1 \dots T_n$ of Problem SM, a $\mathcal{T}$-graph, denoted by $T^G = (V^T, E^T)$, is a directed graph with n vertices and $n-1$ edges such that $V^T = \{1, 2, \dots n\}$ and $E^T = \{(i, i+1) | 1 \leq i < n\}$. For each $i \in V^T$ we define $label(i) = T_i$ and for each edge $e \equiv (i, j) \in E_T$ we define $label(e) \equiv label((i, j)) \equiv (label(i), label(j)) = (T_i, T_j)$.*

Note that the labels in the above definition may not be unique. Also, we normally use the labels of the vertices and the edges to refer to them.

Fig. 1. The corresponding $\mathcal{T}$-graph of Example 3

Example 3. Suppose, $T = acacbaccbacacba$. Then the corresponding T-graph is shown in Figure 1.

Definition 6. *Given a text $P = P_1 \dots P_m$ of Problem SM, a $\mathcal{P}$-graph, denoted by $P^G = (V^P, E^P)$, is a directed graph with $3m-2$ vertices and at most $5m-9$ edges. The vertex set V^P can be partitioned into three disjoint vertex sets namely $V^P_{(+1)}, V^P_0, V^P_{(-1)}$ such that $|V^P_{(+1)}| = |V^P_{(-1)}| = m-1$ and $|V^P_{(0)}| = m$. The partition is defined in a $3 \times m$ matrix $M[3, m]$ as follows. For the sake of notational symmetry we use $M[-1], M[0]$ and $M[+1]$ to denote respectively the rows $M[1], M[2]$ and $M[3]$ of the matrix M.*

1. $V^P_{(-1)} = \{M[-1, 2], M[-1, 3], \dots M[-1, m]\}$
2. $V^P_{(0)} = \{M[0, 1], M[0, 2], \dots M[0, m]\}$
3. $V^P_{(+1)} = \{M[+1, 1], M[+1, 2], \dots M[+1, m-1]\}$

The labels of the vertices are derived from P as follows:

1. *For each vertex $M[-1,i] \in V^P_{(-1)}, 1 < i \leq m$:*

$$label(M[-1,i]) = \begin{cases} P_{i-1} & \text{if } P_{i-1} \neq P_i, \\ \mathcal{X} & \text{if } P_{i-1} = P_i, \text{ where } \mathcal{X} \notin \Sigma \end{cases} \quad (1)$$

2. *For each vertex $M[0,i] \in V^P_{(0)}, 1 \leq i \leq m, label(M[0,i]) = P_i$*
3. *For each vertex $M[+1,i] \in V^P_{(+1)}, 1 \leq i < m$:*

$$label(M[+1,i]) = \begin{cases} P_{i+1} & \text{if } P_i \neq P_{i+1}, \\ \mathcal{X} & \text{if } P_i = P_{i+1}, \text{ where } \mathcal{X} \notin \Sigma \end{cases} \quad (2)$$

The edge set E^P is defined as the union of the sets $E^P_{(-1)}, E^P_{(0)}$ and $E^P_{(+1)}$ as follows:

1. $E^P_{(-1)} = \{(M[-1,i], M[0,i+1]), (M[-1,i], M[+1,i+1]) \mid 2 \leq i \leq m-2 \bigwedge label(M[-1,i]) \neq \mathcal{X}\} \bigcup \{(M[-1,m-1], M[0,m]) \mid label(M[-1,m-1]) \neq \mathcal{X}\}$
2. $E^P_{(0)} = \{(M[0,i], M[0,i+1]) \mid 1 \leq i \leq m-1\} \bigcup \{((M[0,i], M[+1,i+1]) \mid 1 \leq i \leq m-2 \bigwedge label(M[+1,i+1]) \neq \mathcal{X}\}$
3. $E^P_{(+1)} = \{(M[+1,i], M[-1,i+1]) \mid 1 \leq i \leq m-1 \bigwedge label(M[+1,i]) \neq \mathcal{X}\}$[2]

The labels of the edges are derived from using the labels of the vertices in the obvious way.

Example 4. Suppose, $P = acbab$. Then the corresponding $\mathcal{P}$-graph P^G is shown in Figure 2. On the other hand, the corresponding $\mathcal{P}$-graph $P^{G'}$ for $P' = accab$ is shown in Figure 3. Note that in P' we have $P'_2 = P'_3 = c$. The dotted edges in Figure 3 are non-existent in $P^{G'}$ and are shown only for the sake of understanding.

Definition 7. *Given a $\mathcal{P}$-graph P^G, a path $Q = u_1 \quad u_\ell = u_1 u_2 \ldots u_\ell$ is a sequence of consecutive directed edges $\langle (u_1,u_2),(u_2,u_3),\ldots(u_{\ell-1},u_\ell)\rangle$ in P^G starting at node u_1 and ending at node u_ℓ. The length of the path Q, denoted by $len(Q)$, is the number of edges on the path and hence is $\ell - 1$ in this case. It is easy to note that the length of a longest path in P^G is $m-1$.*

Definition 8. *Given a $\mathcal{P}$-graph P^G and a $\mathcal{T}$-graph T^G, we say that P^G matches T^G at position $i \in [1..n]$ if and only if there exists a path $Q = u_1 u_2 \ldots u_m$ in P^G having $u_1 \in \{M[0,1], M[+1,1]\}$ and $u_m \in \{M[-1,m], M[0,m]\}$ such that for $j \in [1..m]$ we have $label(u_j) = T_{i-m+j}$*

The above definitions set up our model to solve the swap matching problem. The following Lemma presents the idea for the solution.

[2] Note that, if $label(M[+1,i]) = \mathcal{X}$ then $label(M[-1,i+1]) = \mathcal{X}$ as well.

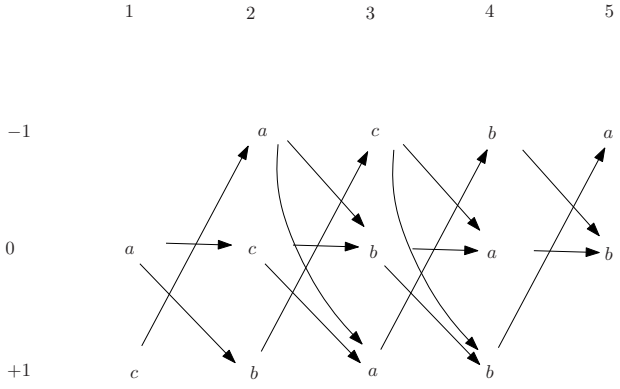


Fig. 2. $\mathcal{P}$-graph of the Pattern $P = acbab$

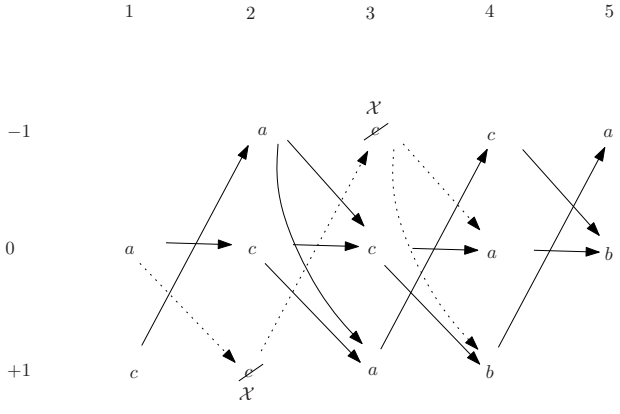


Fig. 3. $\mathcal{P}$-graph of the Pattern $P' = accab$

Lemma 1. *Given a pattern P of length m and a text T of length n, suppose P^G and T^G are the $\mathcal{P}$-graph and $\mathcal{T}$-graph of P and T, respectively. Then, P swap matches T at location $i \in [1..n]$ of T if and only if P^G matches T^G at position $i \in [1..n]$ of T^G.*

Proof. The proof basically follows easily from the definition of the $\mathcal{P}$-graph. At each column of the matrix M, we have all the characters as nodes considering the possible swaps as explained below. Each node in row (-1) and $(+1)$ represents a swapped situation. Now consider column i of M corresponding to P^G. According to definition, we have $M[-1, i] = P_{i-1}$ and $M[+1, i-1] = P_i$. These two nodes represents the swap of P_i and P_{i-1}. Now, if this swap takes place, then in the resulting pattern, P_{i-1} must be followed by P_i. To ensure that, in P^G, the only edge starting at $M[+1, i-1]$, goes to $M[-1, i]$. On the other hand, from $M[-1, i]$ we can either go to $M[0, i+1]$ or to $M[+1, i+1]$: the former is when there is no swap for the next pair and the later is when there is another swap for the next

pair. Recall that, according to the definition, the swaps are disjoint. Finally, the nodes in row 0 represents the normal (non-swapped) situation. As a result, from each $M[0,i]$ we have an edge to $M[0,i+1]$ and an edge to $M[+1,i+1]$: the former is when there is no swap for the next pair as well and the later is when there is a swap for the next pair. So it is easy to see that all the paths of length $m-1$ in P^G represents all combinations considering all possible swaps in P. Hence the result follows.

It is clear that the number of possible paths of length $m-1$ in P^G is exponential in m. So spelling all the paths and then perform a pattern matching against, possibly, a index of T is very time consuming unless m is constant. We on the other hand exploit the above model in a different way and apply a modified version of the classic shift-or [5] algorithm to solve the swap matching problem. In the rest of this section, we present a notion of "Forbidden Graph" and in the next section we show how to exploit this notion and modify the shift-or algorithm to solve the swap matching problem.

Definition 9. *Given a $\mathcal{P}$-graph $P^G = (V^P, E^P)$, the* forbidden Graph $\overline{P}^G = (\overline{V}^P, \overline{E}^P)$ *is such that $\overline{V}^P = V^P$ and $\overline{E}^P$ is defined as follows:* $\overline{E}^P = \{(M[i,j], M[i,j+1]) \mid i \in \{-1,0,+1\}, 1 \leq j < m, (label(M[i,j]) \neq \mathcal{X} \bigvee label(M[i,j+1]) \neq \mathcal{X}) \bigwedge (\forall (M[k,j], M[k,j+1]) \in E^P, k \in \{-1,0,+1\}, label((M[k,j], M[k,j+1])) \neq label((M[i,j], M[i,j+1])))\}$.

In other words, the forbidden graph $\overline{P}^G$ contains an edge (u,v) from column j to $j+1$, where $1 \leq j < m$, if, and only if, there exists no edge from j to $j+1$ in $\mathcal{P}$-graph having the same label.

Example 5. Suppose, $P = acbab$. Then the forbidden graph $\overline{P}^G$ corresponding to the $\mathcal{P}$-graph P^G is shown in Figure 4. The edges of P^G are shown in dashed lines

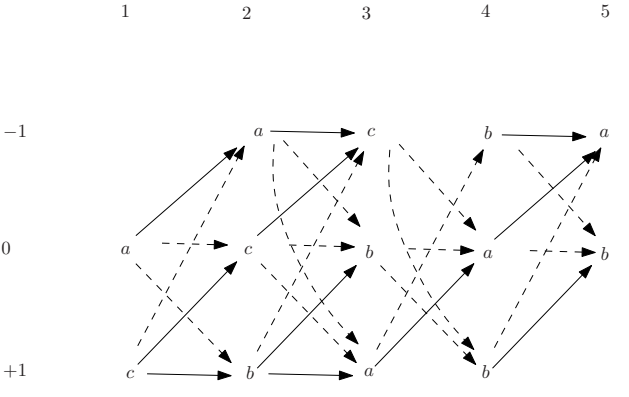


Fig. 4. Forbidden graph (solid edges) corresponding to the $\mathcal{P}$-graph (dashed edges) of the Pattern $P' = acbab$

and the edges of $\overline{P}^G$ are shown in solid lines. Note that, $(M[+1,3], M[+1,4])$ is nonexistent in $\overline{P}^G$, because, $label((M[+1,3], M[+1,4])) = (a,b)$ and we have $(M[+1,3], M[-1,4]) \in E^P$ with the same label (a,b).

4 Algorithm for Swap Matching

In this section, we present a new efficient algorithm based on the model presented in Section 3. Our algorithm is a modified version of the classic shift-or algorithm for pattern matching. For the sake of completeness, we first present a brief account of the shift-or algorithm in the following subsection. In Section 4.2 we present the modifications needed to adapt it to solve the swap matching problem.

4.1 Shift-Or Algorithm

The shift-or algorithm uses the bitwise techniques and is very efficient if the size of the pattern is no greater than the word size of the target processor. The following description of the shift-or algorithm is taken from [6] after slight adaptation to accommodate our notations.

Let R be a bit array of size m. Vector R_j is the value of the array R after text character T_j has been processed. It contains information about all matches of prefixes of P that end at position j in the text. So, for $1 \leq i \leq m$ we have:

$$R_j[i] = \begin{cases} 0 & \text{if } P[1..i] = T[j-i+1..j], \\ 1 & \text{Otherwise.} \end{cases} \tag{3}$$

The vector R_{j+1} can be computed after R_j as follows. For each $R_j[i] = 0$:

$$R_{j+1}[i+1] = \begin{cases} 0 & \text{if } P_{i+1} = T_{j+1}, \\ 1 & \text{Otherwise.} \end{cases} \tag{4}$$

and

$$R_{j+1}[0] = \begin{cases} 0 & \text{if } P_0 = T_{j+1}, \\ 1 & \text{Otherwise.} \end{cases} \tag{5}$$

If $R_{j+1}[m] = 0$ then a complete match can be reported.

The transition from R_j to R_{j+1} can be computed very fast as follows. For each $c \in \Sigma$ let D_c be a bit array of size m such that for $1 \leq i \leq m$, $D_c[i] = 0$ if and only if $P_i = c$.The array D_c denotes the positions of the character c in the pattern P. Each D_c for all $c \in \Sigma$ can be preprocessed before the pattern search. And the computation of R_{j+1} reduces to two operations, shift and or:

$$R_{j+1} = SHIFT(R_j)\ OR\ D_{T_{j+1}}$$

4.2 Modifying Shift-Or Algorithm for Swap Matching

In this section, we modify the shift-or algorithm to solve swap matching problem. To do that we use the graph model, particularly the forbidden graph, presented in Section 3. The idea is quite simple and described as follows. First of all, the shift-or algorithm can be extended easily for the degenerate patterns [5]. In our swap matching model the pattern can be thought of a having a set of letters at each position as follows: $\widetilde{P} = [M[0,1]M[+1,1]]\ [M[-1,2]M[0,2]M[+1,2]] \ldots [M[-1,m-1]M[0,m-1][+1,m-1]]\ [M[-1,m]M[0,m]]$. Note that we have used $\widetilde{P}$ instead of P above because, in our case, the sets of characters in the consecutive positions in the pattern P don't have the same relation as in a usual degenerate pattern. In particular, in our case, a match at position of $i+1$ of P will depend on the previous match of position i as the following example shows.

Example 6. Suppose, $P = acbab$ and $T = bcbaaabcba$. The $\mathcal{P}$-graph of P is shown in Figure 2. So, in line of above discussion, we can say that $\widetilde{P} = [ac][acb][cba][ba][ab]$. Now, as can be easily seen, if we consider degenerate match, then $\widetilde{P}$ matches T at position 2 and 6. However, P swap matches T only at position 6; not at position 2. To elaborate, note that at position 2, the match is due to c. So, according to the graph P^G the next match has to be an a and hence at position 2 we can't have a swap match.

In what follows, we present a novel technique to adapt the shift-or algorithm to tackle the above situation. We use the forbidden graph as follows. For the sake of convenience, in the discussion that follows, we refer to both $\widetilde{P}$ and the pattern P as though they are equivalent; but it will be clear from the context what we really mean. Suppose we have a match up to position $i < m$ of $\widetilde{P}$ in $T[j-i+1..j]$. Now we have to check whether there is a 'match' between T_{j+1} and P_{i+1}. For simple degenerate match, we only need to check whether $T_{j+1} \in P_{i+1}$ or not. However, as the Example 6 shows, for our case we need to do more than that. What we do is as follows. Suppose that $T_j = c = M[\ell, i]$. Now, from the forbidden graph we know which of the $M[k, i+1], k \in [-1, 0, +1]$ can't follow $M[\ell, i]$. So, for example, even if $M[q, i+1] = T[j+1]$ we can't continue if there is an edge from $M[\ell, i]$ to $M[q, i+1]$ in the forbidden graph (or equivalently if there is no edge from $M[\ell, i]$ to $M[q, i+1]$ in the $\mathcal{P}$-graph).

In the rest of this section, we show how we use the forbidden graph to modify the shift-or algorithm to solve the swap matching problem. Recall that, we first process the pattern to compute the masks D_c for every $c \in \Sigma$. This can be done in $O(m/w(m+\Sigma))$ time [5] when pattern is not degenerate. However, in our case, we need to assume that our pattern has a set of letters in each position. In this case, we require $O(m/w(m'+\Sigma))$ time where m' is the sum of the cardinality of the sets at each position [5]. In general degenerate strings, m' can be $m|\Sigma|$ in the worst case. However, in our case, $m' = |V^P| = O(m)$, where V^P is the vertex set of the $\mathcal{P}$-graph. So, computation of the D-mask requires $O(m/w(m+\Sigma))$ time in the worst case. Then we do a further processing on P as follows. We compute the forbidden graph $\overline{P}^G = (\overline{V}^P, \overline{E}^P)$ from the $\mathcal{P}$-graph $P^G = (V^P, E^P)$. Recall that

$V^P = O(m)$ and $E^P = O(m)$ and therefore, by definition, we have $\overline{V}^P = O(m)$ and $\overline{E}^P = O(m)$. So we can compute the forbidden graph in $O(m)$ time.

Two edges $(u, v), (x, y)$ of the forbidden graph (and the $\mathcal{P}$-graph) are said to be 'same' if $label(u) = label(x)$ and $label(v) = label(y)$, i.e. if the two edges have the same labels. Also, given an edge $(u, v) \equiv (M[i_1, j_1], M[i_2, j_2])$ we say that edge (u, v) 'belongs to' column j_2, i.e. where the edge ends; and we say $col((u, v)) \equiv col((M[i_1, j_1], M[i_2, j_2])) = j_2$. Now we traverse all the edges and construct a set of sets $\mathcal{S} = \{S_1 \dots S_\ell\}$ such that each $S_i, 1 \leq i \leq \ell$ contains the edges that are 'same'. The set S_i is named by the (same) label of the edges it contains and we may refer to S_i using its name. Now, we construct forbidden masks $F_{S_i}, 1 \leq i \leq \ell$ such that $F_{S_i}[k] = 1$ if, and only if, there is an edge $(u, v) \in S_i$ having $col((u, v)) = k$. Note that $\ell = O(m)$.

The construction of the forbidden mask can be done in $O(m/w\ m \log m)$ time as follows. We first initialize all the entries of the forbidden masks to 0 which requires $O(m/w\ m)$ time. Then we start traversing the edges. Consider the first edge (u_1, v_1). We know the label of this edge is $label((u_1, v_1)) \equiv (label(u_1), label(v_1))$. We include the label of this edge in a name database and assign a set $S_i \in \mathcal{S}$ to this name and keep pointers for constant time reference later. We also set $F_{S_i}[j] = 1$, where $col((u_1, v_1)) = j$. Now, consider another edge (u_k, v_k). This time we first check whether $label((u_k, v_k))$ already exists in the name database. If yes, then we use the existing name to do the update otherwise we include the label in the name database and continue as before. It is clear that this check in the database can be done in $O(\log \ell) = O(\log m)$. Since we have $O(m)$ edges, the complete construction of the forbidden mask requires $O(m/w\ m \log m)$ time.

With the forbidden masks at our hand, for our problem, we simply need to compute R_{j+1} as follows:

$$R_{j+1} = \ SHIFT(R_j)\ OR\ D_{T_{j+1}}\ OR\ F_{(T_j, T_{j+1})}$$

Note that, to locate the appropriate forbidden mask we again need to perform a look up in the name database constructed during the construction of the forbidden mask. So, in total the construction of the R values require $O(n \log m)$ time. One detail is that, if $F_{(T_j, T_{j+1})}$ doesn't exist then we assume the mask to have all 0's. It is easy to see that this works because the forbidden mask allows R_{j+1} to have 0 at position i if, and only if, the edge (T_j, T_{j+1}) is not 'forbidden'. Example 7 shows a complete execution of our algorithm.

Example 7. Suppose, $P = accab$ and $T = acacbaccbacacba$. The $\mathcal{P}$-graph and corresponding forbidden graph of P is shown in Figure 3 and 5 respectively. The D-masks and F-masks are shown in Figure 6 and 7 respectively. Figure 8 shows the detail computation of the R bit array up to the first match found. Figure 8 shows the complete computed values of R.

The running times of the different phases of the algorithm are listed in Figure 10. Therefore, in total the running time of our algorithm is $O(m/w(m \log m + |\Sigma| + n \log m))$. So, when pattern size is similar to the word size of the target machine,

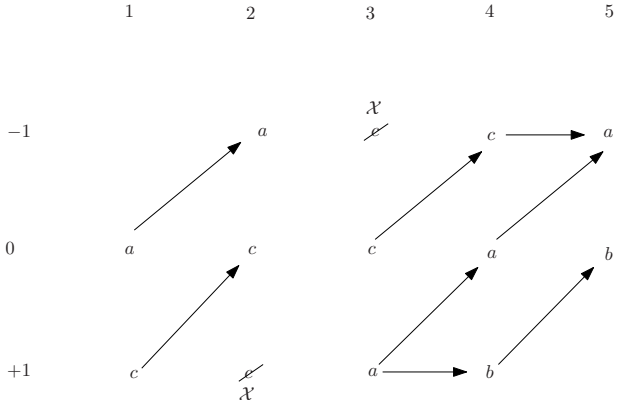


Fig. 5. Forbidden graph for $P = accab$

	D	D_a	D_b	D_c	D_X [a]
1	$[ac]$	0	1	0	1
2	$[acc]$	0	1	0	1
3	$[acc]$	0	1	0	1
4	$[cab]$	0	0	0	1
5	$[ab]$	0	0	1	1

[a] Here X indicates all letters that are not present in P.

Fig. 6. The D-masks for Example 7

	$F_{(a,a)}$	$F_{(a,b)}$	$F_{(b,b)}$	$F_{(c,c)}$	$F_{(c,a)}$	$F_{(X,X)}$ [a]
1	0	0	0	0	0	0
2	1	0	0	1	0	0
3	0	0	0	0	0	0
4	1	1	0	1	0	0
5	1	0	1	0	1	0

[a] Here (X, X) indicates all edges that are not present in the forbidden graph.

Fig. 7. The F-masks for Example 7

we achieve a very good running time of $O(m \log m + |\Sigma| + n \log m) = O(n \log m)$. This follows because we can safely assume that $m \leq n$ and $|\Sigma| \leq n$. Therefore, we have the following theorem.

Theorem 1. *The swap matching problem can be solved in* $O(m/w(m+n) \log m)$ *worst case running time.*

Corollary 1. *The swap matching problem can be solved in* $O(n \log m)$ *worst case running time if the pattern is similar to the word size of the target machine.*

	–	SH	D_a	$F_{(X,X)}$	OR	SH	D_c	$F_{(a,c)}$	OR	SH	D_a	$F_{(c,a)}$	OR	SH	D_c	$F_{(a,c)}$	OR	SH	D_b	$F_{(c,b)}$	OR	...
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	...
2	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	...
3	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	...
4	1	1	0	0	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	...
5	1	1	0	0	1	1	1	0	1	1	0	1	1	1	1	0	1	0	0	0	0	...
																						...

Fig. 8. Detail steps up to the first reported match of Example 7. Here SH means Shift operation on the previous column and OR means or operation on the previous 3 columns.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	a	c	a	c	b	a	c	c	b	a	c	a	c	b	a
1 a	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0
2 c	1	0	0	0	1	1	0	1	1	1	0	0	0	1	1
3 c	1	1	0	0	1	1	1	0	1	1	1	0	0	1	1
4 a	1	1	1	0	0	1	1	1	0	1	1	1	0	0	1
5 b	1	1	1	1	0	0	1	1	1	0	1	1	1	0	0

Fig. 9. The complete computed values of R in Example 7. The occurrences of swap match are shown using tick marks. Note that the end location of the matches are identified here.

Phase	Running Time
Computation of D-masks	$O(m/w(m + \|\Sigma\|))$
Computation of F-masks	$O(m/w\ m \log m)$
Computation of R-values	$O(m/w\ n \log m)$

Fig. 10. Running times of the different phases

5 Conclusion

In this paper, we have revisited the Pattern Matching with Swaps problem, a well-studied variant of the classic pattern matching problem. We have presented a new graph-theoretic approach to model the problem which opens a new and so far unexplored avenue to solve the problem. Then, using the model, we have devised an efficient algorithm to solve the swap matching problem. The resulting algorithm is an adaptation of the classic shift-or algorithm and runs in $O(n \log m)$ if the pattern-length is similar to the word-size in the target machine. Notably, the best known algorithm for swap matching runs in $O(n \log m \log \sigma)$ and uses the FFT technique, which has large hidden constants inside its good theoretical bound. This seems to be the first attempt to provide an efficient solution to the swap matching problem without using FFT techniques. Moreover the techniques used in our algorithm is quite simple and easy to implement. We believe that the new graph theoretic model could be used to devise more efficient algorithms

and a similar approach can be taken to model similar other variants of the classic pattern matching problem. Furthermore, it would be interesting to 'swap' the definitions of $\mathcal{T}$- graph and $\mathcal{P}$- graph and investigate whether efficient pattern matching techniques for Directed acyclic graph can be employed to devise efficient off-line and online algorithms for swap matching.

References

1. Amir, A., Aumann, Y., Landau, G.M., Lewenstein, M., Lewenstein, N.: Pattern matching with swaps. J. Algorithms 37(2), 247–266 (2000)
2. Amir, A., Cole, R., Hariharan, R., Lewenstein, M., Porat, E.: Overlap matching. Inf. Comput. 181(1), 57–74 (2003)
3. Amir, A., Landau, G.M., Lewenstein, M., Lewenstein, N.: Efficient special cases of pattern matching with swaps. Inf. Process. Lett. 68(3), 125–132 (1998)
4. Amir, A., Lewenstein, M., Porat, E.: Approximate swapped matching. Inf. Process. Lett. 83(1), 33–39 (2002)
5. Baeza-Yates, R., Gonnet, G.: A new approach to text searching. Communications of the ACM 35, 74–82 (1992)
6. Charras, C., Lecroq, T.: Handbook of Exact String Matching Algorithms. Texts in Algorithmics. King's College, London (2004)
7. Zhang, H., Guo, Q., Iliopoulos, C.S.: String matching with swaps in a weighted sequence. In: Zhang, J., He, J.-H., Fu, Y. (eds.) CIS 2004. LNCS, vol. 3314, pp. 698–704. Springer, Heidelberg (2004)

Certification of Proving Termination of Term Rewriting by Matrix Interpretations*

Adam Koprowski and Hans Zantema

Eindhoven University of Technology
Department of Computer Science
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
{A.Koprowski,H.Zantema}@tue.nl

Abstract. We develop a Coq formalization of the matrix interpretation method, which is a recently developed, powerful approach to proving termination of term rewriting. Our formalization is a contribution to the CoLoR project and allows to automatically certify matrix interpretation proofs produced by tools for proving termination. Thanks to this development the combination of CoLoR and our tool, TPA, was the winner in 2007 in the new certified category of the annual Termination Competition.

1 Introduction

Termination is an important concept in term rewriting. Many methods for proving termination have been proposed over the years. Recently the emphasis in this area is on automation and a number of tools have been developed for that purpose. One of such tools is TPA [13] developed by the first author.

To evaluate termination tools and stimulate their improvement the annual Termination Competition [3] is organized, where such tools compete on a set of problems from the Termination Problems Database (TPDB), [4]. This competition has become a de-facto standard in evaluation of new termination techniques and developments of termination tools.

However, every year termination tools are becoming more and more complex and are changing rapidly as new techniques are being developed and old ones re-implemented. Therefore ensuring correctness of such tools is a challenging task. This was one of the motivations to start the CoLoR [6] project, initiated by Frédéric Blanqui in 2004. The goal of the project is to use the Coq [1] theorem prover to fully automatically verify results produced by tools for proving termination.

The main subject of this paper is our contribution to the CoLoR project, namely formalization of the matrix interpretation method [9]. This recent method turned out to be very powerful for proving termination and was incorporated into many modern termination provers. Due to space restrictions we present this development for termination only, but it is also applicable to relative termination problems and in the setting of dependency pairs [5]. For a more detailed description we refer to [15].

* Some preliminary results of this paper were first announced in [14].

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 328–339, 2008.

This year in the termination competition the new certified category has been introduced, where tools must not only find a termination proof but also ensure its correctness by stating and proving it in an established theorem prover. Our contribution to CoLoR allowed the combined entry of TPA+CoLoR to win the 2007 edition of the competition in this newly introduced category.

Concerning related work in the first place we should mention the Coccinelle library which uses approach similar to the one employed by CoLoR and also uses Coq theorem prover. We will say more about it in Section 4, where we evaluate the results of CoLoR in the context of the termination competition.

The recent work of Alexander Krauss [16] is another effort toward certified termination. It is different in several aspects. Its main aim is to automatically generate certified termination proofs for recursive functions used in Isabelle/HOL theorem prover. However external termination provers are not involved and the only termination technique supported by this method is the size-change principle.

The rest of this paper is organized as follows. First in Section 2 we recapitulate the theory of matrix interpretations from [9]. Section 3 presents an overview of the Coq formalization of the theoretical results from the preceding section. It is followed by Section 4 where the method is evaluated in the context of the Termination Competition. We conclude in Section 5.

2 Theory of Matrix Interpretations

2.1 Preliminaries

Let Σ be a signature. For a set of variable symbols $\mathcal{V}$, let $\mathcal{T}(\Sigma, \mathcal{V})$ be the set of terms over Σ and $\mathcal{V}$. We denote application of a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ to a term t by $t\sigma$.

A *term rewriting system* (TRS) $\mathcal{R}$ over $\Sigma, \mathcal{V}$ is a set of pairs $(\ell, r) \in \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$, for which $\ell \notin \mathcal{V}$ and all variables in r occur in ℓ. Pairs (ℓ, r) are called *rewrite rules* and are usually written as $\ell \to r$.

For a TRS $\mathcal{R}$ the *top rewrite relation* $\overset{\text{top}}{\to}_{\mathcal{R}}$ on $\mathcal{T}(\Sigma, \mathcal{V})$ is defined by $t \overset{\text{top}}{\to}_{\mathcal{R}} u$ if and only if there is a rewrite rule $\ell \to r \in \mathcal{R}$ and a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ such that $t = \ell\sigma$ and $u = r\sigma$. The *rewrite relation* $\to_{\mathcal{R}}$ is defined to be the smallest relation such that $\overset{\text{top}}{\to}_{\mathcal{R}} \subseteq \to_{\mathcal{R}}$ and if $t_i \to_{\mathcal{R}} u_i$ and $t_j = u_j$ for $j \neq i$, then $f(t_1, \ldots, t_n) \to_{\mathcal{R}} f(u_1, \ldots, u_n)$ for every $f \in \Sigma$ of arity n.

A binary relation $\to$ is called *terminating* or *strongly normalizing*, notation $\mathsf{SN}(\to)$, if it is well-founded. A TRS $\mathcal{R}$ is called terminating if $\mathsf{SN}(\to_{\mathcal{R}})$ holds, shortly written as $\mathsf{SN}(\mathcal{R})$

2.2 Monotone Algebras

Here we summarize the monotone algebra theory as presented in [9,10]. There is one difference: in contrast to [9] we do not consider many-sortedness. It is not essential for certification as every proof in the many-sorted setting can be trivially translated to the one-sorted setting. The reason for this more complex

setup in [9] is that it allows for an optimization in the search for termination proofs using matrix interpretations.

The monotone algebra approach works for all non-empty sets A; when using matrix interpretations the set A always consists of the set of vectors over $\mathbb{N}$ of a fixed dimension.

Definition 1. *An operation* $[f] : A \times \cdots \times A \to A$ *is* monotone *with respect to a binary relation* $\to$ *on* A *if for all* $a_i, b_i \in A$ *for* $i = 1, \ldots, n$ *with* $a_i \to b_i$ *for some* i *and* $a_j = b_j$ *for all* $j \neq i$ *we have* $[f](a_1, \ldots, a_n) \to [f](b_1, \ldots, b_n)$.

An extended weakly monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ *is a* Σ*-algebra* $(A, [\cdot])$ *equipped with two binary relations* $>, \gtrsim$ *on* A *such that* $>$ *is well-founded,* $> \cdot \gtrsim \subseteq >$ *and for every* $f \in \Sigma$ *the operation* $[f]$ *is monotone with respect to* $>$.

Up to presentation details the following theorem is the one-sorted version of the main theorem for the matrix interpretations from [9, Theorem 2].

Theorem 2. *Let* $\mathcal{R}, \mathcal{R}'$ *be TRSs over a signature* Σ. *Let* $(A, [\cdot], >, \gtrsim)$ *be an extended monotone* Σ*-algebra such that* $[\ell, \alpha] \gtrsim [r, \alpha]$ *for every rule* $\ell \to r$ *in* $\mathcal{R}$ *and* $[\ell, \alpha] > [r, \alpha]$ *for every rule* $\ell \to r$ *in* $\mathcal{R}'$, *for every* $\alpha : \mathcal{V} \to A$.

Then $\mathsf{SN}(\to_{\mathcal{R}})$ *implies* $\mathsf{SN}(\to_{\mathcal{R}} \cup \to_{\mathcal{R}'})$.

2.3 Matrix Interpretations

Now we present matrix interpretations with a fixed dimension d as an instance of monotone algebras. For the interpretation $[f]$ of a symbol $f \in \Sigma$ of arity n we choose a vector $\boldsymbol{f} \in \mathbb{N}^d$ and n matrices $F_1, F_2, \ldots, F_n$ over $\mathbb{N}$, each of size $d \times d$, such that the upper left elements $(F_i)_{1,1}$ are positive for all $i = 1, 2, \ldots, n$. Now we define

$$[f](\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}) \;=\; F_1\boldsymbol{v_1} + \cdots + F_n\boldsymbol{v_n} + \boldsymbol{f} \tag{1}$$

for all $\boldsymbol{v_1}, \ldots, \boldsymbol{v_n} \in A$.

So we fix a monotone algebra with $A = \mathbb{N}^d$, interpretations $[\cdot]$ defined as above and we use the following orders on algebra elements:

$$(u_1, \ldots, u_d) \gtrsim (v_1, \ldots, v_d) \iff \forall i : u_i \geq_{\mathbb{N}} v_i$$
$$(u_1, \ldots, u_d) > (v_1, \ldots, v_d) \iff (u_1, \ldots, u_d) \gtrsim (v_1, \ldots, v_d) \wedge u_1 >_{\mathbb{N}} v_1$$

One easily checks that $(A, [\cdot], >, \gtrsim)$ is an extended monotone Σ-algebra.

Let $x_1, \ldots, x_k$ be the variables occurring in ℓ, r. Then due to the linear shape of the functions $[f]$ we can compute matrices $L_1, \ldots, L_k, R_1, \ldots, R_k$ and vectors $\boldsymbol{l}, \boldsymbol{r}$ such that

$$[\ell, \alpha] = L_1\boldsymbol{x_1} + \cdots + L_k\boldsymbol{x_k} + \boldsymbol{l}, \qquad [r, \alpha] = R_1\boldsymbol{x_1} + \cdots + R_k\boldsymbol{x_k} + \boldsymbol{r} \tag{2}$$

where $\alpha(x_i) = \boldsymbol{x_i}$ for $i = 1, \ldots, k$.

For matrices $B, C \in \mathbb{N}^{d \times d}$ write $B \succcurlyeq C$ as a shorthand for $\forall i, j : (B)_{i,j} \geq (C)_{i,j}$. The following lemma provides a decision procedure for orders $>$ and $\gtrsim$ lifted to terms as used in Theorem 2.

Lemma 3. *Let ℓ, r be terms and let matrices $L_1, \ldots, L_k, R_1, \ldots, R_k$ and vectors $\boldsymbol{l}, \boldsymbol{r}$ be defined as above. Then $\forall \alpha : \mathcal{V} \to A, [\ell, \alpha] \gtrsim [r, \alpha]$ (resp. $[\ell, \alpha] > [r, \alpha]$) iff $\forall i : L_i \succcurlyeq R_i$ and $\boldsymbol{l} \gtrsim \boldsymbol{r}$ (resp. $\boldsymbol{l} > \boldsymbol{r}$)*

Now the approach of applying Theorem 2, for proving $\mathsf{SN}(\mathcal{R})$ is as follows:

- Fix a dimension d.
- For every symbol $f \in \Sigma$ choose a vector $\boldsymbol{f} \in \mathbb{N}^d$ and matrices $F_i \in \mathbb{N}^{d \times d}$ for $i = 1, 2, \ldots, n$ for n being the arity of f, such that the upper left elements $(F_i)_{1,1}$ are positive for all $i = 1, 2, \ldots, n$.
- For every rule $\ell \to r \in \mathcal{R}$ check that $L_i \succcurlyeq R_i$ for $i = 1, \ldots, k$ and $\boldsymbol{l} \gtrsim \boldsymbol{r}$ for the corresponding matrices L_i, R_i and vectors $\boldsymbol{l}$, $\boldsymbol{r}$ as defined above.
- Remove all rules from $\mathcal{R}$ moreover satisfying $l_1 > r_1$.
- If the remaining $\mathcal{R}$ is empty we are finished, otherwise the process is repeated for the reduced TRS $\mathcal{R}$.

Example 4. We illustrate the above procedure on an example. Consider the TRS consisting of the following single rule: $\mathrm{a}(\mathrm{a}(x)) \to \mathrm{a}(\mathrm{b}(\mathrm{a}(x)))$. It is worth noting that this TRS is not simply terminating and hence simplification orders are bound to fail for it. For the approach of matrix interpretations we choose dimension $d = 2$ and the following interpretation of symbols:

$$[\mathrm{a}(x)] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad [\mathrm{b}(x)] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We proceed by computing interpretation of the left and right hand side of the single rule.

$$[\mathrm{a}(\mathrm{a}(x))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$[\mathrm{a}(\mathrm{b}(\mathrm{a}(x)))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Evaluating that expressions to linear form, as in Equation 2 yields:

$$[\mathrm{a}(\mathrm{a}(x))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad [\mathrm{a}(\mathrm{b}(\mathrm{a}(x)))] = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

We observe that coefficients standing by x are equal and for the constant terms we have $\begin{bmatrix} 1 \\ 1 \end{bmatrix} > \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ as we have strict decrease in the first position and equality in the second. Hence by Lemma 3 we conclude that $[\mathrm{a}(\mathrm{a}(x)), \alpha] > [\mathrm{a}(\mathrm{b}(\mathrm{a}(x))), \alpha]$ for all $\alpha : \mathcal{V} \to A$. Application of Theorem 2 allows us to remove this rule. As this is the only rule we have proven termination of this one rule TRS.

3 Coq Formalization

Our formalization was developed within the CoLoR project, so we begin by a short introduction of CoLoR in Section 3.1. Then we continue with a description of the

formalization of matrix interpretations, which consists of several parts. The formalization of monotone algebras, introduced in Section 2.2, is presented in Section 3.2. To deal with matrices we had to develop a Coq library of matrices; this is the subject of Section 3.3. Then in Section 3.4 we present the formalization of the matrix interpretations method, corresponding to the theory developed in Section 2.3.

3.1 CoLoR: Certification of Termination

The CoLoR [6] project was founded by Frédéric Blanqui in March 2004, with the goal of certification of termination proofs found by termination provers in Coq. It is available at the following address: `http://color.loria.fr`.

It essentially consists of three parts:

- TPG (Termination Proofs Grammar): a formal grammar for the termination proofs.
- CoLoR (Coq Library on Rewriting and Termination): a library of results on termination of rewriting, formalized in Coq.
- Rainbow: a tool for transforming termination proofs in the TPG format into Coq scripts certifying termination by employing results from CoLoR.

The general approach to certifying termination with CoLoR is presented in Figure 1. For a given TRS $\mathcal{R}$ some termination prover is called. If it succeeds in proving termination, it outputs a termination proof in the TPG format. Such an encoding of a proof is given to Rainbow which translates it into a Coq script containing a formal proof of the claim that $\mathcal{R}$ is terminating by using results from the CoLoR library. Then Coq is executed on such a script to verify that the termination proof found by the termination tool is indeed correct.

For a more detailed description of the TPG format we refer to [15].

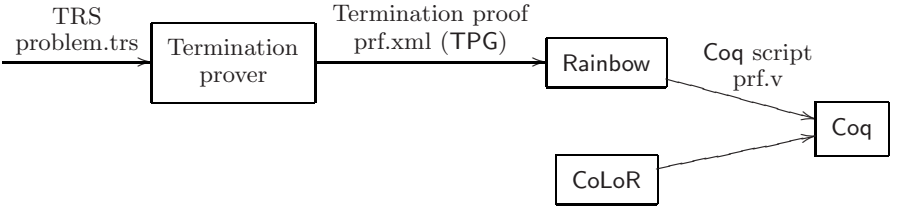


Fig. 1. Certifying termination with CoLoR

3.2 Monotone Algebras

While doing this formalization we faced a number of design choices. The essential question was whether to simply formalize matrix interpretations as they are or to try to make the development as general as possible, such that hopefully (parts of) it could be reused for other techniques and also extensions to the technique itself would be feasible. We opted for the latter. Hence we formalized monotone algebras in their full generality and only later instantiated them to

matrix interpretations; as in the theory presented in Sections 2.2 and 2.3. This, later on, allowed us to easily express the technique of polynomial interpretations in the setting of monotone algebras, making it more powerful and more generally applicable. For more details we again refer to [15].

To achieve such a generic formalization we found the module mechanism of Coq especially useful. It allows for mass abstraction by encapsulating a number of declarations and definitions in modules. Such modules can be parameterized by means of functors, that is functions from modules to modules. For instance we formalized monotone algebras in Coq as a functor, which takes as an argument a structure describing a weakly monotone Σ-algebra instance, as introduced in Section 2.2.

For the formalization there is however one more thing that we need in order to be able to deal with concrete examples. For an application of Theorem 2 we need to check for arbitrary terms ℓ and r whether $[\ell, \alpha] > [r, \alpha]$ for every $\alpha : \mathcal{V} \to A$ and similarly for $\gtrsim$. Our first approach was to require the relations $>$ and $\gtrsim$ lifted to terms to be decidable, that is to require a proof that for two arbitrary elements the relation between them either holds or not. Such decidability results proven in the constructive logic of Coq provide a decision procedure. By making proofs transparent and hence allowing to reduce associated proof terms, one effectively obtains an algorithm for checking whether two given terms can be oriented with the given relation.

This approach however has one limitation: we require a decidability proof, so indeed the relations in question must be decidable. This is the case for matrix interpretations due to the characterization of Lemma 3 but it is not so for instance for non-linear polynomial interpretations. Therefore to make our development more general we actually require two decidable relations $\rhd$ and $\trianglerighteq$ such that $\rhd \subseteq >$ and $\trianglerighteq \subseteq \gtrsim$ and those relations are used in application of Theorem 2 to check whether a rule can be (weakly) oriented. The fact that they are subsets of $>$ and $\gtrsim$ ensures soundness of this approach. But there is no completeness requirement allowing to use some heuristics in cases where the intended relations are not decidable, such as in case of polynomial interpretations; see [15] for more details.

To give a feeling of how theorems from Section 2.2 are stated in the theorem prover we present the Coq equivalent of Theorem 2.

```
Lemma ma_termination:
 let R_gt := partition part_succ R in
 let R_ge := partition part_succeq R in
   monotone I succ -> snd R_ge = nil -> WF (red (snd R_gt)) -> WF (red R).
```

Let us try to explain the components of this statement. To begin with `partition P l` is a function that given a predicate `P` and a list `l`, splits this list into two parts and returns them as a pair `l1, l2`, such that `P` holds for every element of the list `l1` and does not hold for every element of `l2`.

Now `part_succ` and `part_succeq` are predicates for the `partition` function, corresponding to the relations `succ` ($>$) and `succeq` ($\gtrsim$). We demand `succ` to be monotone, `monotone I succ`. Now we require the second component of the pair `R_ge` to be empty, hence all the rules of `R` must be weakly oriented. Finally this theorem states that we can conclude `WF (red R)` if, on top of all the other

requirements that we mentioned, we can prove `WF (red (snd R_gt))` so of the TRS consisting of the rules from `R` that could not be oriented strictly. Stating this problem in such "operational" style allows us to easily apply it for concrete instances of termination problems.

The monotone algebra module also contains Coq tactics allowing to deal with proving termination for concrete examples. This means that for using a monotone algebra approach one only needs to provide a monotone algebra instance and as a result one obtains all the results and a full machinery for proving termination. We will sketch in Section 3.4 how we instantiated monotone algebras to the matrix interpretation method. We also did that for polynomial interpretations; the interested reader is referred to [15].

3.3 Matrices

To begin with, the sole fact that we had to formalize matrices may be surprising — one would expect such a general notion to be readily available in a theorem prover. But it is not present in the Coq standard library. Moreover we could find only one Coq development where matrices were used: the contribution by Nicolas Magaud [17], where he proves ring properties of square matrices. We decided not to use this formalization for the reasons that we discuss at the end of this section.

To implement matrices we used a generic approach by allowing entries in the matrices to be arbitrary elements from some semi-ring structure. For that firstly we expressed semi-rings as a module type. Then we defined matrices as a functor taking as its argument such a semi-ring structure and as a result producing the structure of matrices of arbitrary size with entries from the semi-ring domain.

Internally we represent matrices as vectors of vectors. Vectors are defined in the standard library of Coq (Coq.Bool.BVector) with the type `vector A n` representing a vector of `n` elements of type `A`. Apart from this definition the Coq standard library provides only few basic properties and operations on this type. But on the other hand, building on that, the CoLoR project provides a rich set of results about vectors that were further extended in the course of this development. Here we informally define some of these functions, which we will need later on in the presentation:

$$\begin{aligned}
\text{Vnth } [a_1; \ldots a_n]\ i &= a_i \\
\text{Vfold_left } f\ [a_1; \ldots a_n]\ b &= f\ a_1\ (f \ldots (f\ a_n\ b) \ldots) \\
\text{Vmap } f\ [a_1; \ldots a_n] &= [f\ a_1; \ldots f\ a_n] \\
\text{Vmap2 } f\ [a_1; \ldots a_n]\ [b_1; \ldots b_n] &= [f\ a_1\ b_1; \ldots f\ a_n\ b_n]
\end{aligned}$$

Ability to reuse those results was our main motivation to represent matrices in the following way:

```
Definition matrix (m n : nat) : matrix m n := vector (vector A n) m.
```

Then a number of operations on matrices was defined and some of its properties proven. The library is by no means complete and contains little more than

the results needed for certification of matrix interpretations. The provided operations include: matrix creation (given matrix size and a function providing values for all matrix entries), several accessor functions to retrieve matrix elements, columns and rows, conversions from vectors to 1-row and 1-column matrices and few standard matrix operations such as transposition, addition and multiplication. To show how reusing results about vectors substantially eased our task we present below the definition of multiplication.

First we need a few auxiliary functions on matrices. We begin with three accessor functions: `get_row`, `get_col` and `get_elem` to retrieve, respectively, the i'th row, the j'th column and element at position (i, j) of a given matrix. [1]

```
Definition get_row m n (M : matrix m n) i (ip : i < m) := Vnth M ip.
Definition get_col m n (M : matrix m n) j (jp : j < n) :=
  Vmap (fun v => Vnth v jp) M.
Definition get_elem m n (M : matrix m n) i j (ip : i < m) (jp : j < n) :=
  Vnth (get_row M ip) jp.
```

Note that those functions are partial as indexes `i` and `j` must be within the boundaries of a matrix `M`. In Coq all functions are total and to deal with this we use additional arguments for those functions, the so-called domain predicates, which ensure that the arguments are within the domain of the function.

Next we introduce the `mat_build` function, which constructs a $m \times n$ matrix from two natural numbers `m` and `n`, and a function `f` which, given a matrix position, returns the value of a matrix element to be placed at that position. Again, this function `f` is partial as it is defined only for coordinates i, j such that $0 \leq i < m$ and $0 \leq j < n$.[2] Defining function `mat_build` explicitly is not an easy task due to the presence of domain predicates and dependent types. Therefore we use Coq proving capabilities to prove existence of such a function using its specification.[3]

```
Definition mat_build_spec m n (gen : forall i j, i < m -> j < n -> A),
  { M : matrix m n | forall i j (ip : i < m) (jp : j < n),
    get_elem M ip jp = gen i j ip jp }.
Proof. [...] Defined.
```

and we extract the computational content from the above constructive proof to obtain the required `mat_build` function.

Having all those auxiliary, general purpose functions on vectors and matrices defining matrix multiplication is fairly straightforward. First we introduce a dot product of two vectors as:

```
Definition dot_product (n : nat) (l r : vector A n) : vector A n :=
  Vfold_left Aplus A0 (Vmap2 Amult l r).
```

[1] Note that variables `m`, `n`, `i` and `j` below do not have type annotations as their types can be inferred by Coq and hence can be omitted. In this case all those variables range over natural numbers as a careful reader can easily check.

[2] We index matrix rows and columns starting from 0.

[3] Please note that we are using the Coq mechanism of implicit arguments to skip arguments that can be inferred by Coq due to type dependencies. So for the function `get_elem M i j ip jp` arguments `i` and `j` can be inferred from the domain predicates `ip` and `jp`.

where A0 is the zero element of the domain (the additive identity of the semi-ring) and Aplus is the addition. Then multiplication becomes:

```
Definition mat_mult m n p (L : matrix m n) (R : matrix n p) :=
  mat_build (fun i j ip jp => dot_product (get_row L ip) (get_col R jp)).
```

As can be seen from this example abstracting away natural operations on vectors and matrices and then using them for more complex constructs has big advantages. Not only the definitions became significantly simpler but also reasoning about them, as one can first prove properties about such auxiliary functions and then use them to reason about more complex constructs.

In fact this was the main reason against using the development by Nicolas Magaud, mentioned at the beginning of this section. It provides nice results by proving the ring properties for square matrices. But the fact that it is stand-alone and does not provide this kind of separation as mentioned above, made it difficult to use in our setting. For instance a function for matrix addition is realized there by a relatively complex Fixpoint construct (which is 16 lines long), whereas we can simply write

```
Definition vec_plus n (L R : vector A n) := Vmap2 Aplus L R.
Definition mat_plus m n (L R : matrix m n) := Vmap2 (@vec_plus n) L R.
```

and use all CoLoR properties of Vmap2 to prove properties of matrix addition. Similarly other operations could be expressed easily and concisely by using operations and properties of vectors available in CoLoR.

3.4 Matrix Interpretations

Now we will explain how monotone algebras are instantiated for the matrix interpretation method, so we will develop the Coq counter-part of the theory described in Section 2.3. First we introduce a data type representing a matrix interpretation of a function symbol:

```
Variables (Sig : Signature) (f : symbol Sig) (dim : nat).
Record matrixInt (argCnt : nat) : Type := mkMatrixInt {
  const : vector nat dim;
  args : vector (matrix dim dim) argCnt
}.
```

So matrixInt n is a type of matrix interpretation for a function symbol of arity n, defined as a record with two fields: const being a constant vector of the interpretation of size dim and args representing coefficients for the arguments with a dim×dim matrix per argument. Comparing with equation 1, const represents the $\boldsymbol{f}$ vector and args the list of matrices $F_1, \cdots, F_n$.

Now we enclose all the parameters required for the application of Theorem 2 specialized to the monotone algebra for matrix interpretations, in a module type:

```
Module Type TMatrixInt.
  Parameter sig : Signature.
  Parameter dim : nat.
```

```
  Parameter dim_pos : dim > 0.
  Parameter trsInt : forall f : sig, matrixInt dim (arity f).
End TMatrixInt.
```

So we take a signature `sig`, dimension for matrices (`dim`; d in Section 2.3), a proof that dimension is positive (`dim_pos`) and interpretations for all function symbols of the signature, with respective arities (`trsInt`).

Given those parameters we construct the respective monotone algebra. We cannot present it here in details due to space limitations. The most difficult property was actually decidability of algebra relations $>$ and $\gtrsim$ lifted to terms. This corresponds to proving the 'if' part of Theorem 3. Note that we did not prove the 'only-if' part of that theorem, which state completeness of this characterization and which is not needed for the correctness of the approach. Proving the 'if' part required performing linearization of the computation of a matrix interpretation, such as in Equation 2. Then we proved that evaluating this linearized expression leads to the same result as simply evaluating this expression without any simplifications beforehand. Performing those two steps in Coq required a substantial effort.

4 Evaluation

We already mentioned the termination competition [3,18], the battlefield for termination provers, in Section 3.1. This year, for the first time, a new category of certified termination has been introduced, showing the recognition for the importance of certification efforts. Indeed ensuring reliability of constantly evolving and more and more complex tools is difficult and every year we observe some disqualifications due to erroneous proofs produced by some of the tools.

In this new category every claim made by a termination prover must be backed up by a full formal proof expressed and checked by some well established theorem prover (and not only by a textual informal description of such a proof, as is the case in the standard category). This makes the results reliable with the highest standards of reliability available in verification.

The combination of the CoLoR project (with Rainbow) and the termination prover TPA [13], developed by the first author, was the winning entry in this newly introduced category of the Termination Competition in 2007. It achieved the score of 354, meaning that for 354 out of the total 975 TRSs used in the competition, TPA could find a termination proof and using CoLoR correctness of this proof could be verified by Coq.

Due to the fact that this category was introduced only this year there were only two other participants. The termination prover C*i*ME [8] using the Coccinelle [7] library to certify termination results, again using Coq theorem prover. It got the second place with a score of 317. The third participating tool was the entry of T$_{\mathsf{T}}$T [12] using CoLoR as the certifying back-end with a score of 289.

For comparison we would like to mention that in the standard category, which is run on the same set of problems, the scores ranged from 330 to 723. This shows that many proofs are beyond reach of the certification at the moment,

which is completely understandable. But it also shows that for a substantial part of proofs we can not only produce them with termination tools but also fully automatically ensure their correctness, including difficult problems for which establishing termination results by human is very hard. We believe this is a big step forward and a very promising future for the termination results.

Considering evaluation of our contribution, every single termination proof produced by TPA in the competition was using matrix interpretations at some point. This is not so surprising given the fact that CoLoR, at the moment, is supporting only two basic orders: polynomial and matrix interpretations. But this also shows that for winning the competition, our contribution was crucial.

When it comes to performance finding a proof took TPA on average 2.0 sec and certification required 2.6 sec per system. There were however few systems were the certification time was substantially longer. During the competition verification for 4 problems reached the 5 minutes timeout. Currently we are busy experimenting and trying to improve the performance of the verification routines but, although we did achieve some speedups, so far they were of rather minor effect.

5 Conclusions

We presented our contribution to the CoLoR project — a Coq formalization of matrix interpretations method for proving termination of rewriting. This allows us to fully automatically certify termination of non-trivial rewrite systems, such as the Zantema/z086.srs from the TPDB [4]:

$$\mathrm{a}(\mathrm{a}(x)) \to \mathrm{c}(\mathrm{b}(x)), \quad \mathrm{b}(\mathrm{b}(x)) \to \mathrm{c}(\mathrm{a}(x)), \quad \mathrm{c}(\mathrm{c}(x)) \to \mathrm{b}(\mathrm{a}(x))$$

Until recently termination of this innocent looking system was an open problem [2, Problem 104] and now not only it can be automatically proven terminating by termination tools but also that results can be warranted by Coq.

It is worth noting that typically Coq is used as a proof assistant, where the formalization is built by a human interacting with the system. It is not so in our application as the Coq script formalizing termination of a given system is generated fully automatically by Rainbow from a proof description produced by some termination prover; again, automatically. However the proof assistance capabilities of Coq are crucial for the development of CoLoR.

The natural way of continuing work on certification of termination is to formalize further termination techniques. Although matrix interpretations provide a very powerful base ordering, they do not subsume other orders. Even more advantageous would be formalization of more involved refinements of the dependency pair framework [11]; a modular, powerful approach to proving termination, employed by most, if not all, successful modern termination provers. By supporting relative top termination problems our development is fully ready to benefit from such refinements (see [15] for details), but the CoLoR library at the moment supports only the basic computation of dependency pairs. Implementing extensions such as usable rules or dependency graph approximations is on-going work.

Acknowledgements

We would like to thank Frédéric Blanqui for helpful comments and encouragement for this work.

References

1. The Coq proof assistant, http://coq.inria.fr
2. The RTA list of open problems, http://www.lsv.ens-cachan.fr/rtaloop
3. Termination competition, http://www.lri.fr/~marche/termination-competition
4. Termination problems data base, http://www.lri.fr/~marche/tpdb
5. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theoretical Computer Science 236(1-2), 133–178 (2000)
6. Blanqui, F., Delobel, W., Coupet-Grimal, S., Hinderer, S., Koprowski, A.: CoLoR, a Coq library on rewriting and termination. In: 8th WST (2006)
7. Contejean, É., Courtieu, P., Forest, J., Pons, O., Urbain, X.: Certification of automated termination proofs. In: Konev, B., Wolter, F. (eds.) FroCoS 2007. LNCS(LNAI), vol. 4720, pp. 148–162. Springer, Heidelberg (2007)
8. Contejean, E., Marché, C., Monate, B., Urbain, X.: The CiME rewrite tool, http://cime.lri.fr
9. Endrullis, J., Waldmann, J., Zantema, H.: Matrix interpretations for proving termination of term rewriting. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 574–588. Springer, Heidelberg (2006)
10. Endrullis, J., Waldmann, J., Zantema, H.: Matrix interpretations for proving termination of term rewriting. Journal of Automated Reasoning (accepted, 2007)
11. Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: Combining techniques for automated termination proofs. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS (LNAI), vol. 3452, pp. 301–331. Springer, Heidelberg (2005)
12. Hirokawa, N., Middeldorp, A.: Tyrolean termination tool: Techniques and features. Information and Computation 205(4), 474–511 (2007)
13. Koprowski, A.: TPA: Termination proved automatically. In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 257–266. Springer, Heidelberg (2006)
14. Koprowski, A., Zantema, H.: Certification of matrix interpretations in Coq. In: 9th WST (2007)
15. Koprowski, A., Zantema, H.: Certification of proving termination of term rewriting by matrix interpretations. Technical Report CS-Report 07/22, Eindhoven University of Technology (August 2007), http://www.win.tue.nl/ akoprows/papers/mint-cert-TR.pdf
16. Krauss, A.: Certified size-change termination. In: CADE 2007. LNCS (LNAI), vol. 4603, pp. 460–475. Springer, Heidelberg (2007)
17. Magaud, N.: Ring properties for square matrices. Coq contributions, http://coq.inria.fr/contribs-eng.html
18. Marché, C., Zantema, H.: The Termination Competition 2007. In: RTA 2007. LNCS, vol. 4533, pp. 303–313. Springer, Heidelberg (2007)

Extension of Rescheduling Based on Minimal Graph Cut

Marián Lekavý and Pavol Návrat

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
lekavy@fiit.stuba.sk

Abstract. An important role of a workflow system is to schedule and reschedule the workflow process and to allow the user to monitor and guide the overall progress of the workflow and its activities. This paper presents extensions of a rescheduling algorithm based on minimal graph cut. The initial approach allowed to find an optimal rescheduling, which changed the initial schedule by shortening and moving of activities. The new schedule had the lowest overall price, counted as the sum of prices of shortening activities. The extension presented in this paper allows several extensions, while keeping the optimality: handling several delayed activities, handling the price of moving an activity and handling the price of deadline violation. The rescheduling algorithm is used within an ontology-based workflow management system for the process of military exercise preparation in Centre of simulation technologies National Academy of Defence.

1 Introduction

A common problem is that the schedule is not always 100% respected. If some activity is not accomplished in time, it is necessary to reschedule dependent activities, which require the output of the delayed activity. This is especially important if the schedule is directed towards fulfilling a deadline (DAY-D) and must not exceed this limit. It is therefore necessary to reschedule and shorten the time for dependent activities.

Rescheduling is often bound to replanning if the workflow is not well-defined and stable. This paper, however, focuses on rescheduling, without allowing replanning. In general, there are 4 approaches to rescheduling:

1. **Schedule enforcement, responsibility left on agents.** We can avoid the problem of rescheduling by deliverable-oriented management, enforcing deliverable dates and disallowing an agent to change the schedule in a way, which would affect the schedules of other agents (e.g. EVM [3] used at CERN). If an agent fails to produce a deliverable on time, all affected agents have to agree on a new schedule. This places big confidence into agents' reliability in fulfilling the deliverable dates and the agents need the capability to create a new schedule.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 340–351, 2008.

2. **Include all possibilities into the workflow definition** All possible execution scenarios are listed in the workflow definition (e.g. PROTEUS [4]). It is not necessary to reschedule at run-time, but the manual definition of all possibilities is a difficult process. It is also possible to pre-schedule reserves for unexpected situations, generating a robust schedule [12].
3. **Complete rescheduling (CR)** Probably the most common method of rescheduling (other than manual) is to create a new schedule from scratch (e.g. MicroBoss [16]). This has, however, several disadvantages. The main disadvantage is that we discard previous scheduling information, causing a major waste of computational time. Another disadvantage is that the new schedule is not related to the previous schedule, so the amount of changes compared to the previous schedule can be higher than necessary.
4. **Modify the old schedule.** Many approaches, including human-made rescheduling, use a set of heuristics to correct the initial schedule. The new schedule is created with low effort, but the rescheduling is not guaranteed to be optimal with respect to any metrics (i.e. there are more changes than really necessary). In the human-made rescheduling, it is possible to use various supporting methods, like the ones based on Critical Path Method (CPM) [13] or Program Evaluation and Review Technique (PERT) [13], which allow to optimise the schedule with respect to the schedule length, but don't take the cost of changes into account. There are two basic methods of automatic rescheduling: Right-shift rescheduling (RSR) and Partial rescheduling (PR). RSR simply postpones all activities affected by a failure. PR [15], tries not to modify more activities than necessary. One of PR approaches is the Match-up rescheduling (MUR) [1], which tries to make the new schedule completely consistent with the initial schedule from a certain time point. PR approaches (including MUR) usually use heuristics to find a near-to-optimal solution. There is also an optimal approach based on mixed integer linear programming (MILP)[17]. It uses an overall cost of the schedule, including the rescheduling cost. It constructs all possible reschedulings and selects the one with the lowest cost. Generating all possible reschedulings causes NP-completeness, making the use for large problems difficult.

Most existing approaches focus on creating the schedule with the smallest execution time (makespan), but usually, the change itself also has some cost, including the risks that the changes will cause unexpected delays and endanger the schedule execution. Recently, the stability of rescheduling became more important. In this context, stability is the measure of the impact of disruptions caused by changes of activities, i.e. to achieve stability, we need to minimise the cost of rescheduling. Combining the makespan and stability criteria results in multiobjective solutions [14,2].

Some approaches use actuality penalty function for changing the activities close to current time [14]. This way, they prefer to change activities with later start time, avoiding some organisational problems with rush orders. On the other hand, the actuality penalty is in direct conflict with MUR, because actuality

penalty prefers the rescheduling of later activities, while MUR tries to minimise the impact of schedule changes to the later activities. For the same reason, actuality penalty is in conflict with an unmovable deadline (DAY-D), because it increases the risk at the end of the schedule.

Taking rescheduling cost into account is especially important for projects, where the project execution time is fixed and known in advance and we need to minimise the impact of a delay in some activity. If the deadline cannot be moved and some activity fails to complete on time, then the only possibility to cope with this situation is to shorten the remaining schedule by shortening some activities. We can do this by adding manpower to these activities (perhaps taking it from other, less important projects) or by decreasing the quality of the final project result/product. The possibility to shorten activities is overlooked by rescheduling approaches, but in workflow systems, shortening of activities is used very often as a possibility to compensate a delay.

This paper presents the rescheduling algorithm to be used in an ontology-based workflow management system for the process of military exercise preparation in Centre of simulation technologies National Academy of Defence.

The introduced rescheduling algorithm creates the optimal rescheduling according to some rescheduling metrics (e.g. minimal number of rescheduled tasks or some other rescheduling cost function) by using minimal graph cut. In the worst case, the algorithm has cubic time complexity with respect to number of activities and dependencies. Previously, no algorithms based on graph cut for rescheduling were used. We show, that under the given constraints, minimal rescheduling is equivalent to minimal graph cut.

This paper presents an extended version of the algorithm presented previously [10,11]. The extension additionally handles several delayed activities, the price of moving an activity, the price of deadline violation and the price of aborting the delayed activity.

The algorithm, however, doesn't handle replanning, nor does it handle resources. Therefore, it is only usable for well defined domains, where the necessary tasks are known and planned in advance.

2 The RAPORT System

The RAPORT system [7] is designed for a pilot application: organisation of military exercise in Centre of simulation technologies (CST) National Academy of Defence (NAO) in Liptovský Mikuláš. CST organises training and education for headquarters' staff and commanders with support of information and communication technologies.

CST staff executes the requested activities of the military exercise preparation, while the tasks are distributed among CST staff members. Coordination is made during control meetings. Preparation of several exercises may overlap.

Most similar workflows are organised manually, using office software and paper documents.

The RAPORT knowledge management support system is designed to work for arbitrary administration process. It is designed to fulfil several requirements:

- Prepare necessary information related to the current working context, role in the organisation and role in the workflow process instance (predefined e-mails, documents, forms etc.)
- Support users' experience exchange and collaboration by allowing interactive collaboration and exchanging notes and hints.
- Check current plans for important dates, evaluate the workflow for each agent and activity and adapt the plans if necessary.
- Collect experience from users and present it to users in similar working context.

The RAPORT system combines both: the process-oriented and deliverable-oriented paradigms. The workflow activities are well defined and the system supports these activities by providing necessary documents, process descriptions and guidance to the assigned agents. At the same time, each activity is ended if all output documents for the activity are delivered.

The whole schedule is relative to the date of the exercise - DAY-D. All activities have their accomplishment date relative to DAY-D. At DAY-D, all activities have to be accomplished successfully in order to meet the exercise requirements.

3 Optimal Rescheduling as Minimal Graph Cut

This section provides a basic outline of the rescheduling algorithm and shows, how the rescheduling problem is related to the minimal graph cut problem.

The goal of rescheduling is to compensate the unexpected delay of one or more activities by shortening the schedule behind the delayed activities.

The first step of the rescheduling algorithm is to create a dependence graph for the actual schedule, where each dependence and activity is represented by an edge. Then, we mark the edges with the cost of shortening the edge. If the edge cannot be shortened, the cost of the edge is set to infinite. In this graph, we find the minimal cut and move all parts of the graph before the cut by 1 time unit.

This was the basic outline of the algorithm. Individual steps are discussed in following subsections in more detail.

The fulfilment check of the schedule is done on an every-day basis (or some other unit time), so activities are only shortened or moved by 1 time unit. If we need to move the activities by more than 1 time unit, we have to invoke the rescheduling process several times.

3.1 Rescheduling as Graph Cut

Activities are bound by documents. If an activity A_{use} uses a document D created by another activity $A_{produce}$, then the activity A_{use} is dependent on the activity $A_{produce}$. A_{use} cannot start before $A_{produce}$ ended and created the document D. If some activities depend on each other, but there is no document

transferred between them, a virtual empty document is added to the system to express this dependency.

An activity has three times defined: time of start, time of end and minimal time necessary for execution. Initially, these times come from the workflow default schedule. The start and end time can further be changed due to rescheduling.

So an activity for the purpose of this algorithm is the tuple $A = (id, D_{used}, D_{produced}, t_{start}, t_{end}, t_{min})$, where id is the activity identifier, D_{used} is the set of documents required by the activity, $D_{produced}$ is the set of documents created by the activity, t_{start} and t_{end} are the times of activity start and end in the current schedule and t_{min} is the minimal time needed for the activity ($t_{end} - t_{start} \geq t_{min}$). Activity definition in the domain ontology contains other information like roles, responsible agents, document templates, active hints and others. For the purpose of the rescheduling algorithm, we can neglect these.

From the workflow model (Figure 1a) stored in the domain ontology, we can construct a dependence graph for the activities. In this graph, activities are vertices and the dependencies are edges. This way, we get an acyclic dependence graph (Figure 1b). For the purpose of rescheduling, we remove all activities (and corresponding dependencies), which are not dependent on the activities, which failed to complete in time - $A_{f\#}$ (# stands for the activity identifier).

For the minimal cut algorithm, we also need the activities to be expressed by edges. Therefore, we split each vertex for some activity $A_{\#}$ (except the failed activities $A_{f\#}$) into two activities $A_{\#start}$ and $A_{\#end}$ connected by an edge $e_{A\#}$. All incoming edges of the previous vertex $A_{\#}$ are connected to $A_{\#start}$ and outgoing edges to $A_{\#end}$. Additionally, we add the vertex A_{DAY-D}, representing the final deadline. A_{DAY-D} is dependent on every activity in the schedule, as all activities have to be completed before this deadline. Finally, we add an artificial source A_{init} and artificial sink A_{final}. A_{init} is connected to all failed activities ($A_{f\#}$) and A_{DAY-D} is connected to A_{final} (Figure 1c).

This way, we created a dependency graph, which has a vertex for each event in the schedule, which is possibly affected by the change of duration of the failed activities ($A_{f\#}$). These events are the starts ($A_{\#start}$) and ends ($A_{\#end}$) of dependent activities, the final deadline (A_{DAY-D}) and ends of the failed activities ($A_{f\#}$). (The symbol # stands for the identifier of an activity.)

Theorem 1. *Every possible rescheduling, which changes the schedule by postponing the start and/or end times of some activities by 1 time unit, corresponds to some cut of the dependency graph.*

Proof (sketch). We define two sets: $V_{changed}$ containing vertices which are postponed in the rescheduling and $V_{unchanged}$ containing vertices which are not affected by the rescheduling. Each event of the dependency graph ($A_{\#start}$, $A_{\#end}$, A_{DAY-D}, $A_{f\#}$; # standing for an activity identifier) is either postponed or not, i.e. it belongs to $V_{changed}$ or $V_{unchanged}$, while $V_{changed} \cap V_{unchanged} = \emptyset$. A graph cut is the division of a graph into two sets, dividing two vertices (in this case $A_{init} \in V_{changed}$ and $A_{final} \in V_{unchanged}$). This means that every possible combination of sets $(V_{changed}, V_{unchanged})$ representing some rescheduling is a cut of the dependency graph. □

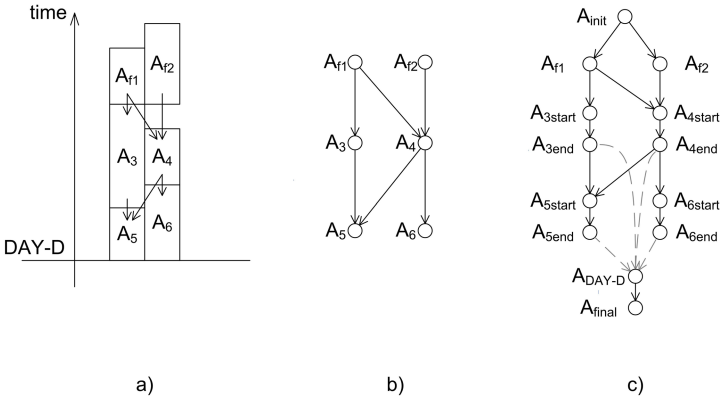


Fig. 1. A simple example of conversion from activities and dependences (a) to oriented graph (b), splitting the activity vertices and adding the deadline, initial and final vertices (c)

Several cases may occur in the distribution of events in sets ($V_{changed}$, $V_{unchanged}$) representing the graph cut.

- If an activity $A_{\#}$ is moved forward by 1 time unit, both, the start and stop time of the activity are increased ($A_{\#start} \in V_{changed}$, $A_{\#stop} \in V_{changed}$).
- If an activity $A_{\#}$ is shortened by 1 time unit, only the start time of the activity is increased. That means $A_{\#start} \in V_{changed}$ and $A_{\#stop} \in V_{unchanged}$.
- It is also possible to expand an activity by increasing the stop time ($A_{\#start} \in V_{unchanged}$, $A_{\#stop} \in V_{changed}$). Expanding an activity is usually not very useful when shortening the schedule and is only recommended by the algorithm if the cost of shortening is negative.
- For activities which are not changed, both $A_{\#start}, A_{\#stop} \in V_{unchanged}$.
- The ends of the failed activities $A_{f\#}$ should be in $V_{changed}$, because the goal of the algorithm is to move them. However, the cost function may allow not to move some failed activities.
- The final deadline (A_{DAY-D}) should be in $V_{unchanged}$. Otherwise, the deadline is violated. (More details in the Cost of the Edges section.)
- The initial activity $A_{init} \in V_{changed}$ and the final activity $A_{final} \in V_{unchanged}$.

The edges inside $V_{changed}$ or $V_{unchanged}$ represent activities or dependences, whose length remains unchanged, as we move both, starting and ending vertex or none of them. Activities and dependencies represented by edges going from $V_{changed}$ to $V_{unchanged}$ are shortened by 1 time unit, because their starting vertices are moved, but the ending vertices are not (Figure 2).

If the minimal cut capacity (maximal graph flow, the minimal cost of shortening the schedule) is infinite, then it is impossible to reschedule without violating the constraints on dependencies and minimal activity duration. In that case, the

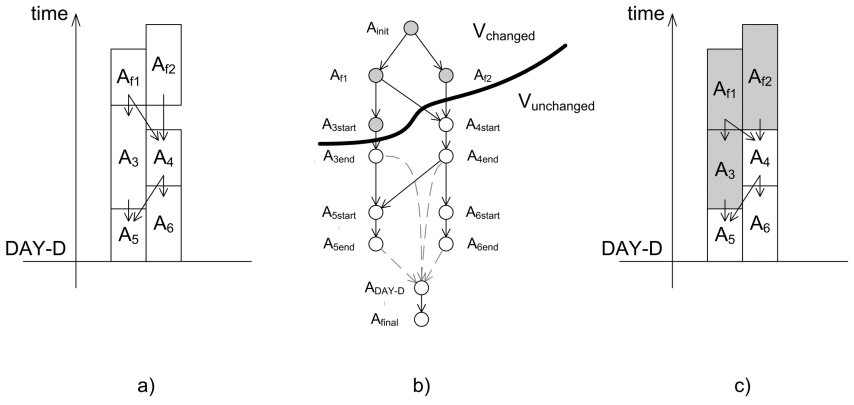


Fig. 2. A simple example of rescheduling minimal cut. From the original schedule (a), the dependence is created and according to the minimal cut (b), the delay of activities A_{f1} and A_{f2} is compensated by shortening of activity A_3 (c).

system notifies the responsible persons about a critical state, when the workflow cannot be fulfilled on time.

We can also extend Theorem 1 to rescheduling by more than 1 time unit.

Theorem 2. *Every possible rescheduling corresponds to a superposition of dependency graph cuts.*

Proof (sketch). We can divide the starting and ending vertices of all activities into several sets, each of them containing vertices moved by the same amount of time ($V_i = \{v | v \text{ is moved by } i \text{ time units}\}$). For each combination of sets $(V_i)_{i=0}^n$, we can simply construct n cuts $(V_{changed_j}, V_{unchanged_j})_{j=0}^n$ in a way that each $v \in V_i$ belongs to i sets $V_{changed_j}$ and to $n - i$ sets $V_{unchanged_j}$. □

Theorem 2 shows that we can express every rescheduling as a sequence of several graph cuts, thus it is possible to use several iterations of the proposed algorithm for more complex, k-step reschedulings. This theorem, however, does not say that a minimal k-step rescheduling corresponds to a superposition of minimal graph cuts. This is only guaranteed if the shortening is commutative (i.e. cost of shortening of some edge by n time units is equal to the sum of n shortenings by 1 time unit, regardless of the order in which this and the other edges are shortened), reversible (the sum cost of sequential shortening and expanding an edge by 1 time unit is zero) and the law of diminishing returns is not violated (the cost of shortening an edge does not decrease by its shortening). This includes constant costs of the edges during all iterations. If these conditions are not fulfilled, the superposition of minimal graph cuts can be more expensive than the minimal k-step rescheduling. On the other hand, minimal graph k-cut problem is NP-complete. Therefore, k minimal graph cuts, each having cubic time complexity, can be used as a cheap polynomial approximation, but with no guaranty being minimal.

3.2 Cost of the Edges

Each graph cut in the dependence graph, dividing the initial vertex and the final vertex, represents one possible schedule change. We try to find the change, which would shorten the activities at minimal costs. Therefore, we have to assign costs to the individual edges, relative to the cost of shortening the edge (and corresponding activity or dependence). The cost will then be assigned to the edge maximal throughput.

The meaning of edges' costs/throughputs in a dependency graph is following:

- Shortening an activity: Cost of $e_{A_\#} = (A_{\#start}, A_{\#stop})$ is the cost of shortening the activity $A_\#$.
- Shortening a dependence: Cost of $e_{A_iA_j} = (A_{istop}, A_{jstart})$ (or (A_{fi}, A_{jstart}) for the failed activities) is the cost of shortening the delay between activities A_i and A_j.
- Moving an activity: Cost of $e_{A_\#D} = (A_{\#stop}, A_{DAY-D})$ is the cost of moving the activity $A_\#$ (i.e. shortening the delay between $A_\#$ and the final deadline A_{DAY-D}).
- Violating the deadline: Cost of $e_{A_DA_{final}} = (A_{DAY-D}, A_{final})$ is the cost of violating the deadline, i.e. the cost of moving the whole schedule. This cost is usually infinite, as we want to avoid deadline violation.
- Not moving a failed activity: Cost of $e_{A_{init}A_{f\#}} = (A_{init}, A_{f\#})$ is the cost of not moving (i.e. aborting) the failed activity $A_{f\#}$. Sometimes, it is possible to abort some activity instead of changing the remaining schedule, but for most activities, it is not possible and the cost should be infinite.

Theorem 3. *If the costs of rescheduling are expressed by the above cost/throughput policy and no other costs are connected to the rescheduling, then the minimal cut of the dependency graph corresponds to the rescheduling with the lowest cost and the cost is equal to the minimal cut capacity.*

Proof (sketch). It is obvious that if the meaning of edges' throughput is as described above, then the minimal cut is the rescheduling with the minimal cost, while the minimal cost is equal to the cut capacity. The minimal graph cut capacity is the sum of costs of edges going across the cut. It is easy to show that cost of some edge is added to the sum iff the shortening of this edge has the effect mentioned above:

- The cost of $e_{A_\#} = (A_{\#start}, A_{\#stop})$ is added to the cut iff the cut goes across this edge, i.e. $A_{\#start} \in V_{changed}$, $A_{\#stop} \in V_{unchanged}$, which means that the activity $A_\#$ was shortened.
- The cost of $e_{A_iA_j} = (A_{istop}, A_{jstart})$ (or (A_{fi}, A_{jstart}) for the failed activities) is added to the cut iff the cut goes accross this edge, i.e. $A_{istop} \in V_{changed}$, $A_{jstart} \in V_{unchanged}$, which means that the delay between A_i and A_j was shortened.
- The cost of $e_{A_\#D} = (A_{\#stop}, A_{DAY_D})$ is just a special case of $e_{A_iA_j}$.
- The cost of $e_{DA_{final}} = (A_{DAY-D}, A_{final})$ is added to the cut iff the cut goes across this edge, i.e. $(A_{DAY-D} \in V_{changed}$ (and $A_{final} \in V_{unchanged})$, which means that the deadline was moved.

– The cost of $e_{A_{init}A_{f\#}} = (A_{init}, A_{f\#})$ is added to the cut iff the cut goes across this edge, i.e. $A_{f\#} \in V_{unchanged}$ (and $A_{init} \in V_{changed}$), which means that the end of the failed activity $A_{f\#}$ was not moved.

(Note that a vertex is moved forward in time by 1 time unit iff the vertex belongs to $V_{changed}$.) The resulting capacity of the minimal cut is the sum of capacities of edges going across the cut (i.e. between $V_{changed}$ and $V_{unchanged}$) and is therefore equal to the cost of changes made in the schedule. □

We can set the cost of shortening an activity to 1 and shortening a dependence to 0. This way, we say that we want to shorten as few activities as possible and don't care about shortening dependencies. The cost of moving an activity can be set to a small number (e.g. 0.001) to reduce unnecessary moving of activities, while placing the main focus on the cost of shortening. This cost policy is used in the RAPORT system. We don't allow violation of the deadline ($cost(e_{DA_{final}}) = \infty$) and aborting of a failed activity ($cost(e_{A_{init}A_{f\#}}) = \infty$).

We could also use a different cost policy. Activities can have different costs defined. Or we can prefer the shortening of activities with the highest time reserve. The cost of shortening an activity can change (usually increase) after shortening the activity. This means that if we are invoking the algorithm several times, we may have to re-compute the costs of affected activities.

We have to assure that an activity or dependence will not be shortened below its minimal duration. If an activity already has its minimal duration or a dependence has duration 0, the cost of shortening is set to infinite.

We can manually give several "commands" to the algorithm by changing the costs of the edges. This way, we can manually adjust the result of the algorithm in almost arbitrary way, when necessary. For example, the shortening or movement of activity $A_{\#}$ can be disallowed by setting the cost of the $e_{A_{\#}}$ or $e_{A_{\#}D}$ edges to infinite. The same way, we can "encourage" the algorithm to shorten or move some activity by setting the costs of the $e_{A_{\#}}$ or $e_{A_{\#}D}$ edges to zero. Possible commands supported by the rescheduling algorithm and used in the RAPORT system are listed in Table 1. Commands for enforcing/allowing/disallowing of deadline violation or aborting a failed activity by modifying the costs of edges $e_{DA_{final}}$ and $e_{A_{init}A_{f\#}}$ are also possible, but are not included here because deadline violation or aborting an activity is not admissible in the RAPORT system.

Additionally, other changes of the cost function may be done manually when necessary. On the other hand, changes other than the ones listed in Table 1 have to be done carefully, because they require deeper understanding of the underlying rescheduling algorithm.

3.3 Minimal Cut and Time Complexity

The presented rescheduling algorithm is in fact just conversion of the rescheduling problem to the minimal cut problem. These two problems are equivalent.

We use the Ford-Fulkerson algorithm [6] in our prototype implementation, however any algorithm for finding minimal cut can be used. There are minimal

Table 1. Possible manual cost changes, affecting the rescheduling

Command	Cost function change
Don't shorten activity $A_\#$	$cost(e_{A_\#}) = \infty$
Shorten activity $A_\#$ when necessary	$cost(e_{A_\#}) = 0$
Shorten activity $A_\#$	$cost(e_{A_\#}) = -\infty$
Don't move activity $A_\#$	$cost(e_{A_\# D}) = \infty$
Move activity $A_\#$ when necessary	$cost(e_{A_\# D}) = 0$
Move activity $A_\#$	$cost(e_{A_\# D}) = -\infty$
Don't shorten the time reserve between activities A_i and A_j	$cost(e_{A_i A_j}) = \infty$

cut algorithms with lower time complexity. The only requirement is the ability to work with infinite edge throughputs.

The conversion of the schedule to graph and back is very straightforward (time complexity $O(|A|+|D|)$, where $|A|$ is the number of activities, $|D|$ is the number of dependencies), so the main complexity issue is to find the minimal cut. The Ford-Fulkerson algorithm we use has time complexity $O((|A|+|D|)^2 * |A|)$. This is also the complexity of the whole rescheduling algorithm.

It is hard to compare our algorithm to existing approaches, as we were not able to find an algorithm solving the same set of problems. We chose 2 approaches, which seem to be closest to our approach and the authors published time measurements for at least some problems. PRDO [2] only creates sequential schedule and does not allow shortening of activities. For 200 tasks, PRDO needs up to 30s, while our approach only needs 0.2-20ms on similar hardware (Pentium4@1.8GHz). The MILP-based rescheduling [17] handles resources, but does not allow shortening of activities. For 22 tasks, it needs 130-330ms (on unknown hardware), while our approach needs 0.1-10ms. Additionally, MILP is NP-complete, so from [17] no predictions can be made how the approach scales.

4 Future Work

This paper showed that the minimal graph cut algorithm can be used to find the rescheduling with the minimal cost with respect to the cost of moving or shortening an activity. The method can be used for automatic or semi-automatic rescheduling. In the RAPORT project, we use semi-automatic rescheduling, allowing the human user to keep full control of the new schedule and to modify it if necessary. This is especially important in the first year of use, as the cost function is not verified and it may be necessary to modify it according to the experience with the system. Later, it may be possible to switch to automated rescheduling. The human intervention, however, will always be allowed. The semi-automatic method can also be used to adjust actions' cost. If the user allows/disallows the shortening of some edge, the cost of the edge will be slightly decreased/increased the next time by default.

It will be very useful to combine this method with existing scheduling methods, especially CPM or PERT -based methods [13]. For the semi-automatic use,

these methods can provide the human user with additional information about the schedule, like the list of critical activities and overall time reserve for optimistic/pessimistic execution scenario. For the automatic use, it is possible to use this additional information to modify the cost function, e.g. by increasing the cost of shortening of near-to-critical activities with low time reserve.

The challenge for the future is adding resources. This is, for example, possible by adding resource links, as proposed in the RCPM scheduling algorithm [8]. The resources links would form additional dependencies in the dependency graph.

As a side effect, we also plan to visualise the algorithm for educational purposes, to support our e-learning courses in theoretical computer science [5].

5 Conclusions

The presented rescheduling algorithm is able to compensate the violation of a deadline by moving and shortening of dependent activities.

The presented rescheduling algorithm uses the cost information and creates the schedule with minimal cost in polynomial (cubic) time.

We showed that the minimal-cost rescheduling problem with costs expressed as costs of moving/shortening of activities and shortening of dependencies is convertible to the minimal graph cut problem.

It can be proven (the proof is out of scope of this paper) that the presented algorithm can also find minimal cost k-step rescheduling if the shortening is commutative, reversible and the law of diminishing returns is not violated. This, however, doesn't guarantee optimality if two rescheduling requests arrive at different times.

The rescheduling algorithm presented in this paper is suitable for workflows under following conditions:

1. The workflow process is well-defined and planned. We know the activities' interdependencies. All activities are planned for execution for an exact time.
2. Activities may be shortened, but not below a critical length. There is a "nominal" length of an activity, which can be further modified if necessary, but not below the critical length.
3. Cost of rescheduling can be expressed as the sum of costs for moving/shortening of activities and shortening of delays between activities.
4. Workflow participants (agents) are willing to accept the new schedule, as long as the new schedule doesn't violate the condition 2.

There is no other limitation on the agents, participating in the workflow. The RAPORT system, for which the algorithm is designed, contains only human agents. The rescheduling can also be used for artificial agents or hybrid systems.

Acknowledgments. This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT 51-024604; by the Slovak Research and Development Agency under the contract No. APVV-0391-06; by the Scientific Grant Agency of Slovak Republic, grant No. VG1/3102/06.

References

1. Bean, J.C., Birge, J.R., Mittenehal, J., Noon, C.E.: Match-up scheduling with multiple resources, release dates and disruption. Operations Research 39(3), 470–483 (1991)
2. Bing, W., Yu-Geng, X.: Rolling Partial Rescheduling with Dual Objectives for Single Machine Subject to Disruptions. Acta Automatica Sinica 32(5), 667–673 (2006)
3. Bonnal, P., De Jonghe, J., Ferguson, J.: A Deliverable-Oriented Evm System Suited to a Large-Scale Project. Project Management Journal (2006)
4. Corkill, D.D., Rubinstein, Z.B., Lander, S.E., Lesser, V.R.: Live-Representation Process Managment. In: Proc. 5th International Conference on Enterprise Information Systems, Angers, France (2003)
5. Chuda, D.: Evaluation and Security Features in e-learning. In: e-learning Conference 2007, Istanbul, Turkey, pp. 81–85 (2007)
6. Ford, L.R., Fulkerson, D.R.: Maximal Flow Through a Network. Canadian Journal of Mathematics 8, 399–404 (1956)
7. Forgac, R., Budinska, I., Gatial, E., Nguyen, G., Laclavik, M., Balogh, Z., Mokris, I., Hluchy, L., Ciglan, M., Babik, M.: Ontology based knowledge management for organizational learning. In: ISIM 2006. Proc. of 9-th Intl. Conf. Information Systems Implementation and Modelling, Brno, April, MARQ Ostrava, pp. 177–184 (2006)
8. Kim, K.: A Resource-constrained CPM (RCPM) Scheduling and Control Technique with Multiple Calendars. Dissertation, Faculty of Virginia Polytechnic Institute and State University, USA (2003)
9. Kučera, L.: Kombinatorické algoritmy. Praha, SNTL (1993)
10. Lekavý, M., Návrat, P.: Dynamic Workflow Schedule Adjusting. In: INFORMATICS 2007. Proceedings of the Ninth International Conference on Informatics, SSAKI, Bratislava, pp. 117–123 (2007) ISBN 978-80-969243-7-0
11. Lekavý, M., Návrat, P.: Dynamic Rescheduling as a Minimal Graph Cut Problem. In: Software Engineering in Progress: Work in Progress section of CEE-SET 2007, Poznań, pp. 141–153 (2007)
12. Lin, X., Janak, S.L., Floudas, C.: A new robust optimization approach for scheduling under uncertainty: I. Bounded uncertainty. Computers and Chemical Engineering 28, 1069–1085 (2004)
13. Moder, J.J., Phillips, C.R., Davis, E.W.: Project management with CPM, PERT, and precedence diagramming. Van Nostrand Reinhold Company, NY (1983)
14. Pfeiffer, A., Kádár, B., Monostori, L.: Stability-oriented evaluation of hybrid rescheduling methods in a job-shop with machine breakdowns. In: 39th CIRP international seminar on manufacturing systems. The morphology of innovative manufacturing systems, Ljubljana, pp. 173–178 (2006)
15. Sabucuonglu, I., Bayiz, M.: Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research 126, 567–586 (2000)
16. Sadeh, N.: Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling, Ph.D. Thesis. School of Computer Science, Carnegie Mellon University (1991)
17. Vin, J.P., Ierapetritou, M.G.: A new approach for efficient rescheduling of multiproduct batch plants. Industrial Engineering and Chemical Research 39, 4228–4238 (2000)

Deriving Complexity Results for Interaction Systems from 1-Safe Petri Nets

Mila Majster-Cederbaum and Christoph Minnameier*

Institut für Informatik
Universität Mannheim, Germany
cmm@informatik.uni-mannheim.de

Abstract. Interaction systems are a formal model for component-based systems, where components are combined via connectors to form more complex systems. We compare interaction systems (*IS*) to the well-studied model of 1-safe Petri nets (*1SN*) by giving a translation map_1: *1SN* $\rightarrow$ *IS* and a translation map_2: *IS* $\rightarrow$ *1SN*, so that a 1-safe Petri net (an interaction system) and its according interaction system (1-safe Petri net) defined by the respective mapping are isomorphic up to some label relation R. So in some sense both models share the same expressiveness. Also, the encoding map_1 is polynomial and can be used to reduce the problems of reachability, deadlock and liveness in *1SN* to the problems of reachability, deadlock and liveness in *IS*, yielding PSPACE-hardness for these questions.

1 Introduction

In [GS03], Gössler and Sifakis presented *interaction systems*, a model for component-based concurrent systems. As typical for component-based systems, interaction systems display two different layers of description: On the one hand the components, which are used to describe the communicating units, together with the ports over which they communicate. On the other hand the *glue code*, i.e. the information about the way components may communicate with each other. I/O-Automata [LT89] and interface automata [dAH01] can be considered as subclasses of interaction systems, for the latter feature a more general notion of communication. E.g. interaction systems allow different degrees of parallelism, i.e. different interactions may involve different numbers of participants.

Interaction systems seem to be an appropriate model for a variety of different types of systems. More details about interaction systems and their properties can be found in [Sif04, Sif05, GGM+07b, GGM+07a, MMM07b, MMM07a]. A framework for component-based modelling using interaction systems has been implemented in the BIP-project [BBS06, GQ07, BS07] and applied to [BMP+07]. Furthermore, interaction systems have been used to model biochemical reactions [MSW07] and they serve as a commom semantic framework for the SPEEDS-project [BCSM07].

* Corresponding author.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 352–363, 2008.

The aim of this paper is to answer some relevant questions concerning the inherent complexity of properties of interaction systems. A first result, concerning these matters is given in [Min07], where it is shown that the problems of local and global deadlock are NP-hard. Here, we obtain stronger results by establishing a relation between interaction systems and the well-studied model of 1-safe Petri nets.

In [CEP93] important results about PSPACE-completeness of behavioral questions in 1-safe Petri nets have been established, which is the starting point of our investigation. In particular, we consider the traditional Petri net token-game semantics for Petri nets, which does not allow the concurrent performance of multiple transitions (even if their presets and postsets are disjoint). In other words, we restrict ourselves to the intrinsic concurrency of the Petri net model, i.e. the fact that a transition may already involve multiple places.

This decision is natural for our purpose of comparing the model of 1-safe nets to the model of interaction systems, because in the latter's semantics, we also may concurrently perform multiple actions within an interaction but only one interaction at a time.

The main part of this work consists of giving mappings from one model to the other and isomorphism relations for the resulting pairs of nets and systems.

The mappings and isomorphism relations are then used to derive PSPACE-hardness results for some important behavioral questions for interaction systems, namely reachability, global deadlock and liveness.

The paper is organized as follows. Section 2 contains the basic definitions. Sections 3 and 4 give the respective translations between 1-safe Petri nets and interaction systems. Section 5 contains a conclusion and a discussion of related work.

2 Definitions

2.1 1-Safe Petri Nets

A ***Petri net*** [CEP93] is a fourtuple $N = (P, T, F, M_0)$ such that:

- P and T are finite disjoint sets. Their elements are called ***places*** and ***transitions***, respectively.
- $F \subseteq (P \times T) \cup (T \times P)$. F is called the ***flow relation***.
- $M_0 : P \to \mathbb{N}$ is called the ***initial marking*** of N. In general, a mapping $M : P \to \mathbb{N}$ is called a ***marking*** of N. By $\mathcal{M}$ we denote the set of all markings of a net.

For places as well as transitions we define the notion of preset and postset:
For $p \in P$, ***preset(p)*** $:= \{t \in T \mid (t, p) \in F\}$, ***postset(p)*** $:= \{t \in T \mid (p, t) \in F\}$.
For $t \in T$, ***preset(t)*** $:= \{p \in P \mid (p, t) \in F\}$, ***postset(t)*** $:= \{p \in P \mid (t, p) \in F\}$.

For technical reasons we only consider nets in which every node has a nonempty preset or a nonempty postset. We let $+$ denote the union of multisets.

Let $N = (P, T, F, M_0)$ be a Petri net. A transition $t \in T$ is ***enabled*** under a marking M if $M(p) > 0$ for every place p in the preset of t. Given a transition t, we define a relation $\stackrel{t}{\rightarrow}_N$ as follows: $M \stackrel{t}{\rightarrow}_N M'$ if t is enabled under M and $M'(p) = M(p) + F(t,p) - F(p,t)$, where $F(x,y)$ is 1 if $(x,y) \in F$ and 0 otherwise. We say that the transition t is performed at M. We define the global transition system (or global behavior) T_N of N by $T_N = (\mathcal{M}, T, \rightarrow_N, M_0)$.

For $M, M' \in \mathcal{M}$, we write $\boldsymbol{M \rightarrow_N^* M'}$ if there are ($k \in \mathbb{N}$ and) markings $M^1, \ldots, M^k \in \mathcal{M}$ and transitions $t_1, \ldots, t_{k+1} \in T$ that build a transition sequence $M \stackrel{t_1}{\rightarrow}_N M^1 \stackrel{t_2}{\rightarrow}_N \ldots \stackrel{t_k}{\rightarrow}_N M^k \stackrel{t_{k+1}}{\rightarrow}_N M'$ in T_N.

A marking M of a net N is called 1-safe, if for every place p of the net $M(p) \leq 1$. We identify a 1-safe marking with the set of places such that $M(p) = 1$. A net N is called 1-safe if all its reachable markings are 1-safe. 1-safe nets are a well studied computation model. The following questions are known to be PSPACE-complete [CEP93].

The ***reachability*** problem for 1-safe nets consists of deciding, given a 1-safe net $N = (P, T, F, M_0)$ and a marking M of N, whether $M_0 \rightarrow_N^* M$.

The ***liveness*** problem for 1-safe nets consists of deciding, given a 1-safe net $N = (P, T, F, M_0)$ if every transition can always occur again. More precisely, if for every reachable marking M and every transition t, there is $M' \in \mathcal{M}$ with $M \rightarrow_N^* M'$ and M' enables t.

The ***deadlock*** problem for 1-safe nets consists of deciding, given a 1-safe net $N = (P, T, F, M_0)$, if every reachable marking enables some transition. If this is the case we call the net deadlock-free.

Example 1

The 1-safe net N_1 is given (by its graphical representation) in Figure 1. N_1 is deadlock free and even live and the set of reachable markings is $\{\{p_1, p_2, p_3\}, \{p_3, p_4, p_5\}, \{p_1, p_6\}\}$.

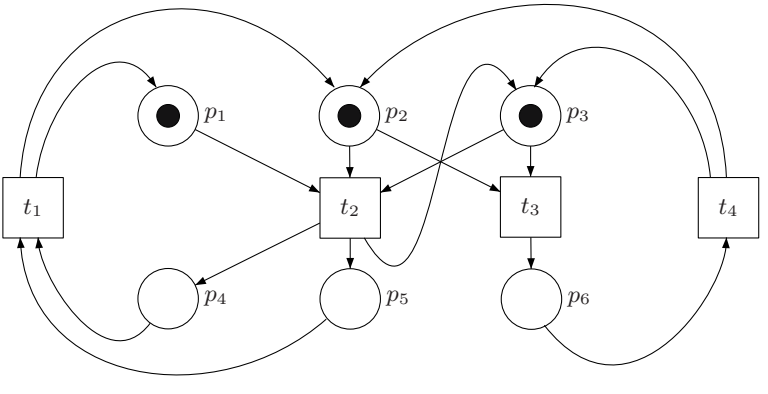


Fig. 1. A 1-safe net N_1

2.2 Interaction Systems

We review here interaction systems, a model for component-based systems that was proposed and discussed in detail in [GS03, Sif05, GS05, BBS06, GGM+07b, GGM+07a, MMM07a]. An ***interaction system*** is a tuple $Sys = (K, \{A_i\}_{i\in K}, C, Comp, \{T_i\}_{i\in K})$, where K is the set of ***components***. W.l.o.g. we assume $K = \{1, \ldots, n\}$. Each component $i \in K$ offers a finite set A_i of ***ports*** (also called ***actions***) for cooperation with other components. The ***port sets*** A_i are pairwise disjoint. Cooperation is described by connectors and complete interactions. A ***connector*** is a finite set of actions $c \subseteq \bigcup_{i\in K} A_i$, subject to the constraint that for each component i at most one action $a_i \in A_i$ is in c. A connector $c = \{a_{i_1}, \ldots, a_{i_k}\}$ with $a_{i_j} \in A_{i_j}$ describes that the components $i_1, \ldots, i_j$ cooperate via these ports.

A ***connector set*** C is a finite set of connectors, s.t. every action of every component occurs in at least one connector of C and no connector contains any other connector. Sometimes not all components involved in a connector are ready to perform their respective action. Still, we might want to allow those that are ready to go on. For this we may designate certain subsets of connectors as ***complete interactions***. Let $Comp$ be a designated set of complete interactions. $Comp$ has to be upwards-closed w.r.t. C, i.e.: $\forall \alpha \in Comp\ \forall c \in C\ ((\alpha \subset \alpha' \subseteq c) \Rightarrow \alpha' \in Comp)$.

We call $Int := C \cup Comp$ the set of ***interactions***[1]. (The distinction between connectors and complete interactions is irrelevant for our encodings).

The local behavior of each component i is described by a transition system $T_i = (Q_i, A_i, \rightarrow_i, q_i^0)$, where Q_i is the finite set of local states, $\rightarrow_i \subseteq Q_i \times A_i \times Q_i$ the local transition relation and $q_i^0 \in Q_i$ is the local starting state.

Given an interaction $\alpha \in Int$ and a component $i \in K$ we denote by $i(\alpha) := A_i \cap \alpha$ the ***participation*** of i in α. For ease of notation, we identify a singleton set with its element.

For $q_i \in Q_i$ we define the set of ***enabled actions*** $ea(q_i) := \{a_i \in A_i \mid \exists q_i' \in Q_i, \text{s.t. } q_i \xrightarrow{a_i}_i q_i'\}$. We assume that the T_i's are non-terminating, i.e. $\forall i \in K\ \forall q_i \in Q_i\ ea(q_i) \neq \emptyset$.

The ***global behavior*** $T_{Sys} = (Q, Int, \rightarrow_{Sys}, q^0)$ of Sys (henceforth also referred to as global transition system) is obtained from the behaviors of the individual components, given by the transition systems T_i, and the interactions Int in a straightforward manner:

- $Q = \prod_{i\in K} Q_i$, the Cartesian product of the Q_i, which we consider to be order independent. We denote states by tuples $(q_1, \ldots, q_n)$ and call them global states.
- the relation $\rightarrow_{Sys} \subseteq Q \times Int \times Q$, defined by
  $\forall \alpha \in Int\, \forall q, q' \in Q\ \ q = (q_1, \ldots, q_n) \xrightarrow{\alpha}_{Sys} q' = (q_1', \ldots, q_n')$ iff
  $\forall i \in K\ \ (q_i \xrightarrow{i(\alpha)}_i q_i'$ if $i(\alpha) \neq \emptyset$ and $q_i' = q_i$ otherwise).
- $q^0 = (q_1^0, \ldots, q_n^0)$ is the starting state for Sys.

[1] In the original nomenclature of [GS03], subsets of connectors in general are called interactions. This more general notion of interaction is however only needed for the purpose of composing interaction systems out of smaller interaction systems.

Less formally, a transition labeled by α may take place in the global transition system when each component i participating in α is ready to perform $i(\alpha)$.

Example 2
Let $Sys_1 = \{\{1,2,3\}, \{A_i\}_{1\leq i\leq 3}, C, Comp, \{T_i\}_{1\leq i\leq 3})$, where $A_1 = \{a_1, b_1\}$, $A_2 = \{a_2, b_2\}, A_3 = \{a_3, b_3, d_3\}$, $C = \{\{a_1, a_2, a_3\}, \{b_1, b_2, b_3\}, \{d_3\}\}$, $Comp = \{\{b_1, b_2\}\}$ and the local transition systems T_i are given in Figure 2.

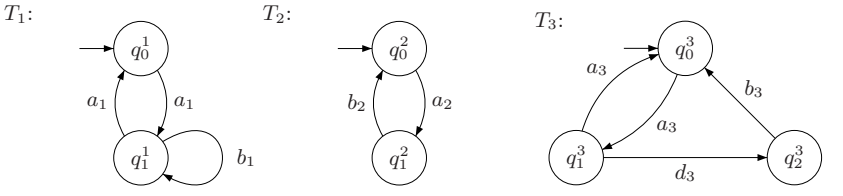


Fig. 2. The T_i's for Sys_1

For the following definitions let Sys be an interaction system:

Let $\rightarrow^*_{Sys}$ denote the reflexive and transitive closure of $\rightarrow_{Sys}$.

Given a state $q \in Q$ we denote by ***reachability of q*** the question, whether q is reachable in T_{Sys}, i.e. whether $q^0 \rightarrow^*_{Sys} q$.

The question whether Sys contains a **global deadlock** (henceforth simply referred to as a deadlock) is the question whether there is a reachable global state q such that $q \not\rightarrow$.

We say **a component** $i \in K$ ***is live***[2] in Sys, if for any reachable global state there is some $q' \in Q$ with $q \rightarrow^*_{Sys} q'$ such that there exist $\alpha \in Int$ and $q'' \in Q$ with $q' \stackrel{\alpha}{\rightarrow}_{Sys} q''$, where i participates in α.

If a component $i \in K$ is live in Sys then at each reachable global state a clever scheduler can continue in such a way that eventually an interaction may be performed in which i participates.

2.3 Isomorphism up to a Label Relation *R*

We define a notion of isomorphism, namely isomorphism up to a label relation R, which we use to establish a relation between transition systems that use different label sets L_1 and L_2. R then defines which labels in L_1 we want to correspond to which labels in L_2.

Let $T_i = (Q_i, L_i, \rightarrow_i, q_i^0)$, $i \in \{1, 2\}$ be two labeled transition systems. Given a *label relation* $R \subseteq (L_1 \times L_2)$, that relates labels of L_1 to labels of L_2, we say that T_1 and T_2 are isomorphic up to R iff there exists a bijective function $f : Q_1 \rightarrow Q_2$, such that $f(q_1^0) = q_2^0$ and $\forall q_1 \in Q_1, q_2 \in Q_2$ the following two propositions hold:

[2] Note that this notion of liveness does not coincide with the one defined in [MMM07a].

1) $q_1 \xrightarrow{l_1}_1 q_1' \;\Rightarrow\; \exists l_2 \in L_2$, s.t. $(l_1, l_2) \in R \wedge f(q_1) \xrightarrow{l_2}_2 f(q_1')$.
2) $q_2 \xrightarrow{l_2}_2 q_2' \;\Rightarrow\; \exists l_1 \in L_1$, s.t. $(l_1, l_2) \in R \wedge f^{-1}(q_2) \xrightarrow{l_1}_1 f^{-1}(q_2')$.

We say an interaction system and a 1-safe net are isomorphic up to a label relation R iff this holds for their respective global transition systems.

3 Translating 1-Safe Nets to Interaction Systems

Let $N = (P, T, F, M_0)$ be a 1-safe net. We give a translation map_1 from 1-safe Petri nets to interaction systems as follows. We introduce a component $\hat{p}$ for each place $p \in P$. The transition system $T_{\hat{p}}$ has only two states, one state $s^1_{\hat{p}}$ to reflect the fact that p contains a token, one state $s^0_{\hat{p}}$ to reflect that it doesn't. The transitions t adjacent to p define the transition relation of $T_{\hat{p}}$, where we distinguish three cases:

a) $t \in (preset(p) \setminus postset(p))$. When such a transition is performed in N, this means that p is empty before the performance of t and contains a token afterwards. Thus, we introduce an edge from $s^0_{\hat{p}}$ to $s^1_{\hat{p}}$ labeled by $a_{(t,p)}$.
b) $t \in (postset(p) \setminus preset(p))$. Inverse to a), i.e. we introduce an edge from $s^1_{\hat{p}}$ to $s^0_{\hat{p}}$ labeled by $a_{(p,t)}$.
c) $t \in (preset(p) \cap postset(p))$. This means there has to be a token in p to perform t and there will still be one there afterwards. In this case, we introduce a loop at $s^1_{\hat{p}}$ labeled by $a_{(t,p,t)}$.

For an example of a place with pre- and postset resp. its corresponding component, see Figure 3 (a) resp. (b). (Note that only edges adjacent to p are depicted.)

Now we define a connector $c(t)$ for each transition t. For the places adjacent to t again we distinguish three cases:

a) $p \in (preset(t) \setminus postset(t))$. This means that in order to perform t, there has to be a token in p, and there will be no token in p after performing t. Thus we include the action $a_{(p,t)}$ in $c(t)$ which already occurs in the component $\hat{p}$ in such a way that this fact is perfectly reflected.
b) $p \in (postset(t) \setminus preset(t))$. Inverse to a), i.e. we include the action $a_{(t,p)}$ in $c(t)$.
c) $p \in (preset(t) \cap postset(t))$. This means that in order to perform t, there has to be a token in p, and there still be a token in p after performing t. Thus we include the action $a_{(t,p,t)}$ in $c(t)$ which already occurs in the component $\hat{p}$ in the corresponding way.

For an example of a transition with pre- and postset resp. its corresponding connector, see Figure 4 (a) resp. (b). (Note that only edges adjacent to t are depicted.)

Formal definition of map_1:

$map_1(N) = \{K, \{A_i\}_{i\in K}, C, Comp, \{T_i\}_{i\in K}\}$, where

$$\boldsymbol{K} := \{\hat{p} \mid p \in P\}$$

$$\begin{aligned}
\text{For } \hat{p} \in K: A^{in}_{\hat{p}} &:= \{a_{(t,p)} \mid \exists t \in T, \text{ s.t. } p \in (preset(t) \setminus postset(t))\},\\
A^{out}_{\hat{p}} &:= \{a_{(p,t)} \mid \exists t \in T, \text{ s.t. } p \in (postset(t) \setminus pretset(t))\},\\
A^{inout}_{\hat{p}} &:= \{a_{(t,p,t)} \mid \exists t \in T, \text{ s.t. } p \in (preset(t) \cap postset(t))\}, \text{ and}\\
\boldsymbol{A_{\hat{p}}} &:= A^{in}_{\hat{p}} \cup A^{out}_{\hat{p}} \cup A^{inout}_{\hat{p}}.
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{T_{\hat{p}}} &:= (\{s^0_{\hat{p}}, s^1_{\hat{p}}\}, A_{\hat{p}}, \rightarrow_{\hat{p}}, q^{\hat{p}}_s), \text{ where } A_{\hat{p}} \text{ has already been given,}\\
\rightarrow_{\hat{p}} &:= \{(s^0_{\hat{p}}, a_{(t,p)}, s^1_{\hat{p}}) \mid a_{(t,p)} \in A^{in}_{\hat{p}}\}\\
&\quad \cup \{(s^1_{\hat{p}}, a_{(p,t)}, s^0_{\hat{p}}) \mid a_{(p,t)} \in A^{out}_{\hat{p}}\}\\
&\quad \cup \{(s^1_{\hat{p}}, a_{(t,p,t)}, s^1_{\hat{p}}) \mid a_{(t,p,t)} \in A^{inout}_{\hat{p}}\}\\
q^0_{\hat{p}} &:= s^0_{\hat{p}} \text{ if } M_0(p) = 0 \text{ and } q^0_{\hat{p}} := s^1_{\hat{p}} \text{ if } M_0(p) = 1.
\end{aligned}$$

In order to define a connector for a transition we now relate the actions in $\bigcup_{i\in K} A_i$ to the transitions in the way described above.

$$\begin{aligned}
\text{For } t \in T\text{: } A^{in}_t &:= \{a_{(p,t)} \mid p \in (preset(t) \setminus postset(t))\},\\
A^{out}_t &:= \{a_{(t,p)} \mid p \in (postset(t) \setminus pretset(t))\},\\
A^{inout}_t &:= \{a_{(t,p,t)} \mid p \in (preset(t) \cap postset(t))\}\\
c(t) &:= A^{in}_t \cup A^{out}_t \cup A^{inout}_t\\
\boldsymbol{C} &:= \{c(t) \mid t \in T\}\\
\boldsymbol{Comp} &:= \emptyset
\end{aligned}$$

It remains to prove that C is indeed a connector set.

We observe that $\{A^{sup}_t \mid t \in T, sup \in \{in, out, inout\}\}$ is a disjoint decomposition of $\bigcup_{i\in K} A_i$. This is due to the fact that the A^{sup}_t's are defined following the definition of the $A^{sup}_{\hat{p}}$'s. C is just a coarser decomposition obtained from the one above by merging some of the disjoint subsets. So each action occurs exactly once in a connector, i.e. it occurs in at least one connector and no connector can be a subset of another connector.

Also, as $Comp = \emptyset$ we have upwards-closedness of $Comp$ w.r.t. C.

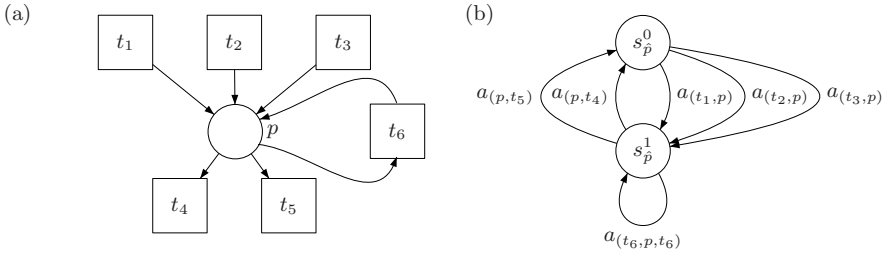


Fig. 3. A place with ingoing and outgoing transitions and its corresponding component

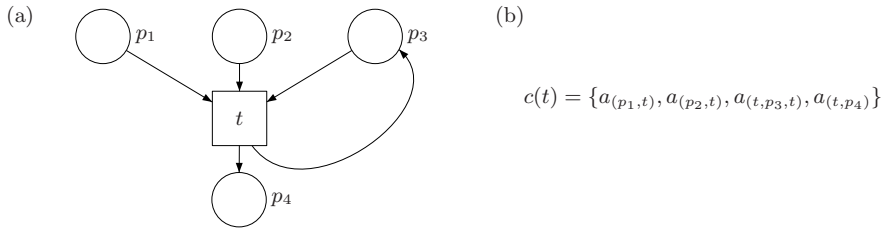


Fig. 4. A transition with its pre- and postset and its corresponding connector

Example 1 continued
Let $N_1 = (P, T, F, M_0)$ be the 1-safe net from Example 1. The corresponding interaction system is $map_1(N_1) = \{\{1, \ldots, 6\}, \{A_i\}_{1 \le i \le 6}, C, \emptyset, \{T_i\}_{1 \le i \le 6}\}$, where
$C = \{\{\ a_{(p_4,t_1)}, a_{(p_5,t_1)}, a_{(t_1,p_1)}, a_{(t_1,p_2)}\}, \{a_{(p_1,t_2)}, a_{(p_2,t_2)}, a_{(t_2,p_4)}, a_{(t_2,p_5)},$
$a_{(p_3,t_2,p_3)}\}, \{a_{(p_2,t_3)}, a_{(p_3,t_3)}, a_{(t_3,p_6)}\}, \{a_{(p_6,t_4)}, a_{(t_4,p_3)}, a_{(t_4,p_2)}\}\}$
and the T_i's (and implicitely the A_i's) are given in Figure 5.

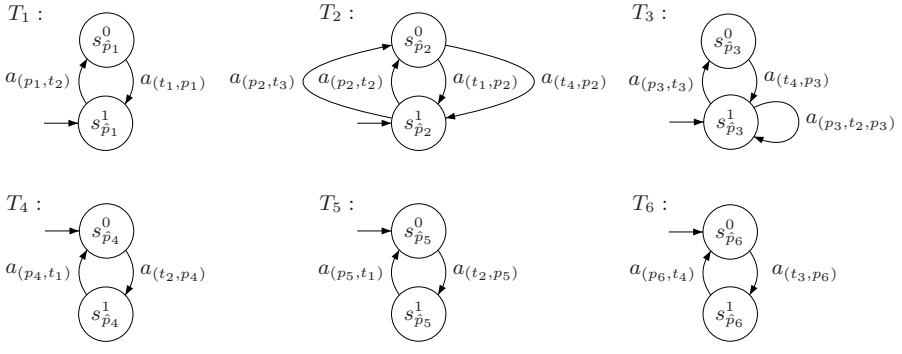


Fig. 5. The T_i's for $map_1(N_1)$

Let q be a global state of $map_1(N)$. Then $q(\hat{p})$ denotes the projection of q to $\hat{p}$.

Theorem 1. *Let N be a 1-safe net and $Sys = map_1(N)$. With $R := \{(c,t) \in (Int \times T) \mid c = c(t)\}$ and with the bijection $f : Q \to \mathcal{M}$, defined by $f(q) = \{p \in P \mid q(\hat{p}) = s^1_{\hat{p}}\}$ we have defined an isomorphism up to R for Sys and N.*

Let *Sys* be an interaction system. We consider questions for typical properties and prove them PSPACE-hard using Theorem 1 (and, of course, building on the evident fact that map_1 can be determined in polynomial time):

Corollary 1. *The question, whether some state q can be reached in Sys is PSPACE-hard.*

We know that the reachability question for 1-safe nets, i.e. the question whether some marking M is reachable in N is PSPACE-hard [CEP93].

By Theorem 1 we know that this question can be answered by answering instead the reachability question for M's corresponding global state $f^{-1}(M)$ in *Sys*.

Corollary 2. *The question, whether Sys is free of global deadlock is PSPACE-hard.*

We know that the question of deadlock for 1-safe nets, i.e. the question whether there is a reachable marking M in N where no transition is enabled is PSPACE-hard [CEP93]. By Theorem 1 we may conclude that this is the case iff there is a reachable global state in $map_1(N)$, where no interaction is enabled.

Corollary 3. *The question, whether a component $i \in K$ is live is PSPACE-hard.*

We know that the question of liveness for 1-safe nets, i.e. the question whether every transition can always occur again is PSPACE-hard [CEP93].

As liveness in 1-safe nets concerns transitions, which are translated to interactions, and, in contrast, liveness in interaction systems concerns components, we introduce a place p_t for each transition t, such that the place's corresponding component $\hat{p}_t$ will be live iff t can always occur again. This can be done by employing a (polynomial) preencoding map_{pre} on N before applying map_1.

More formally, let $map_{pre}(N) = (P \cup \{p_t \mid t \in T\}, T, F \cup \{(p_t, t), (t, p_t) \mid t \in T\}, M_0 \cup \{p_t \mid t \in T\})$.

Now N is live iff every $\hat{p}_t$ $(t \in T)$ is live in $map_1(map_{pre}(N))$.

4 Translating Interaction Systems to 1-Safe Nets

In this section, we present the encoding map_2 from interaction systems to 1-safe nets. Our interest in such a translation is mainly of theoretic nature, i.e. we want to gain more understanding of the properties of these two models. Still, as interaction models are a relatively young model, for which so far not many tools have been developed, there is some practical benefit: One could translate a system into a net and apply Petri net tools in order to investigate some behavioral questions of the system.

Let $Sys = (K, \{A_i\}_{i \in K}, C, Comp, \{T_i\}_{i \in K})$ be an interaction system. We introduce a place $\hat{q}_i$ for each local state $q_i \in Q_i$ of a component $i \in K$. A global state of *Sys* is a tuple of the present local states of the components, so for every reachable state in N, there will always be exactly one place $\hat{q}_i$ for each $i \in K$ that contains a token. This reflects that q_i is the present state of component i.

It remains to translate the glue code given by the interactions *Int* to the notion of transition. An action a_i in A_i may occur multiple times in the local transition system T_i of component i. Thus the performance of an interaction α may cause differnet state changes in Sys.

As a consequence we are going to map an interaction α not to a single transition but to a set of transitions $T(\alpha)$. Each transition in $T(\alpha)$ represents one of

these possible global state changes and will shift the tokens in N according to the local state changes that are caused for the components that participate in α.

More formally, we define the mapping map_2 from interaction systems to 1-safe nets as follows:
$map_2(Sys) = (P, T, F, M_0)$, where
$P = \bigcup_{i \in K} \{\hat{q}_i \mid q_i \in Q_i\}$.

For $\alpha = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\} \in Int$, we introduce a set of transitions $T(\alpha) := \{\{(q_{i_1}, a_{i_1}, q'_{i_1}), \ldots, (q_{i_k}, a_{i_k}, q'_{i_k})\} \mid \forall 1 \leq j \leq k (q_{i_j}, a_{i_j}, q'_{i_j}) \in \rightarrow_{i_j}\}$.

Then we define $T = \bigcup_{\alpha \in Int} T(\alpha)$.

For each α and each transition $t = \{(q_{i_1}, a_{i_1}, q'_{i_1}), \ldots, (q_{i_k}, a_{i_k}, q'_{i_k})\}$ in $T(\alpha)$ we introduce arcs as follows:
$F(t) = \{(\hat{q}_{i_1}, t), \ldots, (\hat{q}_{i_k}, t)\} \cup \{(t, \hat{q}'_{i_1}), \ldots, (t, \hat{q}'_{i_k})\}$
$F(\alpha) = \bigcup_{t \in T(\alpha)} F(t)$.
$F = \bigcup_{\alpha \in Int} F(\alpha)$.
$M_0 = \{\hat{q}_i \in P \mid q_i = q_i^0\}$.

This means that in the initital marking exactly those places that correspond to the local starting states of the components contain a token.

Remark: Let T_i be the local labeled transition system of component i and let $a_i \in A_i$ be an action of i. We denote the number of ***occurences*** of a_i in T_i by $occ(a_i)$. Note that for one interaction $\alpha = \{a_{i_1}, \ldots, a_{i_k}\}$ there are $(occ(a_{i_1}) \cdot \ldots \cdot occ(a_{i_k}))$ instances of α. This means we might have exponentially (in n) many instances for a single interaction α, which will result in an exponential blowup in our mapping from interaction systems to 1-safe nets. (See, e.g. Example 2, where we would gain $occ(a_1) \cdot occ(a_2) \cdot occ(a_3) = 2 \cdot 1 \cdot 2 = 4$ transitions of the interaction $\{a_1, a_2, a_3\}$ in $T(\{a_1, a_2, a_3\})$.)

Theorem 2. *Let Sys be an interaction system and $N = map_2(Sys)$. With $R := \{(\alpha, t) \in (Int \times T) \mid t \in T(\alpha)\}$ and with the bijection $f : Q \rightarrow \mathcal{M}$, defined by $f(q_1, \ldots, q_n) = \{\hat{q}_1, \ldots, \hat{q}_n\}$ we have defined an isomorphism up to R for Sys and N.*

Remark: One application of our translation of interaction systems to Petri nets is to answer behavioral questions for an interaction system *Sys* by translating it to a 1-safe net and answering the (corresponding) question there. Also the translation preserves component identity, i.e. a component i is represented in $map_2(Sys)$ exactly by the places $\{\hat{q}_i \mid q_i \in Q_i\}$.

5 Conclusion and Related Work

Interaction systems are a model for component-based systems. The increasing relevance of interaction systems demands a profound theoretical basis for this model. In this paper we study complexity results for interaction systems. We do so by establishing a relation between the model of interaction systems and the well-studied model of 1-safe Petri nets for which complexity results have been investigated in [CEP93]. We show that anything described by a 1-safe net

can easily be described by an interaction system without a blowup in notation. Similarly, interaction systems can be translated into 1-safe nets. However, it seems unavoidable to have a (worst case) exponential blowup for this translation.

The results with the greatest impact are that the problems of deadlock-freeness and reachability are PSPACE-hard for interaction systems. These are the first PSPACE-hardness results concerning interaction systems and they partially outrun the complexity results given in [Min07]. The established results provide an essential basis for future work: Given these "master"-reductions we may extend the PSPACE-hardness results (by polynomial reductions) to almost all behavioral questions for interaction systems.

Furthermore these results suggest that there is no polynomial time algorithm for solving the questions of deadlock, reachability or liveness in interaction systems and thus provide further motivation for approaches to establish desired properties: e.g. finding sufficient conditions for deadlock-freeness and other properties of interaction systems such as the ones given in [MMM07a] and [GGM+07a] that can be tested in polynomial time or methods making use of compositionality. The model of interaction systems is particularly suited for applying these approaches because they exploit local information about components, whose identities are preserved when composing the interaction system. In contrast to this Petri nets lack compositionality and the identity of a component is lost when a composite system is modeled by a Petri net.

References

[BBS06] Basu, A., Bozga, M., Sifakis, J.: Modeling Heterogeneous Real-time Components in BIP. In: SEFM 2006. Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods, pp. 3–12. IEEE Computer Society, Washington, DC, USA (2006)

[BCSM07] Bozga, M., Constant, O., Skipper, M., Ma, Q.: Speeds Meta-Model Syntax and Static Semantics (2007)

[BMP+07] Basu, A., Mounier, L., Poulhis, M., Pulou, J., Sifakis, J.: Using BIP for Modeling and Verification of Networked Systems - A Case-Study on Tinyos-Based Networks. Technical Report, Verimag, Centre Équation (2007)

[BS07] Bliudze, S., Sifakis, J.: The algebra of connectors: structuring interaction in bip. In: EMSOFT 2007. Proceedings of the 7th ACM & IEEE international conference on Embedded software, pp. 11–20. ACM, New York (2007)

[CEP93] Cheng, A., Esparza, J., Palsberg, J.: Complexity Results for 1-safe Nets. In: Shyamasundar, R.K. (ed.) FSTTCS 1993. LNCS, vol. 761, pp. 326–337. Springer, Heidelberg (1993)

[dAH01] de Alfaro, L., Henzinger, T.: Interface automata. In: Proceedings of FSE 2001, ACM Press, New York (2001)

[GGM+07a] Goessler, G., Graf, S., Majster-Cederbaum, M., Martens, M., Sifakis, J.: An Approach to Modelling and Verification of Component Based Systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, Springer, Heidelberg (2007)

[GGM+07b] Goessler, G., Graf, S., Majster -Cederbaum, M., Martens, M., Sifakis, J.: Ensuring Properties of Interaction Systems by Construction. In: Program Analysis and Compilation, Theory and Practice. LNCS, vol. 4444, Springer, Heidelberg (2007)

[GQ07] Graf, S., Quinton, S.: Contracts for BIP: Hierarchical Interaction Models for Compositional Verification. In: FORTE 2007. LNCS, vol. 4574, pp. 1–18. Springer, Heidelberg (2007)

[GS03] Goessler, G., Sifakis, J.: Component-based Construction of Deadlock-free Systems. In: Pandya, P.K., Radhakrishnan, J. (eds.) FST TCS 2003. LNCS, vol. 2914, pp. 420–433. Springer, Heidelberg (2003)

[GS05] Goessler, G., Sifakis, J.: Composition for Component-based Modeling. Sci. Comput. Program. 55(1-3), 161–183 (2005)

[LT89] Lynch, N.A., Tuttle, M.R.: An Introduction to Input/Output Automata. In: CWI-Quarterly, pp. 219–246 (1989)

[Min07] Minnameier, C.: Local and Global Deadlock-Detection in Component-based Systems are NP-hard. In: Information Processing Letters 3630 (2007)

[MMM07a] Majster-Cederbaum, M., Martens, M., Minnameier, C.: A Polynomial-time Checkable Sufficient Condition for Deadlock-Freedom of Component-based Systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, Springer, Heidelberg (2007)

[MMM07b] Majster-Cederbaum, M., Martens, M., Minnameier, C.: Liveness in Interaction Systems. In: Proceedings of FACS 2007, ENTCS (2007)

[MSW07] Majster-Cederbaum, M., Semmelrock, N., Wolf, V.: Interaction Models for Biochemical Reactions. In: BioComp (2007)

[Sif04] Sifakis, J.: Modeling Real-time Systems. In: Keynote talk RTSS 2004 (2004)

[Sif05] Sifakis, J.: A Framework for Component-based Construction (extended abstract). In: SEFM, pp. 293–300 (2005)

Computing Longest Common Substring and All Palindromes from Compressed Strings

Wataru Matsubara[1], Shunsuke Inenaga[2], Akira Ishino[1], Ayumi Shinohara[1], Tomoyuki Nakamura[1], and Kazuo Hashimoto[1]

[1] Graduate School of Information Science, Tohoku University, Japan
{matsubara@shino., ishino@, ayumi@, nakamura@aiet., hk@aiet.}ecei.tohoku.ac.jp
[2] Department of Computer Science and Communication Engineering, Kyushu University, Japan
inenaga@c.csce.kyushu-u.ac.jp

Abstract. This paper studies two problems on compressed strings described in terms of *straight line programs* (*SLPs*). One is to compute the length of the longest common substring of two given SLP-compressed strings, and the other is to compute all palindromes of a given SLP-compressed string. In order to solve these problems efficiently (in polynomial time w.r.t. the compressed size) decompression is never feasible, since the decompressed size can be exponentially large. We develop combinatorial algorithms that solve these problems in $O(n^4 \log n)$ time with $O(n^3)$ space, and in $O(n^4)$ time with $O(n^2)$ space, respectively, where n is the size of the input SLP-compressed strings.

1 Introduction

The importance of algorithms for *compressed texts* has recently been arising due to the massive increase of data that are treated in compressed form. Of various text compression schemes introduced so far, *straight line program* (*SLP*) is one of the most powerful and general compression schemes. An SLP is a context-free grammar of either of the forms $X \to YZ$ or $X \to a$, where a is a constant. SLP allows *exponential* compression, i.e., the original (uncompressed) string length N can be exponentially large w.r.t. the corresponding SLP size n. In addition, resulting encoding of most grammar- and dictionary-based text compression methods such as LZ-family [1,2], run-length encoding, multi-level pattern matching code [3], Sequitur [4] and so on, can quickly be transformed into SLPs [5,6,7]. Therefore, it is of great interest to analyze what kind of problems on SLP-compressed strings can be solved in polynomial time w.r.t. n. Moreover, for those that are polynomial solvable, it is of great importance to design efficient algorithms. In so doing, one has to notice that decompression is never feasible, since it can require exponential time and space w.r.t. n.

The first polynomial time algorithm for SLP-compressed strings was given by Plandowski [8], which tests the equality of two SLP-compressed strings in $O(n^4)$ time. Later on Karpinski et al. [9] presented an $O(n^4 \log n)$-time algorithm for

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 364–375, 2008.

the substring pattern matching problem for two SLP-compressed strings. Then it was improved to $O(n^4)$ time by Miyazaki et al. [10] and recently to $O(n^3)$ time by Lifshits [11]. The problem of computing the minimum period of a given SLP-compressed string was shown to be solvable in $O(n^4 \log n)$ time [9], and lately in $O(n^3 \log N)$ time [11]. Gąsieniec et al. [5] claimed that all squares of a given SLP-compressed string can be computed in $O(n^6 \log^5 N)$ time.

On the other hand, there are some hardness results on SLP-compressed string processing. Lifshits and Lohrey [12] showed that the subsequence pattern matching problem for SLP-compressed strings is NP-hard, and that computing the length of the longest common subsequence of two SLP-compressed strings is also NP-hard. Lifshits [11] showed that computing the Hamming distance between two SLP-compressed strings is #P-complete.

In this paper we tackle the following two problems: one is to compute the length of the *longest common substring* of two SLP-compressed strings, and the other is to find all maximal *palindromes* of an SLP-compressed string. The first problem is listed as an open problem in [11]. This paper closes the problem giving an algorithm that runs in $O(n^4 \log n)$ time with $O(n^3)$ space. For second the problem of computing all maximal palindromes, we give an algorithm that runs in $O(n^4)$ time with $O(n^2)$ space.

Comparison to previous work. *Composition system* is a generalization of SLP which also allows "truncations" for the production rules. Namely, a rule of composition systems is of one of the following forms: $X \to Y^{[i]}Z_{[j]}$, $X \to YZ$, or $X \to a$, where $Y^{[i]}$ and $Z_{[j]}$ denote the prefix of length i of Y and the suffix of length j of Z, respectively. Gąsieniec et al. [5] presented an algorithm that computes all maximal palindromes from a given composition system in $O(n \log^2 N \times Eq(n))$ time, where $Eq(n)$ denotes the time needed for the equality test of composition systems. Since $Eq(n) = O(n^4 \log^2 N)$ in [5], the overall time cost is $O(n^5 \log^4 N)$.

Limited to SLPs, $Eq(n) = O(n^3)$ due to the recent work by Lifshits [11]. Still, computing all maximal palindromes takes $O(n^4 \log^2 N)$ time in total, and therefore our solution with $O(n^4)$ time is faster than the previous known ones (recall that $N = O(2^n)$). The space requirement of the algorithm by Gąsieniec et al. [5] is unclear. However, since the equality test algorithm of [11] takes $O(n^2)$ space, the above-mentioned $O(n^4 \log^2 N)$-time solution takes at least as much space as ours.

2 Preliminaries

For any set U of pairs of integers, we denote $U \oplus k = \{(i+k, j+k) \mid (i,j) \in U\}$. We denote by $\langle a, d, t\rangle$ the arithmetic progression with the minimal element a, the common difference d and the number of elements t, that is, $\langle a, d, t\rangle = \{a + (i-1)d \mid 1 \le i \le t\}$. When $t = 0$, let $\langle a, d, t\rangle = \emptyset$.

Let Σ be a finite *alphabet*. An element of Σ^* is called a *string*. The length of a string T is denoted by $|T|$. The empty string ε is a string of length 0, namely, $|\varepsilon| = 0$. For a string $T = XYZ$, X, Y and Z are called a *prefix*, *substring*, and

suffix of T, respectively. The i-th character of a string T is denoted by $T[i]$ for $1 \leq i \leq |T|$, and the substring of a string T that begins at position i and ends at position j is denoted by $T[i:j]$ for $1 \leq i \leq j \leq |T|$. For any string T, let T^R denote the reversed string of T, namely, $T^R = T[|T|] \cdots T[2]T[1]$.

For any two strings T, S, let $LCPref(T, S)$, $LCStr(T, S)$, and $LCSuf(T, S)$ denote the length of the *longest common prefix, substring and suffix* of T and S, respectively.

A *period* of a string T is an integer p $(1 \leq p \leq |T|)$ such that $T[i] = T[i+p]$ for any $i = 1, 2, \ldots, |T| - p$.

A non-empty string T such that $T = T^R$ is said to be a *palindrome*. When $|T|$ is even, then T is said to be an *even palindrome*, that is, $T = SS^R$ for some $S \in \Sigma^+$. Similarly, when $|T|$ is odd, then T is said to be an *odd palindrome*, that is, $T = ScS^R$ for some $S \in \Sigma^*$ and $c \in \Sigma$. For any string T and its substring $T[i:j]$ such that $T[i:j] = T[i:j]^R$, $T[i:j]$ is said to be the *maximal palindrome* w.r.t. the center $\lfloor \frac{i+j}{2} \rfloor$, if either $T[i-1] \neq T[j+1]$, $i = 1$, or $j = |T|$. In particular, $T[1:j]$ is said to be a *prefix palindrome* of T, and $T[i:|T|]$ is said to be a *suffix palindrome* of T.

In this paper, we treat strings described in terms of *straight line programs* (*SLPs*). A straight line program $\mathcal{T}$ is a sequence of assignments such that

$$X_1 = expr_1, X_2 = expr_2, \ldots, X_n = expr_n,$$

where each X_i is a variable and each $expr_i$ is an expression in either of the following form:

- $expr_i = a \quad (a \in \Sigma)$, or
- $expr_i = X_\ell X_r \quad (\ell, r < i)$.

Denote by T the string derived from the last variable X_n of the program $\mathcal{T}$. The *size* of the program $\mathcal{T}$ is the number n of assignments in $\mathcal{T}$.

When it is not confusing, we identify a variable X_i with the string derived from X_i. Then, $|X_i|$ denotes the length of the string derived from X_i.

For any variable X_i of $\mathcal{T}$ with $1 \leq i \leq n$, we define X_i^R as follows:

$$X_i^R = \begin{cases} a & \text{if } X_i = a \quad (a \in \Sigma), \\ X_r^R X_\ell^R & \text{if } X_i = X_\ell X_r \quad (\ell, r < i). \end{cases}$$

Let $\mathcal{T}^R$ be the SLP consisting of variables X_i^R for $1 \leq i \leq n$.

Lemma 1. *SLP $\mathcal{T}^R$ derives string T^R.*

Proof. By induction on the variables X_i^R. Let Σ_T be the set of characters appearing in T. For any $1 \leq i \leq |\Sigma_T|$, we have $X_i = a$ for some $a \in \Sigma_T$, thus $X_i^R = a$ and $a = a^R$. Let T_i denote the string derived from X_i. For the induction hypothesis, assume that X_j^R derives T_j^R for any $1 \leq j \leq i$. Now consider variable $X_{i+1} = X_\ell X_r$. Note $T_{i+1} = T_\ell T_r$, which implies $T_{i+1}^R = T_r^R T_\ell^R$. By definition, we have $X_{i+1}^R = X_r^R X_\ell^R$. Since $\ell, r < i+1$, by the induction hypothesis X_{i+1}^R derives $T_r^R T_\ell^R = T_{i+1}^R$. Thus, $\mathcal{T}^R = X_n^R$ derives $T_n^R = T^R$. □

Note that $\mathcal{T}^R$ can be easily computed from $\mathcal{T}$ in $O(n)$ time.

3 Computing Longest Common Substring of Two SLP Compressed Strings

Let $\mathcal{T}$ and $\mathcal{S}$ be the SLPs of sizes n and m, which describe strings T and S, respectively. Without loss of generality we assume that $n \geq m$.

In this section we tackle the following problem:

Problem 1. Given two SLPs $\mathcal{T}$ and $\mathcal{S}$, compute $LCStr(T, S)$.

In what follows we present an algorithm that solves Problem 1 in $O(n^4 \log n)$ time and $O(n^3)$ space. Let X_i and Y_j denote any variable of $\mathcal{T}$ and $\mathcal{S}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.

3.1 Overlaps between Two Strings

For any two strings X and Y, we define the set $OL(X, Y)$ as follows:

$$OL(X, Y) = \{k > 0 \mid X[|X| - k + 1 : |X|] = Y[1 : k]\}$$

Namely, $OL(X, Y)$ is the set of lengths of overlaps of suffixes of X and prefixes of Y. Karpinski et al. [9] gave the following results for computation of OL for strings described by SLPs.

Lemma 2 ([9]). *For any variables X_i and X_j of an SLP $\mathcal{T}$, $OL(X_i, X_j)$ can be represented by $O(n)$ arithmetic progressions.*

Theorem 1 ([9]). *For any SLP $\mathcal{T}$, $OL(X_i, X_j)$ can be computed in total of $O(n^4 \log n)$ time and $O(n^3)$ space for any $1 \leq i \leq n$ and $1 \leq j \leq n$.*

As we will show in the sequel, we need to compute $OL(X_i, Y_j)$ and $OL(Y_j, X_i)$ for any $1 \leq i \leq n$ and $1 \leq j \leq m$. In so doing, we produce a new variable $V = X_n Y_m$, that is, V is a concatenation of SLPs $\mathcal{T}$ and $\mathcal{S}$. Then we compute OL for each pair of variables in the new SLP of size $n + m$. On the assumption that $n \geq m$, it takes $O(n^4 \log n)$ time and $O(n^3)$ space in total.

3.2 The *FM* Function

For any two variables X_i, Y_j and integer k with $1 \leq k \leq |X_i|$, we define the function $FM(X_i, Y_j, k)$ which returns the previous position of the first position of mismatches, when we compare Y_j with X_i at position k. Formally,

$$FM(X_i, Y_j, k) = \min\{1 \leq h \leq |Y_j| \mid X_i[k + h - 1] \neq Y_j[h]\} - 1.$$

Namely, $FM(X_i, Y_j, k)$ equals the length of the common prefix of $X_i[k : |X_i|]$ and Y_j when it is not zero. When the common prefix is the empty string ε (when such h does not exist), let $FM(X_i, Y_j, k) = 0$.

Lemma 3 ([9]). *For any variables X_i, Y_j and integer k, $FM(X_i, Y_j, k)$ can be computed in $O(n \log n)$ time, provided that $OL(X_{i'}, X_{j'})$ is already computed for any $1 \leq i' \leq i$ and $1 \leq j' \leq j$.*

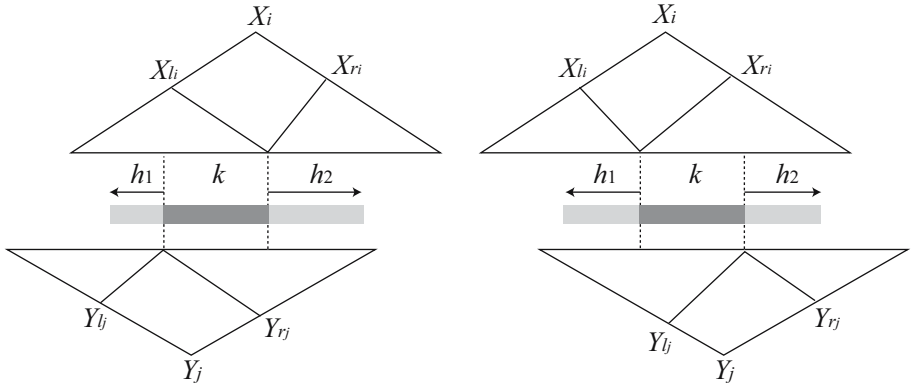


Fig. 1. Illustration of Observation 2 where we "extend" an overlap k as a candidate of $LCStr(T, S)$

3.3 Efficient Computation of Longest Common Substrings

The main idea of our algorithm for computing $LCStr(T, S)$ is based on the following observation.

Observation 1. *For any substring Z of string T, there always exists a variable $X_i = X_{\ell_i} X_{r_i}$ of SLP $\mathcal{T}$ such that:*

- *Z is a substring of X_i and*
- *Z touches or covers the boundary between X_{ℓ_i} and X_{r_i}.*

It directly follows from the above observation that any common substring of strings T, S touches or covers both of the boundaries in X_i and Y_j for some $1 \leq i \leq n$ and $1 \leq j \leq m$.

For any SLP variables $X_i = X_{\ell_i} X_{r_i}$ and $Y_j = Y_{\ell_j} Y_{r_j}$, and $k \in OL(X_i, Y_j)$, let $Ext_{X_i,Y_j}(k) = k + h_1 + h_2$ such that $h_1 = LCSuf(X_{\ell_i}[1 : |X_{\ell_i}| - k], Y_{\ell_j})$ and $h_2 = LCPref(X_{r_i}, Y_{r_j}[k + 1 : |Y_{r_j}|])$. For any $k \notin OL(X_i, Y_j)$, we leave $Ext_{X_i,Y_j}(k)$ undefined. For a set S of integers, we define $Ext_{X_i,Y_j}(S) = \{Ext_{X_i,Y_j}(k) \mid k \in S\}$. $Ext_{Y_j,X_i}(k)$ and $Ext_{Y_j,X_i}(S)$ are defined similarly.

The next observation follows from the above arguments (see also Fig. 1):

Observation 2. *For any strings T and S, $LCStr(T, S)$ equals to the maximum element of the set*

$$\bigcup_{1 \leq i \leq n, 1 \leq j \leq m} (Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j})) \cup Ext_{Y_j,X_i}(OL(Y_{\ell_j}, X_{r_i})) \cup LCStr^*(X_i, Y_j)),$$

$$\text{where} \quad LCStr^*(X_i, Y_j) = LCSuf(X_{\ell_i}, Y_{\ell_j}) + LCPref(X_{r_i}, Y_{r_j}).$$

Based on Observation 2, our strategy for computing $LCStr(T, S)$ is to compute $\max(Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j})))$ and $\max(Ext_{Y_j,X_i}(OL(Y_{\ell_j}, X_{r_i})))$ for each pair of X_i and Y_j. Lemma 4 shows how to compute $\max(Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j})))$ and $\max(Ext_{Y_j,X_i}(OL(Y_{\ell_j}, X_{r_i})))$ using FM.

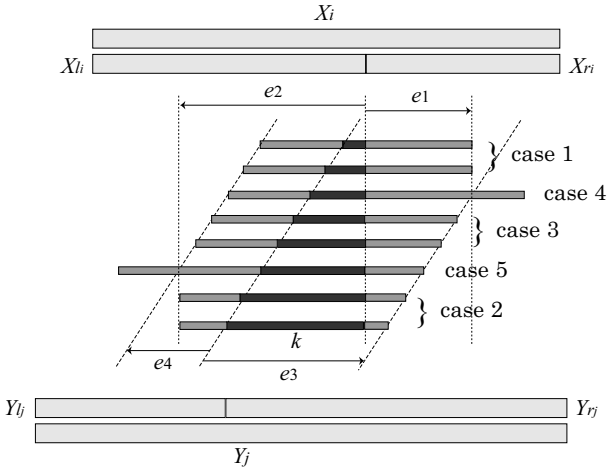


Fig. 2. Illustration for the proof of Lemma 4. The dark rectangles represent the overlaps between X_{ℓ_i} and Y_{r_j}. Case 6 is the special case where cases 4 and 5 happen at the same time and case 3 does not exist.

Lemma 4. *For any variables $X_i = X_{\ell_i}X_{r_i}$ and $Y_j = Y_{\ell_j}Y_{r_j}$, we can compute* $\max(Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j})))$ *and* $\max(Ext_{Y_j,X_i}(OL(Y_{\ell_j}, X_{r_i})))$ *in* $O(n^2 \log n)$ *time.*

Proof. Here we concentrate on computing $\max(Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j})))$, as the case of $\max(Ext_{Y_j,X_i}(OL(Y_{\ell_j}, X_{r_i})))$ is just symmetric. Let $\langle a, d, t\rangle$ be any of the $O(n)$ arithmetic progressions of $OL(X_{\ell_i}, Y_{r_j})$.

Assume that $t > 1$ and $a < d$. The cases where $t = 1$ or $a = d$ are easier to show. Let $u = Y_{r_j}[1 : a]$ and $v = Y_{r_j}[a + 1 : d]$. For any string w, let w^* denote an infinite repetition of w, that is, $w^* = www\cdots$. Firstly we compute

$$e_1 = LCPref(X_{r_i}, (vu)^*) = \begin{cases} FM(Y_{r_j}, X_{r_i}, a+1) & \text{if } FM(Y_{r_j}, X_{r_i}, a+1) < d, \\ FM(X_{r_i}, X_{r_i}, d+1) + d & \text{otherwise,} \end{cases}$$

$$e_2 = LCSuf(X_{\ell_i}, (vu)^*) = FM(X^R_{\ell_i}, X^R_{\ell_i}, d+1) + d,$$

$$e_3 = LCPref(Y_{r_j}, (uv)^*) = FM(Y_{r_j}, Y_{r_j}, d+1) + d,$$

$$e_4 = LCSuf(Y_{\ell_j}, (uv)^*) = \begin{cases} FM(X^R_{\ell_i}, Y^R_{\ell_j}, a+1) & \text{if } FM(X^R_{\ell_i}, Y^R_{\ell_j}, a+1) < d, \\ FM(Y^R_{\ell_j}, Y^R_{\ell_j}, d+1) + d & \text{otherwise.} \end{cases}$$

(See also Fig. 2.) As above, we can compute e_1, e_2, e_3, e_4 by at most 6 calls of FM. Note that $X_i[|X_{\ell_i}| - e_2 + 1 : |X_{\ell_i}| + e_1]$ is the longest substring of X_i that contains $X_i[|X_{\ell_i}| - d + 1 : |X_{\ell_i}|]$ and has a period d. Note also that $Y_j[|Y_{\ell_j}| - e_4 + 1 : |Y_{\ell_j}| + e_3]$ is the longest substring of Y_j that contains $Y_j[|Y_{\ell_j}| + 1 : |Y_{\ell_j}| + d]$ and has a period d.

Let $k \in \langle a, d, t\rangle$. We categorize $Ext_{X_i,Y_j}(k)$ depending on the value of k, as follows.

case 1: When $k < \min\{e_3 - e_1, e_2 - e_4\}$. If $k - d \in \langle a, d, t\rangle$, it is not difficult to see $Ext_{X_i,Y_j}(k) = Ext_{X_i,Y_j}(k - d) + d$. Therefore, we have

$$A = \max\{Ext_{X_i,Y_j}(k) \mid k < \min\{e_3 - e_1, e_2 - e_4\}\} = Ext_{X_i,Y_j}(k'),$$

where $k' = \max\{k \mid k < \min\{e_3 - e_1, e_2 - e_4\}\}$.

case 2: When $k > \max\{e_3 - e_1, e_2 - e_4\}$. If $k + d \in \langle a, d, t\rangle$, it is not difficult to see $Ext_{X_i,Y_j}(k) = Ext_{X_i,Y_j}(k + d) + d$. Therefore, we have

$$B = \max\{Ext_{X_i,Y_j}(k) \mid k > \max\{e_3 - e_1, e_2 - e_4\}\} = Ext_{X_i,Y_j}(k''),$$

where $k'' = \min\{k \mid k > \max\{e_3 - e_1, e_2 - e_4\}\}$.

case 3: When $\min\{e_3 - e_1, e_2 - e_4\} < k < \max\{e_3 - e_1, e_2 - e_4\}$. In this case we have $Ext_{X_i,Y_j}(k) = \min\{e_1 + e_2, e_3 + e_4\}$ for any k with $\min\{e_3 - e_1, e_2 - e_4\} < k < \max\{e_3 - e_1, e_2 - e_4\}$. Thus

$$\begin{aligned} C &= \max\{Ext_{X_i,Y_j}(k) \mid \min\{e_3 - e_1, e_2 - e_4\} < k < \max\{e_3 - e_1, e_2 - e_4\}\} \\ &= \min\{e_1 + e_2, e_3 + e_4\}. \end{aligned}$$

case 4: When $k = e_3 - e_1$. In this case we have

$$\begin{aligned} D = Ext_{X_i,Y_j}(k) &= k + \min\{e_2 - k, e_4\} + LCPref(Y_{r_j}[k + 1 : |Y_{r_j}|], X_{r_i}) \\ &= k + \min\{e_2 - k, e_4\} + FM(Y_{r_j}, X_{r_i}, k + 1). \end{aligned}$$

case 5: When $k = e_2 - e_4$. In this case we have

$$\begin{aligned} E = Ext_{X_i,Y_j}(k) &= k + LCSuf(X_{\ell_i}[1 : |X_{\ell_i}| - k], Y_{\ell_j}) + \min\{e_1, e_3 - k\} \\ &= k + FM(X^R_{\ell_i}, Y^R_{\ell_j}, k + 1) + \min\{e_1, e_3 - k\}. \end{aligned}$$

case 6: When $k = e_3 - e_1 = e_2 - e_4$. In this case we have

$$\begin{aligned} F &= Ext_{X_i,Y_j}(k) \\ &= k + LCSuf(X_{\ell_i}[1 : |X_{\ell_i}| - k], Y_{\ell_j}) + LCPref(Y_{r_j}[k + 1 : |Y_{r_j}|], X_{r_i}) \\ &= k + FM(X^R_{\ell_i}, Y^R_{\ell_j}, k + 1) + FM(Y_{r_j}, X_{r_i}, k + 1). \end{aligned}$$

Then clearly the following inequality stands (see also Fig. 2):

$$F \geq \max\{D, E\} \geq C \geq \max\{A, B\}. \tag{1}$$

A membership query to the arithmetic progression $\langle a, d, t\rangle$ can be answered in constant time. Also, an element $k \in \langle a, d, t\rangle$ such that $\min\{e_3 - e_1, e_2 - e_4\} < k < \max\{e_3 - e_1, e_2 - e_4\}$ of case 3 can be found in constant time, if such exists. k' and k'' of case 1 and case 2, respectively, can be computed in constant time as well. Therefore, based on inequality (1), we can compute $\max(Ext_{X_i,Y_j}(\langle a, d, t\rangle))$ by at most 2 calls of FM, provided that e_1, e_2, e_3, e_4 are already computed.

Since $OL(X_{\ell_i}, Y_{r_j})$ contains $O(n)$ arithmetic progressions by Lemma 2, and each call of FM takes $O(n \log n)$ time by Lemma 3, $\max(Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j})))$ can be computed in $O(n^2 \log n)$ time. □

Now we obtain the main result of this section.

Theorem 2. *Problem 7 can be solved in $O(n^4 \log n)$ time with $O(n^3)$ space.*

Proof. It follows from Theorem 1 that $OL(X_i, Y_j)$ can be computed in $O(n^4 \log n)$ time with $O(n^3)$ space. For any variables $X_i = X_{\ell_i} X_{r_i}$ and $Y_j = Y_{\ell_j} Y_{r_j}$, by Lemmas 2, 3 and 4, $\max(Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j})))$ and $\max(Ext_{Y_j,X_i}(OL(Y_{\ell_j}, X_{r_i})))$ can be computed in $O(n^2 \log n)$ time.

Moreover, it is easy to see that

$$LCSuf(X_{\ell_i}, Y_{\ell_j}) = FM(X^R_{\ell_i}, Y^R_{\ell_j}, 1) \text{ and}$$
$$LCPref(X_{r_i}, Y_{r_j}) = FM(X_{r_i}, Y_{r_j}, 1).$$

Thus $LCStr^*(X_i, Y_j)$ can be computed in $O(n \log n)$ time. Overall, by Observation 2 it takes $O(n^4 \log n)$ time and $O(n^3)$ to solve Problem 1. □

The following corollary is immediate.

Corollary 1. *Given two SLPs $\mathcal{T}$ and $\mathcal{S}$ describing strings T and S respectively, the beginning and ending positions of a longest common substring of T and S can be computed in $O(n^4 \log n)$ time with $O(n^3)$ space.*

4 Computing Palindromes from SLP Compressed Strings

In this section we present an efficient algorithm that computes a succinct representation of all maximal palindromes of string T, when its corresponding SLP $\mathcal{T}$ is given as input. The algorithm runs in $O(n^4)$ time and $O(n^2)$ space, where n is the size of the input SLP $\mathcal{T}$.

For any string T, let $Pals(T)$ denote the set of pairs of the beginning and ending positions of all maximal palindromes in T, namely,

$$Pals(T) = \{(p, q) \mid T[p : q] \text{ is the maximal palindrome centered at } \lfloor \tfrac{p+q}{2} \rfloor\}.$$

Note that the size of $Pals(T)$ is $O(|T|) = O(2^n)$. Thus we introduce a succinct representation of $Pals(T)$ in the next subsection.

4.1 Succinct Representation of $Pals(T)$

Let X_i denote a variable in $\mathcal{T}$ for $1 \leq i \leq n$. For any variables $X_i = X_\ell X_r$, let $Pals^{\triangle}(X_i)$ be the set of pairs of beginning and ending positions of maximal palindromes of X_i that cover or touch the boundary between X_ℓ and X_r, namely,

$$Pals^{\triangle}(X_i) = \{(p, q) \in Pals(X_i) \mid 1 \leq p \leq |X_\ell| + 1, |X_\ell| \leq q \leq |X_i|, p \leq q\}.$$

Also, let $PPals(T)$ and $SPals(T)$ denote the set of pairs of the beginning and ending positions of the prefix and suffix palindromes of T, respectively, that is,

$$PPals(T) = \{(1, q) \in Pals(T) \mid 1 \leq q \leq |T|\}, \text{ and}$$
$$SPals(T) = \{(p, |T|) \in Pals(T) \mid 1 \leq p \leq |T|\}.$$

Gąsieniec et al. [5] claimed the following lemma:

Lemma 5 ([5]). *For any string T, $PPals(T)$ and $SPals(T)$ can be represented by $O(\log |T|)$ arithmetic progressions.*

We have the following observation for decomposition of $Pals(X_i)$.

Observation 3. *For any variables* $X_i = X_\ell X_r$,

$$Pals(X_i) = (Pals(X_\ell) - SPals(X_\ell)) \cup Pals^{\triangle}(X_i) \cup ((Pals(X_r) - PPals(X_r)) \oplus |X_\ell|).$$

Thus, the desired output $Pals(T) = Pals(X_n)$ can be represented as a combination of $\{Pals^{\triangle}(X_i)\}_{i=1}^n$, $\{PPals(X_i)\}_{i=1}^n$ and $\{SPals(X_i)\}_{i=1}^n$. Therefore, computing $Pals(T)$ is reduced to computing $Pals^{\triangle}(X_i)$, $PPals(X_i)$ and $SPals(X_i)$, for every $i = 1, 2, \ldots, n$. The problem to be tackled in this section follows:

Problem 2. Given an SLP $\mathcal{T}$ of size n, compute $\{Pals^{\triangle}(X_i)\}_{i=1}^n$, $\{PPals(X_i)\}_{i=1}^n$ and $\{SPals(X_i)\}_{i=1}^n$.

Lemma 6 is useful to compute $Pals^{\triangle}(X_i)$ from $SPals(X_\ell)$ and $PPals(X_r)$.

Lemma 6. *For any variable* $X_i = X_\ell X_r$ *and any* $(p, q) \in Pals^{\triangle}(X_i)$, *there exists an integer* $l \geq 0$ *such that* $(p+l, q-l) \in SPals(X_\ell) \cup (PPals(X_r) \oplus |X_\ell|) \cup \{(|X_\ell|, |X_\ell|+1)\}$.

Proof. Since $X_i[p:q]$ is a palindrome, $X_i[p+l:q-l]$ is also a palindrome for any $0 \leq l < \lfloor \frac{p+q}{2} \rfloor$. Then we have the following three cases:

1. When $\lfloor \frac{p+q}{2} \rfloor < |X_\ell|$, for $l = p - |X_\ell|$, we have $(p+l, q-l) \in SPals(X_\ell)$.
2. When $\lfloor \frac{p+q}{2} \rfloor > |X_\ell|$, for $l = |X_\ell| - p + 1$, we have $(p+l, q-l) \in PPals(X_r)$.
3. When $\lfloor \frac{p+q}{2} \rfloor = |X_\ell|$, if $q - p + 1$ is odd, then the same arguments to case 1 apply, since $X_\ell[|X_\ell|] = X_\ell[|X_\ell|]^R$ and $(|X_\ell|, |X_\ell|) \in SPals(X_\ell)$. If $q - p + 1$ is even, let $l = |X_\ell| - p$. In this case, we have $p + q = 2|X_\ell| + 1$. Thus, $p + l = |X_\ell|$ and $q - l = |X_\ell| + 1$. □

By Lemma 6, $Pals^{\triangle}(X_i)$ can be computed by "extending" all palindromes in $SPals(X_\ell)$ and $PPals(X_r)$ to the maximal within X_i, and finding the maximal even palindromes centered at $|X_\ell|$ in X_i. In so doing, for any (maximal or non-maximal) palindrome $P = X_i[p:q]$, we define function Ext_{X_i} so that $Ext_{X_i}(p, q) = (p - h, q + h)$, where $h \geq 0$ and $X_i[p-h : q+h]$ is the maximal palindrome centered at position $\lfloor \frac{p+q}{2} \rfloor$ in X_i. For any p, q with $X_i[p:q]$ not being a palindrome, we leave $Ext_{X_i}(p, q)$ undefined. For a set S of pair of integers, let $Ext_{X_i}(S) = \{Ext_{X_i}(p, q) \mid (p, q) \in S\}$.

The next observations give us a recursive procedure to compute $Pals^{\triangle}(X_i)$.

Observation 4. *For any variable* $X_i = X_\ell X_r$,

$$Pals^{\triangle}(X_i) = Ext_{X_i}(SPals(X_\ell)) \cup Ext_{X_i}(PPals(X_r)) \cup Pals^*(X_i), \text{ where}$$

$$Pals^*(X_i) = \{(|X_\ell| - l + 1, |X_\ell| + l) \in Pals(X_i) \mid l \geq 1\}.$$

$PPals(X_i)$ and $SPals(X_i)$ can be computed from $Pals^{\triangle}(X_i)$ as follows:

Observation 5. *For any variable* $X_i = X_\ell X_r$,

$$PPals(X_i) = PPals(X_\ell) \cup \{(1, q) \in Pals^{\triangle}(X_i)\} \text{ and}$$
$$SPals(X_i) = (SPals(X_r) \oplus |X_\ell|) \cup \{(p, |X_i|) \in Pals^{\triangle}(X_i)\}.$$

4.2 Efficient Computation of $Pals^{\triangle}(X_i)$

Let us first briefly recall the work of [10,11]. For any variables $X_i = X_\ell X_r$ and X_j, we define the set $Occ^{\triangle}(X_i, X_j)$ of all occurrences of X_j that cover or touch the boundary between X_ℓ and X_r, namely,

$$Occ^{\triangle}(X_i, X_j) = \{s > 0 \mid X_i[s : s + |X_j| - 1] = X_j, |X_\ell| - |X_j| + 1 \leq s \leq |X_\ell|\}.$$

Theorem 3 ([11]). *For any variables X_i and X_j, $Occ^{\triangle}(X_i, X_j)$ can be computed in total of $O(n^3)$ time and $O(n^2)$ space.*

Lemma 7 ([10]). *For any variables X_i, X_j and integer k, $FM(X_i, X_j, k)$ can be computed in $O(n^2)$ time, provided that $Occ^{\triangle}(X_{i'}, X_{j'})$ is already computed for any $1 \leq i' \leq i$ and $1 \leq j' \leq j$.*

Lemma 8. *For any variable $X_i = X_\ell X_r$ and any arithmetic progression $\langle a, d, t\rangle$ with $(1, \langle a, d, t\rangle) \subseteq PPals(X_r)$, $Ext_{X_i}((1, \langle a, d, t\rangle))$ can be represented by at most 2 arithmetic progressions and a pair of the beginning and ending positions of a maximal palindrome, and can be computed by at most 4 calls of FM. Similar for $Ext_{X_i}((\langle a, d, t\rangle, |X_\ell|))$ with $(\langle a', d', t'\rangle, |X_\ell|) \subseteq SPals(X_\ell)$.*

Proof. By Lemma 3.4 of [13]. □

We are now ready to prove the following lemma:

Lemma 9. *For any variable $X_i = X_\ell X_r$, $Pals^{\triangle}(X_i)$ requires $O(\log |X_i|)$ space and can be computed in $O(n^2 \log |X_i|)$ time.*

Proof. Recall Observation 4. It is clear from the definition that $Pals^*(X_i)$ is either singleton or empty. When it is a singleton, it consists of the maximal even palindrome centered at $|X_\ell|$. Let $l = FM(X_r, X_\ell^R, 1)$. Then we have

$$Pals^*(X_i) = \begin{cases} \emptyset & \text{if } l = 0, \\ \{(|X_\ell| - l + 1, |X_\ell| + l)\} & \text{otherwise.} \end{cases}$$

Due to Lemma 7, $Pals^*(X_i)$ can be computed in $O(n^2)$ time.

Now we consider $Ext_{X_i}(SPals(X_\ell))$. By Lemma 7 and Lemma 8, each subset $Ext_{X_i}((1, \langle a, d, t\rangle)) \subseteq Ext_{X_i}(SPals(X_\ell))$ requires $O(1)$ space and can be computed in $O(n^2)$ time. It follows from Lemma 5 that $Ext_{X_i}(SPals(X_\ell))$ consists of $O(\log |X_i|)$ arithmetic progressions. Thus $Ext_{X_i}(SPals(X_\ell))$ can be computed in $O(n^2 \log |X_i|)$ time. Similar arguments hold for $Ext_{X_i}(PPals(X_r))$. □

4.3 Results

Theorem 4. *Problem 2 can be solved in $O(n^4)$ time with $O(n^2)$ space.*

Proof. Firstly we analyze the time complexity. From Theorem 3 preprocessing for the FM function takes $O(n^3)$ time. By Lemma 7, each call of FM takes $O(n^2)$ time. It follows from Lemma 9 that $Pals^{\triangle}(X_i)$ can be computed in $O(n^3)$

time. By Observation 5, $PPals(X_i)$ and $SPals(X_i)$ can be computed in $O(n)$ time from $Pals^{\triangle}(X_i)$. Hence the overall time cost to compute $\{PPals(X_i)\}_{i=1}^{n}$, $\{SPals(X_i)\}_{i=1}^{n}$, and $\{Pals^{\triangle}(X_i)_{i=1}^{n}\}$ is $O(n^4)$.

Secondly we analyze the space complexity. The preprocessing for the *FM* function requires $O(n^2)$ due to Theorem 3. From Lemma 5 $PPals(X_i)$ and $SPals(X_i)$ require $O(n)$ space. Lemma 9 states that $Pals^{\triangle}(X_i)$ requires $O(n)$ space. Thus the total space requirement is $O(n^2)$. □

The following two theorems are results obtained by slightly modifying the algorithm of the previous subsections.

Theorem 5. *Given an SLP $\mathcal{T}$ that describes string T, whether T is a palindrome or not can be determined with extra $O(1)$ space and without increasing asymptotic time complexities of the algorithm.*

Proof. It suffices to see if $(1, |T|) \in PPals(T) = PPals(X_n)$. By Lemma 5, $PPals(X_n)$ can be represented by $O(n)$ arithmetic progressions. It is not difficult to see that T is a palindrome if and only if $a + (t-1)d = |T|$ for the arithmetic progression $\langle a, d, t\rangle$ of the largest common difference among those in $PPals(X_n)$. Such an arithmetic progression can easily be found during computation of $PPals(X_n)$ without increasing asymptotic time complexities of the algorithm. □

Theorem 6. *Given an SLP $\mathcal{T}$ that describes string T, the position pair (p, q) of the longest palindrome in T can be found with extra $O(1)$ space and without increasing asymptotic time complexities of the algorithm.*

Proof. We compute the beginning and ending positions of the longest palindrome in $Pals^{\triangle}(X_i)$ for $i = 1, 2, \ldots, n$. It takes $O(n)$ time for each X_i. If its length exceeds the length of the currently kept palindrome, we update the beginning and ending positions. □

Provided that $\{PPals(X_i)\}_{i=1}^{n}$, $\{SPals(X_i)\}_{i=1}^{n}$, and $\{Pals^{\triangle}(X_i)_{i=1}^{n}\}$ are already computed, we have the following result:

Theorem 7. *Given pair (p, q) of integers, it can be answered in $O(n)$ time whether or not substring $T[p:q]$ is a maximal palindrome of T.*

Proof. We binary search the derivation tree of SLP $\mathcal{T}$ until finding the variable $X_i = X_\ell X_r$ such that $1 + \mathit{offset} \le p \le |X_\ell| + \mathit{offset}$ and $1 + \mathit{offset} + |X_\ell| \le q \le |X_i| + \mathit{offset}$. This takes $O(n)$ time. Due to Observation 4, for each variable X_i, $Pals^{\triangle}(X_i)$ can be represented by $O(n)$ arithmetic progressions plus a pair of the beginning and ending positions of a maximal palindrome. Thus, we can check if $(p, q) \in Pals^{\triangle}(X_i)$ in $O(n)$ time. □

Finally we supply pseudo-codes of our algorithms.

Algorithm *ComputeLCStr*
Input: SLP $\mathcal{T} = \{X_i\}_{i=1}^{n}$, $\mathcal{S} = \{Y_j\}_{j=1}^{m}$.
$L = \emptyset$;
for $i = 1 \ldots n$ **do**
  for $j = 1 \ldots m$ **do**
    compute $OL(X_i, Y_j)$ and $OL(Y_j, X_i)$;

Algorithm *ComputePalindromes*
Input: SLP $\mathcal{T} = \{X_i\}_{i=1}^{n}$.
for $i = 1 \ldots n$ **do**
  $SPals(X_i) = \emptyset$; $PPals(X_i) = \emptyset$;
  $Pals^{\triangle}(X_i) = \emptyset$;

for $i = 1 \ldots n$ **do**
 for $j = 1 \ldots m$ **do**
  if $X_i = a$ **then** /* $a \in \Sigma$ */
   $L = L \cup LCSuf(Y_{j_\ell}, X_i)$
    $\cup LCPref(Y_{j_r}, X_i)$;
  else if $Y_j = a$ **then** /* $a \in \Sigma$ */
   $L = L \cup LCSuf(X_{i_\ell}, Y_j)$
    $\cup LCPref(X_{i_r}, Y_j)$;
  else /* $X_i = X_\ell X_r$ and $Y_i = Y_\ell Y_r$ */
   $L = L \cup LCStr^*(X_i, Y_j)$
    $\cup Ext_{X_i,Y_j}(OL(X_{\ell_i}, Y_{r_j}))$
    $\cup Ext_{Y_j,X_i}(OL(Y_{\ell_j}, X_{r_i}))$;
return $\max(L)$;

for $i = 1 \ldots n$ **do**
 if $X_i = a$ **then**
  $SPals(X_i) = (1, 1)$; $PPals(X_i) = (1, 1)$;
  $Pals^{\triangle}(X_i) = (1, 1)$;
 else /* $X_i = X_\ell X_r$ */
  $seed = SPals(X_\ell) \cup (PPals(X_r) \oplus |X_\ell|)$;
  $Pals^{\triangle}(X_i) = Ext_{X_i}(seed) \cup Pals^*(X_i)$;
  $SPals(X_i) = SPals(X_r)$
    $\cup \{(1, q) \in Pals^{\triangle}(X_i)\}$;
  $PPals(X_i) = PPals(X_\ell)$
    $\cup \{(p, |X_\ell|) \in Pals^{\triangle}(X_i)\}$;
return $Pals^{\triangle}(X_1), \ldots, Pals^{\triangle}(X_n)$;

References

1. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Trans. Info. Theory IT-23(3), 337–349 (1977)
2. Ziv, J., Lempel, A.: Compression of individual sequences via variable-length coding. IEEE Trans. Info. Theory 24(5), 530–536 (1978)
3. Kieffer, J., Yang, E., Nelson, G., Cosman, P.: Universal lossless compression via multilevel pattern matching. IEEE Trans. Info. Theory 46(4), 1227–1245 (2000)
4. Nevill-Manning, C.G., Witten, I.H., Maulsby, D.L.: Compression by induction of hierarchical grammars. In: DCC 1994, pp. 244–253. IEEE Press, Los Alamitos (1994)
5. Gasieniec, L., Karpinski, M., Plandowski, W., Rytter, W.: Efficient algorithms for Lempel-Ziv encoding. In: Karlsson, R., Lingas, A. (eds.) SWAT 1996. LNCS, vol. 1097, pp. 392–403. Springer, Heidelberg (1996)
6. Rytter, W.: Grammar compression, lz-encodings, and string algorithms with implicit input. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 15–27. Springer, Heidelberg (2004)
7. Inenaga, S., Shinohara, A., Takeda, M.: An efficient pattern matching algorithm on a subclass of context free grammars. In: Calude, C.S., Calude, E., Dinneen, M.J. (eds.) DLT 2004. LNCS, vol. 3340, pp. 225–236. Springer, Heidelberg (2004)
8. Plandowski, W.: Testing equivalence of morphisms on context-free languages. In: van Leeuwen, J. (ed.) ESA 1994. LNCS, vol. 855, pp. 460–470. Springer, Heidelberg (1994)
9. Karpinski, M., Rytter, W., Shinohara, A.: An efficient pattern-matching algorithm for strings with short descriptions. Nordic Journal of Computing 4, 172–186 (1997)
10. Miyazaki, M., Shinohara, A., Takeda, M.: An improved pattern matching algorithm for strings in terms of straight-line programs. In: Hein, J., Apostolico, A. (eds.) CPM 1997. LNCS, vol. 1264, pp. 1–11. Springer, Heidelberg (1997)
11. Lifshits, Y.: Processing compressed texts: A tractability border. In: CPM 2007. LNCS, vol. 4580, pp. 228–240. Springer, Heidelberg (2007)
12. Lifshits, Y., Lohrey, M.: Querying and embedding compressed texts. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 681–692. Springer, Heidelberg (2006)
13. Apostolico, A., Breslauer, D., Galil, Z.: Parallel detection of all palindromes in a string. Theoretical Computer Science 141, 163–173 (1995)

Basic Sets in the Digital Plane

Neža Mramor-Kosta[1,⋆] and Eva Trenklerová[2,⋆⋆]

[1] University of Ljubljana, Faculty of Computer and Information Science and Institute of Mathematics, Physics and Mechanics
Jadranska 19, 1000 Ljubljana, Slovenia
neza.mramor@fmf.uni-lj.si
[2] Institute of Computer Science
Faculty of Science
P.J.Šafárik University
Jesenná 5, 04001 Košice, Slovakia
eva.trenklerova@upjs.sk

Abstract. A set K in the plane $\mathbb{R}^2$ is basic if each continuous function $f: K \to \mathbb{R}$ can be expressed as a sum $f(x,y) = g(x) + h(y)$ with $g, h: \mathbb{R} \to \mathbb{R}$ continuous functions. Analogously we define a digital set K_k in the digital plane to be basic if for each digital function $f: K_k \to \mathbb{R}$ there exist digital functions $g, h: \mathbb{I}_k \to \mathbb{R}$ on the digital unit interval $\mathbb{I}_k$ such that $f(x,y) = g(x) + h(y)$ for each pixel $(x,y) \in K_k$. Basic subsets of the plane were characterized by Sternfeld and Skopenkov. In this paper we prove a digital analogy of this result. Moreover we explore the properties of digital basic sets, and their possible use in image analysis.

Keywords: Basic Sets in the Plane, Digital Topology.

1 Introduction

A set $K \subset \mathbb{R}^n$ is a *basic subset* of $\mathbb{R}^n$ if for each continuous function $f: K \to \mathbb{R}$ there exist continuous functions $g_1, \ldots, g_n: \mathbb{R} \to \mathbb{R}$ such that $f(x_1, \ldots, x_n) = g_1(x_1) + \ldots + g_n(x_n)$ for each point $(x_1, \ldots, x_n) \in K$.

Basic sets are connected to the thirteenth problem from Hilbert's list of 23 open mathematical problems formulated in his famous lecture at the International Congress of Mathematics in Paris in 1900. The problem contained the conjecture that not all continuous functions of three variables are representable as superpositions of continuous functions of two variables. This conjecture can be reformulated in terms of basic subsets of dimension 3. It was refuted in a series of papers by Arnold [1,2] and Kolmogorov [3,4]. Results of Kolmogorov [4], Ostrand [5] and Sternfeld [9] show that compact sets which can be embedded as basic subsets of the unit cube $\mathbb{I}^n$ for $n > 2$ are characterized by dimension: for a compact set K there exists an embedding (a continuous injective map with continuous inverse) $\varphi: K \to \mathbb{I}^n$ such that $\varphi(K) \subset \mathbb{I}^n$ is a basic subset if and

⋆ Partially funded by the Research Agency of Slovenia, grant no. P1-0292.
⋆⋆ The author was supported by grants VEGA 1/3002/06 and VEGA 1/3128/06.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 376–387, 2008.

only if the dimension of K is not greater than $(n-1)/2$, where $n > 2$. For $n = 2$, Sternfeld [10] also gave a characterization of compact basic subsets K of $\mathbb{I}^2$, which can be expressed in terms of sequences of a special type called *arrays*.

Definition 1. *An* array *is a sequence of points* $\{(x_i, y_i) \mid i \in I\}$, *where* $I = \{1, 2, \ldots, m\}$ *for some* $m \in \mathbb{N}$ *such that either* $x_{2i-1} = x_{2i}$ *and* $y_{2i} = y_{2i+1}$ *for all triples of indeces* $2i-1, 2i, 2i+1 \in I$ *or* $y_{2i-1} = y_{2i}$ *and* $x_{2i} = x_{2i+1}$ *for all* $2i-1, 2i, 2i+1 \in I$ *and no two consecutive points are equal. If* $I = \{1, 2, \ldots, m\}$ *then the* length *of the array is* m. *We say that the array is* closed, *if* $(x_1, y_1) = (x_m, y_m)$.

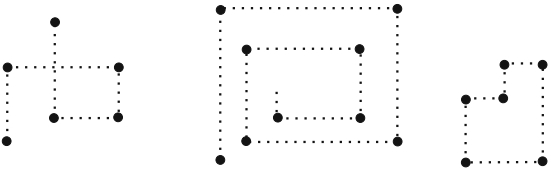


Fig. 1. Arrays in $\mathbb{I}^2$

According to Sternfeld's characterization and its reformulation by Skopenkov [8], a compact subset K of $\mathbb{I}^2$ is basic if and only if there exists a constant $M \in \mathbb{N}$ such that the lengths of arrays in K are bounded by M. Although Sternfeld's characterization is formulated in elementary terms, his proof is not constructive, and uses arguments from functional analysis. This work was motivated by an attempt to find a direct constructive proof of Sternfeld's theorem which is described in [11].

The approach used there leads naturally to a digital formulation of basic sets which we describe in this contribution.

In the digital setting, we consider the digital unit square $\mathbb{I}_k^2$, $k = 1, 2, \ldots$, i.e. the unit square subdivided into $k \times k$ cells (pixels)

$$\mathbb{I}_k^2 = \{e_{ij} \mid i = 1, \ldots, k, j = 1, \ldots, k\} ,$$

where each cell e_{ij} is a product of half-open intervals

$$e_{ij} = a_i \times b_j = [(i-1)/k, i/k) \times [(j-1)/k, j/k) .$$

The digital square can be thought of as the product $\mathbb{I}_k^2 = \mathbb{I}_k \times \mathbb{I}_k$, where $\mathbb{I}_k$ denotes the digital unit interval

$$\mathbb{I}_k = \{[(i-1)/k, i/k) \mid i = 1, \ldots, k\} .$$

A digital set $K_k \subset \mathbb{I}_k^2$ is a collections of cells e_{ij}, and a digital function $f_k \colon K_k \to \mathbb{R}$ associates a real number to each cell in K_k. If we identify a digital set $K_k \subset \mathbb{I}_k^2$ with the union of its cells, then a digital function on K_k can be thought of as a piecewise constant function with constant value on each cell $e_{ij} \in K_k$.

As in the classical case, we define:

Definition 2. *A digital set* $K_k \subset \mathbb{I}_k^2$ *is* basic *if for each function* $f_k : K_k \to \mathbb{R}$ *there exist functions* $g_k, h_k : \mathbb{I}_k \to \mathbb{R}$ *such that* $f_k(e_{ij}) = g_k(a_i) + h_k(b_j)$ *for each cell* $e_{ij} = a_i \times b_j \in K_k$.

In Section 2 we prove a digital analogue of Sternfeld's theorem, stating that a digital set $K_k \subset \mathbb{I}_k^2$ is basic precisely when it contains no closed arrays (Theorem 1). We also prove an upper and lower bound on the norms of the functions g_k, h_k in the decomposition. We show that this condition does not suffice to obtain digitally continuous decompositions in the sense of [7]: a digitally 8-continuous function f_k on $K_k \subset \mathbb{I}_k^2$ can not be decomposed as a sum $f_k = g_k + h_k$ of digitally continuous functions g_k, h_k on the digital interval.

In Section 3 we describe an application of digital basic sets to image analysis. A gray-scale image can be thought of as a digital function on the digital square [6], and a decomposition of the foreground of the picture as a union of digital basic subsets enables, in some cases, a compressed representation of the image.

In the last section, Section 4, the following result of [11] is reformulated in the digital setting. Let $K \subset \mathbb{I}^2$ be a compact set which contains no arrays of length more than two, and $f : K \to \mathbb{R}$ a continuous function. For each $\varepsilon > 0$ there exists a k, a digital subset $K_k \subset \mathbb{I}_k^2$ approximating the set K, and continuous functions $g_k, h_k : \mathbb{I} \to \mathbb{R}$ which are linear on all segments $[(i-1)/k, i/k] \subset [0,1]$, $i = 1, \ldots, k$ such that

$$|f(x,y) - g_k(x) - h_k(y)| < \varepsilon.$$

In addition, the norms of the function g_k and h_k are bounded by twice the norm of f, which implies the convergence of the sequences $\{g_k\}, \{h_k\}$ to continuous functions on $\mathbb{I}$.

2 Digital Basic Sets

Let $p_k, q_k : \mathbb{I}_k^2 \to \mathbb{I}_k$ denote the vertical and the horizontal projections respectively, i.e. $p_k(e_{ij}) = a_i$, $q_k(e_{ij}) = b_j$.

Definition 3. *A* (digital) array *of length* m *in* $\mathbb{I}_k^2$ *is a sequence of cells* $\{e^1, e^2, \ldots, e^m\}$, $e^n = a_{i_n} \times b_{j_n}$, $n = 1, \ldots, m$, *such that both* $a_{i_{2j-1}} = a_{i_{2j}}$ *and* $b_{i_{2j}} = b_{i_{2j+1}}$ *for all* j *or both* $b_{i_{2j-1}} = b_{i_{2j}}$ *and* $a_{i_{2j}} = a_{i_{2j+1}}$ *for all* j, *with no two consecutive cells equal.*

Let us prove a digital analogue of Sternfeld's characterization of compact basic sets in the plane stating that a digital set $K_k \subset \mathbb{I}_k^2$ is basic if and only if it contains no arrays of arbitrary length.

Since a digital set is a finite collection of cells this is equivalent to proving that K_k is basic if and only if it contains no closed array, i.e. no array $\{e^1, e^2, \ldots, e^m\}$ with $e^1 = e^m$.

Theorem 1. *A digital set* $K_k \subset \mathbb{I}_k^2$ *is a basic subset of* $\mathbb{I}_k^2$ *if and only if it contains no closed array.*

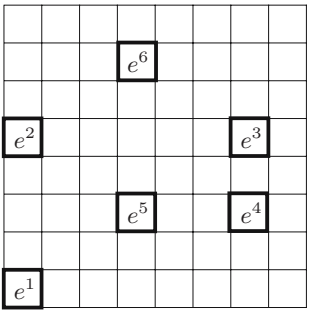


Fig. 2. Digital array

Proof. Let K_k contain a closed array $\{e^1, \ldots, e^m\}$, $e^1 = e^m$. The length m must be odd. Let for example $a_{i_1} = a_{i_2}$, $b_{i_2} = b_{i_3}$, $a_{i_3} = a_{i_4}$, …, $b_{i_{m-1}} = b_{i_m} = b_{i_1}$. Let $f_k : K_k \to \mathbb{R}$ be such that $f_k(e^{2n-1}) = 1$ and $f_k(e^{2n}) = -1$ for all n. Let us assume that f_k is decomposed as $g_k + h_k$. Then

$$\begin{aligned} 1 &= g(a_{i_1}) + h(b_{i_1}) \\ -1 &= g(a_{i_1}) + h(b_{i_2}) \\ 1 &= g(a_{i_3}) + h(b_{i_2}) \\ &\vdots \\ 1 &= g(a_{i_{m-2}}) + h(b_{i_{m-3}}) \\ -1 &= g(a_{i_{m-2}}) + h(b_{i_1}) \end{aligned}$$

By adding up with changing sign we get $m - 1 = 0$. Contradiction. So K_k is not basic.

Let us show the opposite implication. We will call a set $A \subset K_k$ such that for each pair of cells $e \neq e'$ from A there exists an array in A with end cells e, e' an *array structure*. An array structure is *maximal* if it is not contained in any larger array structure in K_k.

Let f_k be a digital function on K_k and let K_k be decomposed as $\bigcup_{i \in I} A^i$, where each A^i is a maximal array structure. Evidently the decomposition is unique and $p_k(A^i) \cap p_k(A^j) = \emptyset$ and $q_k(A^i) \cap q_k(A^j) = \emptyset$ for any two different i, j. So it suffices to find a decomposition

$$f_k(e_{ij}) = g_k(p_k(e_{ij})) + h_k(q_k(e_{ij}))$$

on each maximal array structure separately.

Let A be a maximal array structure in K_k, and $e = a \times b \in A$ any cell. Let $g_k(a)$ be arbitrary, and $h_k(b) = f_k(e) - g_k(a)$. Since any cell in A is connected to e by an array, the values of g_k, h_k on all other cells in $p_k(A)$ and $q_k(A)$ are then determined – they are defined inductively, in the following way.

0. Let $D^0 = \{e\}$ be a single cell.
1. After an even number of steps of the construction, suppose that g_k, h_k are defined for all cells $e_{ij} = a_i \times b_j$ on a subset D^{2k} of A, and let D^{2k+1} be obtained by adding to D^{2k} all cells $e_{i'j'} \in A \setminus D^{2k}$ which form a horizontal pair with some cell $e_{ij'} \in D^{2k}$. For every such cell, h_k is defined but g_k is not defined yet, since this would imply the existence of a closed array in A. On every such cell we define $g_k(a_{i'}) = f(e_{i'j'}) - h_k(b_{j'})$.
2. After an odd number of steps of the construction, g_k and h_k are defined on a set D^{2k+1}, and D^{2k+2} is obtained by adding to D^{2k+1} all cells $e_{i'j'}$ of $A \setminus D^{2k+1}$ which form a vertical pair with some cell of D^{2k+1}. For every such cell, g_k is defined, while h_k is not defined yet, and we set $h_k(b_{j'}) = f(e_{i'j'}) - g_k(a_{i'})$.
3. Since A is a finite collection of cells, $D^l = A$ for some l.

Remark 1. Since a pixel in the digital plane can be represented by its center, a digital sets can be represented as a finite set, which is compact. Theorem 1 thus follows directly from the continuous case. But, unlike the proof of the continuous case [10], this proof is constructive.

Theorem 1 shows that in order to store the values of a function f_k on a basic digital set K_k in $\mathbb{I}_k^2$, it suffices to store the values of g_k and h_k on the projections $p_k(K_k)$ and $q_k(K_k)$. For example, the gray-scale function f_k of a digital image representing an object whose form is a basic digital set K_k can be reconstructed from the values of g_k and h_k. The values of g_k and h_k are not restricted to the range of f_k any more, though. The following proposition gives bounds for the norms of g_k and h_k, which ensure that the computation required to reconstruct f_k from g_k and h_k does not involve very large numbers.

Proposition 1. *Let $K_k \subset \mathbb{I}_k^2$ be a basic digital set, and let M be the maximal length of an array in K_k. For any digital function $f_k : K_k \to \mathbb{R}$ there exists a decomposition $f_k(e_{ij}) = g_k(a_i) + h_k(b_j)$ such that*

$$\max\{\|g_k\|, \|h_k\|\} \leq M\|f_k\|.$$

Proof. Let D^i be as in the proof of Theorem 1. On $D^0 = \{e = a \times b\}$ define $g_k(a) = 0$ and $h_k(b) = f_k(e)$. Then $\|g_k\| = 0$ and $\|h_k\| = \|f_k\|$ on D^0. At each even step of the construction the norm of $h_k|_{D^{2k}}$ is bounded by the sum of the norm of $g_k|_{D^{2k-1}}$ and the norm of f_k. Similarly, at each odd step of the construction the norm of $g_k|_{D^{2k+1}}$ is bounded by the sum of the norm of $h_k|_{D^{2k}}$ and the norm of f_k. Since the number of steps in the construction is at most $M - 1$, we obtain the required bound.

There is also a lower bound on the norms.

Proposition 2. *For each M there exists a set K_k which contains arrays of length at most M and a function f_k on K_k such that for each decomposition $f_k(e_{ij}) = g_k(a_i) + h_k(b_j)$ we have*

$$\min\{\|g_k\|, \|h_k\|\} \geq (M - 2)\|f_k\|.$$

Proof. Let K_k be the set of cells $\{e_{11}, e_{31}, e_{33}, \ldots, e_{MM}\}$ for an odd M and $\{e_{11}, e_{31}, e_{33}, \ldots, e_{(M+1)(M-1)}\}$ for an even M (see Figure 3). Define f_k as $f_k(e_{(2i-1)(2i-1)}) = 1$ and $f_k(e_{(2i+1)(2i-1)}) = -1$, then $\|f_k\| = 1$. For any value of $g_k(a_1)$ we have $h_k(b_1) = 1 - g_k(a_1)$, $g_k(a_3) = -g_k(a_1) - 2$, $h_k(b_3) = 3 + g_k(a_1)$, etc. and the result can easily be seen.

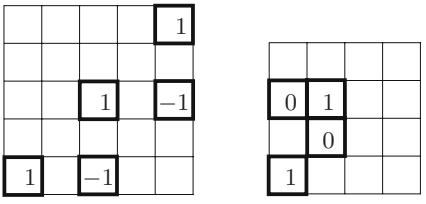


Fig. 3. Proposition 2 for $M = 2$ *(left)* and Proposition 3 *(right)*

Let us consider continuity. A function $f_k : K_k \to \mathbb{Z}$ is said to be 8-*continuous* (see [7]) if $|f_k(e_{ij}) - f_k(e_{i'j'})| \leq 1$ for each cell $e_{ij} \in K_k^2$ and for all cells $e_{i'j'} \in K_k$ in the 8-neighborhood of e_{ij}, i.e. $\max\{|i - i'|, |j - j'|\} = 1$. A function $g_k : \mathbb{I}_k \to \mathbb{Z}$ is said to be *continuous* if $|g_k(a_i) - g_k(a_{i'})| \leq 1$ for each two cells $a_i, a_{i'}$ such that $|i - i'| \leq 1$.

Proposition 3. *There exists a set $K_k \subset \mathbb{I}_k$ which contains no closed array and an 8-continuous function $f_k : K_k \to \mathbb{Z}$ such that there does not exist a decomposition $f_k(e_{ij}) = g_k(a_i) + h_k(b_j)$, $g_k, h_k : \mathbb{I}_k \to \mathbb{Z}$, with both functions g_k, h_k continuous.*

Proof. Let $K = \{e_{11}, e_{13}, e_{22}, e_{23}\}$. Let $f_k(e_{11}) = 1, f_k(e_{13}) = 0, f_k(e_{22}) = 0, f_k(e_{23}) = 1$ (see Figure 3). Let $g_k(a_1)$ be arbitrary. Then

$$
\begin{aligned}
h_k(b_1) &= f_k(e_{11}) - g_k(a_1) = 1 - g_k(a_1) \\
h_k(b_3) &= f_k(e_{13}) - g_k(a_1) = 0 - g_k(a_1) \\
g_k(a_2) &= f_k(e_{23}) - h_k(b_3) = 1 - h_k(b_3) = 1 + g_k(a_1) \\
h_k(b_2) &= f_k(e_{22}) - g_k(a_2) = 0 - g_k(a_2) = -1 - g_k(a_1).
\end{aligned}
$$

Now in the adjacent cells b_1, b_2 we have $|h_k(b_1) - h_k(b_2)| = 2$.

In the discrete case, the number of subsets $K_k \subset \mathbb{I}_k^2$ is finite and the question is how many sets among all nonempty subsets $K_k \subset \mathbb{I}_k^2$ are basic. As noted in the beginning of this section this number is equal to the number of all sets containing no closed arrays, and can be computed by counting. For example, for $k = 3$, it is equal to 328. Figure 4 shows, up to symmetry, all the maximal basic subsets (i.e. those not contained in any bigger basic set) of $\mathbb{I}_3^2$.

For small values of k, we have computed the number of basic sets using an algorithm which searches through the space of all possible sets $\mathbb{I}_3^2$. The idea is to try to add one cell at a time – if the cell does not close any array already in the

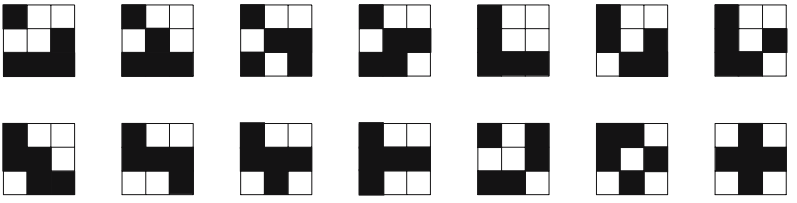


Fig. 4. Basic subsets of $\mathbb{I}_3^2$

Table 1. The number and the mean size of the maximal basic sets in $\mathbb{I}_k^2$

k	No. of non-empty basic sets	Size of maximal basic sets
2	11	3
3	328	5
4	16 145	7
5	1 475 856	9

set we add the cell and continue searching, in any case we continue searching without the cell. The algorithm has exponential complexity. Table 1 gives the numbers of the non-empty basic sets. The maximal basic sets have the same size for a given k.

Problem 1. Find a closed form expression for the number of basic subsets of $\mathbb{I}_k^2$ for an arbitrary k.

The problem of determining whether a given set $K_k \subset \mathbb{I}_k^2$ is basic is much simpler, and can be solved in polynomial time. It is equivalent to finding cycles in a graph of a special type. The algorithm goes through all cells of K_k and constructs the set of arrays.

```
array A := ∅
set of arrays SA := {some cell c'}
for all cells c from K_k not in SA
   for all arrays B from SA
      if c has a horizontal or vertical neighbor in B
         if c has both a horizontal and vertical neighbor in B
            "We have a closed array"
            exit
         else
            if A=∅
               add c to B
               A:=B
            else
               delete B from SA and connect B to A
      if c was not added anywhere
         add {c} as a new array to SA
```

3 An Application

In this section we describe an application of digital basic sets to image analysis.

An example of a digital function on the digital square is the gray-scale function of a digital image of resolution $k \times k$. A digital image representing an object on a uniform background is given by a digital function f_k on $\mathbb{I}_k^2$. The support K_k of f_k is in general not a basic set, but it can be decomposed into a union of basic sets. Our idea is that, if this decomposition is done carefully, only a small number of these basic subsets suffices for an approximate reconstruction of the digital image.

In Figure 5, the left picture represents a gray-scale image of a tree on a white background. The resolution of the image is 249×249.

Fig. 5. A gray-scale image consisting of 249×249 pixels

In order to decompose the foreground, i.e. the union of cells which are not white, into basic sets we have used a straightforward algorithm, which produced a decomposition consisting of 61 basic sets. The right picture in Figure 5 shows the reconstruction obtained from the first 20 sets in the decomposition.

The total number of nonzero values (non-white cells) in this example is 16157, while the total number of nonzero values of the functions g_{249} and h_{249} on all the sets in the decomposition is $8163 + 7968 = 16131$, so the complete reconstruction would require approximately as many stored values as the original image. An approximate reconstruction represents a substantial reduction in the required number of stored values, though. Figure 6 shows two reconstruction obtained from every 2nd and every 10th basic set, respectively.

In order to use basic decomposition for image compression one would need an efficient algorithm for decomposing a given set into basic sets. A part of this problem is finding the smallest number of necessary basic sets, and an upper bound for this is the number of basic sets necessary to cover the whole square $\mathbb{I}_k^2$. Clearly, two basic sets are necessary to cover the square $\mathbb{I}_2^2$. In general k basic sets suffice to cover the square $\mathbb{I}_k^2$, but Figure 4 shows that the square $\mathbb{I}_3^2$ can be covered by 2 basic sets (for example, by the first maximal basic set and its complement).

Problem 2. What is smallest number of basic sets necessary to cover the square $\mathbb{I}_k^2$?

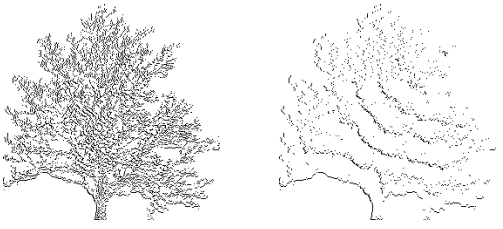


Fig. 6. Reconstructions on the basis of basic sets

In our example the basic sets included in the reconstructions where chosen arbitrarily. A good criterion for choosing the basic sets included would certainly improve the reconstruction. One possibility would be to use the size of the sets as a criterion, that is, include only the basic sets which are big enough.

4 Digital Approximations of Basic Sets in $\mathbb{I}^2$

Definition 4. *The* digital approximation *at resolution k of a set $K \subset \mathbb{I}^2$ is the digital set $K_k \subset \mathbb{I}^k$ consisting of cells e such that $e \cap K \neq \emptyset$.*

The digital approximation of a compact set K in the plane which is basic, i.e. contains only arrays of limited length, might not be a basic digital set. There exist basic sets $K \subset \mathbb{R}^2$ such that the digital approximation at every resolution contains a closed array. An example of such a set is shown on Figure 7. The set consists of a single arc and three additional points, which form an array. There also exist examples of compact sets in the plane containing only arrays of length two, such that each digital approximation contains a closed array (as they are rather complicated we do not give them here).

However, it is possible to construct an approximate decomposition $f_k = g_k + h_k$ of a function $f_k : K_k \to \mathbb{R}$, although the set K_k may contain a closed

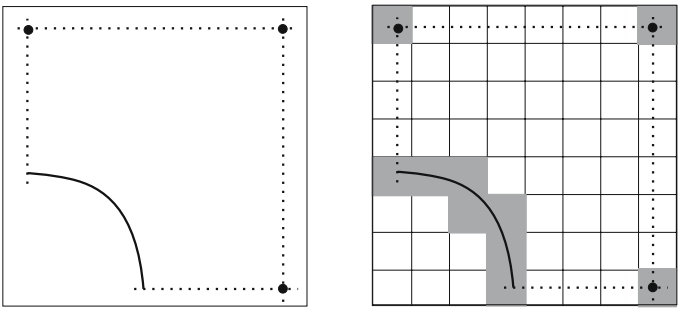


Fig. 7. A set and its digital approximation

digital array. In addition, if K_k is the digital approximation of a basic set K and f_k a digital function approximating a continuous function f on K then, for k big enough, it is possible to find an approximate decomposition $f_k = g_k + h_k$ with digitally continuous functions g_k and h_k. This idea was used in [11] in the construction of a decomposition $f(x,y) = g(x) + h(y)$ of a function f on a set K containing no arrays of length more than two. Such a construction is nontrivial even in this simplest case. We give a digital formulation of the main steps of this construction.

Definition 5. *The* digital approximation *at resolution k of a function $f : K \to \mathbb{R}$ is the digital function $f_k : K_k \to \mathbb{R}$ on the digital approximation K_k of K defined as*

$$f_k(e) = \inf\{f(x,y) \mid (x,y) \in e \cap K\} .$$

Since the digital approximation f_k of a continuous function has values in $\mathbb{R}$, and not in $\mathbb{Z}$, we will use the following natural modification of the definition of digital continuity from Section 2:

Definition 6. *A digital function $g_k : \mathbb{I}_k \to \mathbb{R}$ is said to be* ε-continuous *if $|g_k(a_i) - g_k(a_{i'})| \leq \varepsilon$ for each two cells $a_i, a_{i'}$ such that $|i - i'| \leq 1$.*

The following theorem is a digital reformulation the main result of [11].

Theorem 2. *Let $K \subset \mathbb{I}^2$ be a compact set which contains no arrays of length more than $m = 2$, and $f : K \to \mathbb{R}$ a continuous function. For each $\varepsilon > 0$ there exists a $k \in \mathbb{N}$ such that for the digital set $K_k \subset \mathbb{I}_n^2$ approximating K and for the digital function $f_k : K_k \to \mathbb{R}$ approximating f there exist functions $g_k, h_k : \mathbb{I}_k^2 \to \mathbb{R}$ such that*

$$|f_k(a_i, b_j) - g_k(a_i) - h_k(b_j)| \leq \varepsilon$$

for each cell (a_i, b_j) from K_k. Moreover the functions g_k and h_k are ε-digitally continuous and their norms are bounded by $2||f||$.

The set K_k in Theorem 2 is not necessarily basic in the digital sense. But it has a different property which somehow mimics the fact that K does not contain arrays of length more than 2.

We say that two cells $e_{ij}, e_{i'j'} \in \mathbb{I}^2$ form an *almost vertical* or *almost horizontal pair*, if $|i - i'| \leq 1$ or $|j - j'| \leq 1$, respectively. An *extended arc* is a sequence of cells $\{e^1, \ldots, e^m\}$ in $\mathbb{I}_k^2$ such that each two cells e^i and e^{i+1} form an almost horizontal or vertical pair, see Figure 8. The *length* of this extended arc is m.

Definition 7. *A set $K_k \in \mathbb{I}_k^2$ is called* (α, l)-faithful *if it has the following property: for each almost vertical pair $e^1, e^2 \in K_k$ and each almost horizontal pair $e^3, e^4 \in K_k$ with $|e^1 - e^2| > \alpha$, $|e^3 - e^4| > \alpha$, the length of each extended arc in K_k connecting e^2 to e^3 is at least l.*

The following theorem is a digital version of [11][Lemma 3.5].

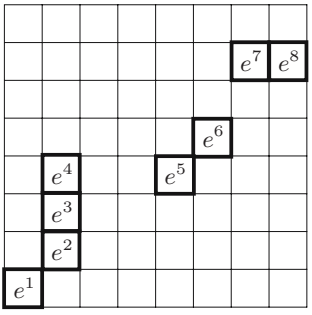


Fig. 8. An extended arc

Theorem 3. *Let a compact set $K \subset \mathbb{I}^2$ contain arrays of length at most two. For each $\alpha \in \mathbb{R}$, $\alpha > 0$ and for each $l \in \mathbb{N}$ there exists $k \in \mathbb{N}$ such that the digital set K_k approximating K is (α, l)-faithful.*

The proof of the next theorem is a digital reformulation of the proof of [11][Theorem 3.4] with different constants.

Theorem 4. *Let K_k be a digital set which is (α, l)-faithful. Let $f_k : K_k \to \mathbb{R}$ be a digital function. Let $\beta = \beta(\alpha, f_k)$ be such that $|f_k(e) - f_k(e')| < \beta$ for all cells e, e' with $|e - e'| < \alpha$. Let $\eta = \max\{\beta, ||f||/l\}$. Then there exist digital functions $g_k, h_k : \mathbb{I}_k^2 \to \mathbb{R}$ such that*

1. *$|f_k(e_{ij}) - g_k(a_i) - h_k(b_j)| \leq 10\eta$ for each cell $e_{ij} \in K_k$*
2. *$|g_k(a_i) - g_k(a_{i+1})| \leq 10\eta$ for all $a_i, a_{i+1} \in p_k(K_k)$*
3. *$|h_k(b_j) - h_k(b_{j+1})| \leq 10\eta$ for all $b_i, b_{i+1} \in q_k(K_k)$*
4. *$\max\{||g_k||, ||h_k||\} \leq 2||f_k||$.*

Proof (Proof of Theorem 2). Let $\varepsilon > 0$ be arbitrary. The function f is uniformly continuous on the compact set K, so there exists an $\alpha > 0$ such that $|f(x, y) - f(x', y')| < \varepsilon/10$ for each $(x, y), (x', y') \in K$ with $|(x, y) - (x', y')| < \alpha$. Let $l = 10||f||/\varepsilon$. According to Theorem 3 there exists a $k \in \mathbb{N}$ such that the set K_k approximating K is (α, l)-faithful. If we set β in Theorem 4 to $\varepsilon/10$, then $\eta = \varepsilon/10$ and it follows that there exist g_k and h_k such that $|f_k(e_{ij}) - g_k(a_i) - h_k(b_j)| \leq \varepsilon$ and $\max\{||g_k||, ||h_k||\} \leq 2||f_k|| \leq 2||f||$.

The construction of an approximate continuous decomposition $f_k \cong g_k + h_k$ in the proof of [11][Theorem 3.4] is quite involved already in the simple case of sets K with arrays of length at most $m = 2$. It can be modified to the case $m = 3$, and we believe that it can be modified to cover the case of a general m. For now the technical difficulties seem too big, though. Such a modification would provide a constructive proof of the theorem of Sternfeld [10] giving an answer to a question posed by Skopenkov. We thus conclude with the following problem which is still open.

Problem 3. Generalize the construction in the proof of Theorem 2 to a general m.

References

1. Arnold, V.I.: On functions of three variables. Dokl. Akad. Nauk SSSR 114, 679–681 (1957)
2. Arnold, V.I.: On the representation of continuous functions of three variables by superpositions of continuous functions of two variables. Math. Sb (N.S.) 48(90), 3–74 (1959)
3. Kolmogorov, A.N.: On the representations of continuous functions of many variables by superpositions of continuous functions fewer variables. Dokl. Akad. Nauk SSSR 108, 179–182 (1956)
4. Kolmogorov, A.N.: On the representations of continuous functions of many variables by superpositions of continuous functions of one variable and addition. Dokl. Akad. Nauk SSSR 114, 953–956 (1957)
5. Ostrand, P.A.: Dimension of metric spaces and Hilbert's problem 13. Bull. Amer. Math. Soc. 71, 619–622 (1965)
6. Rosenfeld, A.: Digital Topology. The Amer. Math. Monthly 86(8), 621–630 (1979)
7. Rosenfeld, A.: 'Continuous' functions on digital pictures. Pattern Recognition Letters 4, 177–184 (1986)
8. Skopenkov, A.: A description of continua basically embeddable in R^2. Topology Appl. 65, 29–48 (1995)
9. Sternfeld, Y.: Dimension, superposition of functions and separation of points, in compact metric spaces. Israel J. Math. 50, 13–52 (1985)
10. Sternfeld, Y.: Hilbert's 13th problem and dimension. Lect. Notes Math., vol. 1376, pp. 1–49. Springer, Heidelberg (1989)
11. Trenklerová, E.: Constructive decomposition of functions of two variables using functions of one variable. In: submitted to Proc. Amer. Math. Soc. (2007), arXiv:math/0702822v1

Algebraic Optimization of Relational Queries with Various Kinds of Preferences*

Radim Nedbal

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
radned@seznam.cz

Abstract. Preferences can be used for information filtering and extraction to deliver the most relevant data to the user. Therefore the efficient integration of querying with preferences into standard database technology is an important issue. The paper resumes a logical framework for formulating preferences and their embedding into relational algebra through a single *preference operator* parameterized by a set of user preferences of sixteen various kinds and returning only the most preferred subsets of its argument relation. Most importantly, preferences between sets of elements can be expressed. To make a relational query language with the preference operator useful for practical applications, formal foundation for algebraic optimization, applying heuristics like *push preference*, has to be provided. Therefore abstract properties of the preference operator and a variety of algebraic laws describing its interaction with other relational algebra operators are presented.

Keywords: logic of preference, relational query, optimization.

1 Introduction

If users have requirements that are to be satisfied completely, their database queries are characterized by *hard constraints*, delivering exactly the required objects if they exist and otherwise empty result. This is how traditional database query languages treat all the requirements on the data. However, requirements can be understood also in the sense of wishes: in case they are not satisfied, database users are usually prepared to accept worse alternatives and their database query is characterized by *soft constraints*. Requirements of the latter type are called preferences.

Preferences are ubiquitous in our daily lives, which suggests that database query languages should support both views of requirements, characterized by

* This work was supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization", and by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications".

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 388–399, 2008.

hard or soft constraints. The research on preferences is extensive and encompasses preference logic, preference reasoning, non-monotonic reasoning, and, recently, preferences also attracted attention in database community (see Sect. 5).

Building on a logical framework for formulating preferences and on their embedding into relational algebra (RA) through a single *preference operator* (PO) to combat the empty result and the flooding effects, this paper presents an approach to algebraic optimization of relational queries with various kinds of preferences. The PO selects from its argument relation the *best-matching alternatives* with regard to user preferences, but *nothing worse*.[1] Preferences are specified using a propositional logic notation and their semantics is related to that of a disjunctive logic program. The language for expressing preferences i) is declarative, ii) includes various kinds of preferences, iii) is rich enough to express preferences between sets of elements, iv) and has an intuitive, well defined semantics allowing for conflicting preferences.

In Sect. 2, the above mentioned framework for formulating preferences and in Sect. 3 an approach to their embedding into RA are revisited. Presenting a variety of algebraic laws that describe interaction with other RA operators to provide a formal foundation for algebraic optimization, Sect. 4 provides the main contribution of this paper.[2] A brief overview of related work in Sect. 5 and conclusions in Sect. 6 end this paper. All the nontrivial proofs are given.

To improve the readability, $\succeq (x, y) \wedge \neg \succeq (y, x)$ and $\succeq (x, y) \wedge \succeq (y, x)$ is substituted by $\succ (x, y)$ and $= (x, y)$, respectively.

2 User Preferences

A user preference is expressed by a preference statement, e.g. "a is preferred to b", or symbolically by an appropriate preference formula (PF). PF's comprise a simple declarative language for expressing preferences. To capture its declarative aspects, model-theoretic semantics is defined: considering a set of states of affaires S and a set $W = 2^S$ of all its subsets – worlds, if $\mathcal{M} = \langle W, \succeq \rangle$ is an order $\succeq$ on W such that $w \succeq w'$ holds for some worlds w, w' from W, then $\mathcal{M}$ is termed a *preference model* (PM) of $w > w'$ – a preference of the world w over the world w', which we express symbolically as $\mathcal{M} \models w > w'$.

The basic differentiation between preferences is based on notions of optimism and pessimism. Defining a-world as a world in which a occurs, if we are optimistic about a and pessimistic about b for example, we expect some a-world to precede at least one b-world in each PM of a preference statement "a is preferred to b". This kind of preference is called *opportunistic*. By contrast, if we are pessimistic about a and optimistic about b, we expect every a-world to precede each b-world in each PM of a preference statement "a is preferred to b". This kind of preference is called *careful*. Alternatively, we might be optimistic or pessimistic about both a and b. Then we expect some a world to precede each b-world or each a-world

[1] A similar concept was proposed by Kießling et al. [1,2] and Chomicki et al. [3] and, in a more restricted form, by Börzsönyi et al. [4] (for more detail refer to Sect. 5).

[2] The presented results correspond to those of [3].

to precede some b-world in each PM of a preference statement "a is preferred to b". This kind of preference is called *locally optimistic* or *locally pessimistic*, respectively. Locally optimistic, locally pessimistic, opportunistic and careful preferences are symbolically expressed by PF's of the form: $a \ {}^{M}{>}^{M}\ b$, $a \ {}^{m}{>}^{m}\ b$, $a \ {}^{M}{>}^{m}\ b$, and $a \ {}^{m}{>}^{M}\ b$, respectively.

Also, we distinguish between strict and non-strict preferences. For example, if w precedes w' strictly in a PM, then we strictly prefer w to w'.

In addition, we distinguish between preferences with and without *ceteris paribus* proviso – a notion introduced by von Wright [5] and generalized by Doyle and Wellman [6] by means of contextual equivalence relation – an equivalence relation on W.[3] For example, a PM of a preference statement "a is *carefully* preferred to b *ceteris paribus*" is such an order on W that a-worlds precede b-worlds in the same contextual equivalence class. Specifically, the preference statement "I prefer playing tenis to playing golf ceteris paribus" might express by means of an contextual equivalence that I prefer playing tenis to playing golf only if the context of weather is the same, i.e., it is not true that I prefer playing tenis in strong winds to playing golf during a sunny day.

Next, we revisit the basic definitions introducing syntax and model-theoretic semantics of the language for expressing user preferences:

Definition 1 (Language). Propositional formulas *are defined inductively:*

Given a finite set of propositional variables $p, q, \ldots$ i) every propositional variable is a propositional formula; ii) if φ, ψ are propositional formulas then so are $\varphi \wedge \psi$ and $\neg\varphi$.

PF's are expressions $\varphi \ {}^{x}{>}^{y}\ \psi$ and $\varphi \ {}^{x}{\geq}^{y}\ \psi$ for $x, y \in \{m, M\}$, where φ, ψ are propositional variables.

If we identify propositional variables with tuples over a relation schema R, then we get *PF's over* R. A relation instance $I(R)$, i.e., a set of tuples over R, creates a *world* w, an element of a set W.

The PM is defined so that any set of (possibly conflicting) preferences is consistent: the partial pre-order, i.e., a binary relation which is reflexive and transitive, in the definition of the PM, enables to express some kind of conflict by incomparability:

Definition 2 (Preference model). *A PM $\mathcal{M} = \langle W, \succeq \rangle$ over a relation schema R is a couple in which W is a set of worlds, relation instances of R, and $\succeq$ is a partial pre-order over W, the* preference relation *over R.*

A set of user preferences of various kinds can by represented symbolically by a *preference specification* (PS), which corresponds to an appropriate complex PF in the above defined language.

Definition 3 (Preference specification). *Let R be a relation schema and $\mathcal{P}_{\rhd}$ a set of PF's over R of the form $\{\varphi_i \rhd \psi_i : i = 1, \ldots, n\}$. A PS $\mathcal{P}$ over R is*

[3] As it has been shown [7] that any preference with contextual equivalence specification can be expressed by a set of preferences without contextual specification, we can restrict ourselves only to preferences without ceteris paribus proviso.

a tuple $\langle \mathcal{P}_\rhd | \rhd \in \{ \ ^x>^y, \ ^x\geq^y \ | x, y \in \{m, M\}\}\rangle$*, and* $\mathcal{M}$ *is its model, i.e., a* PS model*, iff it models all elements* $\mathcal{P}_\rhd$ *of the tuple. Interpreting*

$$\mathcal{M} \models \mathcal{P}_\rhd \iff \forall(\varphi_i \rhd \psi_i) \in \mathcal{P}_\rhd : \mathcal{M} \models \varphi_i \rhd \psi_i \ .$$

3 Preference Operator

To embed preferences into RQL, the PO $\omega_\mathcal{P}$ returning only the best sets of tuples in the sense of user preferences $\mathcal{P}$ is defined:

Definition 4 (Preference operator). *The PO* ω *is a mapping from a powerset into itself. Specifically, if* R *is a relation schema,* $\mathcal{P}$ *a PS over* R*, and* $\mathscr{M}$ *the set of its models; then the PO* $\omega_\mathcal{P}$ *is defined for all sets* $\{I_1(R), \ldots, I_n(R)\}$ *of instances of* R *as follows:*

$$\omega_\mathcal{P}(\{I_1(R), \ldots, I_n(R)\}) = \big\{w \in \{I_1(R), \ldots, I_n(R)\} \big| \\ \exists \mathcal{M}_k = \langle W, \succeq_k \rangle \in \mathscr{M}, \forall w' \in \{I_1(R), \ldots, I_n(R)\} : \succeq_k (w', w) \Rightarrow \succeq_k (w, w')\big\} \ .$$

Remark 1 (Preference operator notation). For brevity, when writing the argument without braces, e.g., $\omega_\mathcal{P}(I(R))$, then the unabbreviated notation is $\omega_\mathcal{P}(\{2^{I(R)}\})$, showing that the argument is the powerset of $I(R)$.

3.1 Basic Properties

The following propositions are essential for investigation of algebraic properties describing interaction of the PO with other RA operations:

Proposition 1. *Given a relation schema* R *and a PS* $\mathcal{P}$ *over* R*, for all instances* $I(R)$ *of* R *the following properties hold:*

$$\omega_\mathcal{P}(I(R)) \subseteq 2^{I(R)} \ , \\ \omega_\mathcal{P}\left(\{\omega_\mathcal{P}(I(R))\}\right) = \omega_\mathcal{P}(I(R)) \ , \\ \omega_{\mathcal{P}_{empty}}(I(R)) = 2^{I(R)} \ ,$$

where $\mathcal{P}_{empty}$ *is the empty PS, i.e., containing no preference.*

The PO is not *monotone* or *antimonotone* with respect to its relation argument. However, partial antimonotonicity holds:

Proposition 2 (Partial antimonotonicity). *Given a relation schema* R *and a PS* $\mathcal{P}$ *over* R*, for all instances* $I(R), I'(R)$ *of* R *the following property holds:*

$$I(R) \subseteq I'(R) \Rightarrow 2^{I(R)} \cap \omega_\mathcal{P}(I'(R)) \subseteq \omega_\mathcal{P}(I(R)) \ .$$

Proof. Assume $w \in 2^{I(R)} \cap \omega_\mathcal{P}(I'(R))$. It follows that $w \subseteq I(R)$ and from the definition (Def. 4) of the PO $w \subseteq I'(R) \wedge \exists \mathcal{M}_k \in \mathscr{M}$ s.t. $\forall w' \in W : w' \subseteq I'(R) \wedge \succeq_k (w', w) \Rightarrow \succeq_k (w, w')$. As $I(R) \subseteq I'(R)$, we can conclude that $\exists \mathcal{M}_k \in \mathscr{M}$ s.t. $\forall w' \in W : w' \subseteq I(R) \wedge \succeq_k (w', w) \Rightarrow \succeq_k (w, w')$, which together with $w \subseteq I(R)$ implies $w \in \omega_\mathcal{P}(I(R))$. □

The following theorems enable to reduce cardinality of an argument relation of the PO without changing the return value and ensure that the empty query result effect is successfully eliminated:

Theorem 1 (Reduction). *Given a relation schema R, a PS $\mathcal{P}$ over R, for all instances $I(R), I'(R)$ of R the following property holds:*

$$I(R) \subseteq I'(R) \wedge \omega_{\mathcal{P}}(I'(R)) \subseteq 2^{I(R)} \Rightarrow \omega_{\mathcal{P}}(I(R)) = \omega_{\mathcal{P}}(I'(R)) \ .$$

Proof. $\subseteq$: Assume $w \in \omega_{\mathcal{P}}(I(R))$. Then, it follows from the definition of the PO $w \subseteq I(R) \wedge \exists \mathcal{M}_k \in \mathscr{M}$ s.t. $\forall w' \subseteq I(R) : \succeq_k (w', w) \Rightarrow \succeq_k (w, w')$. The assumption $\omega_{\mathcal{P}}(I'(R)) \subseteq 2^{I(R)}$ implies $\forall w' \in 2^{I'(R)} - 2^{I(R)} : \neg \succeq_k (w', w)$, and we can conclude $\forall w' \subseteq I'(R) : \succeq_k (w', w) \Rightarrow \succeq_k (w, w')$, which together with the assumption $I(R) \subseteq I'(R)$ implies $w \subseteq I'(R) \wedge \exists \mathcal{M}_k \in \mathscr{M}$ s.t. $\forall w' \subseteq I'(R) : \succeq_k (w', w) \Rightarrow \succeq_k (w, w')$, the definition of $w \in \omega_{\mathcal{P}}(I'(R))$.
$\supseteq$: Immediately follows from Prop. 2. □

Theorem 2 (Non-emptiness). *Given a relation schema R, a PS $\mathcal{P}$ over R, then for every finite, nonempty instance $I(R)$ of R, $\omega_{\mathcal{P}}(I(R))$ is nonempty.*

3.2 Multidimensional Composition

In multidimensional composition, we have a number of PS defined over several relation schemas, and we define PS over the Cartesian product of those relations: the most common ways are Pareto and lexicographic composition.

Definition 5 (Pareto composition). *Given two relation schemas R_1 and R_2, PS's $\mathcal{P}_1$ over R_1 and $\mathcal{P}_2$ over R_2, and their sets of models $\mathscr{M}_1$ and $\mathscr{M}_2$, the Pareto composition $P(\mathcal{P}_1, \mathcal{P}_2)$ of $\mathcal{P}_1$ and $\mathcal{P}_2$ is a PS $\mathcal{P}_0$ over the Cartesian product $R_1 \times R_2$, whose set of models $\mathscr{M}_0$ is defined as:*

$$\forall \mathcal{M}_m = \langle W_1 \times W_2, \succeq_m \rangle \in \mathscr{M}_0,$$
$$\exists \mathcal{M}_k = \langle W_1, \succeq_k \rangle \in \mathscr{M}_1, \exists \mathcal{M}_l = \langle W_2, \succeq_l \rangle \in \mathscr{M}_2 \ s.t.$$
$$\forall w_1, w_1' \in W_1, \forall w_2, w_2' \in W_2 :$$
$$\succeq_m (w_1 \times w_2, w_1' \times w_2') \ \equiv \succeq_k (w_1, w_1') \wedge \succeq_l (w_2, w_2') \ .$$

Definition 6 (Lexicographic composition). *Given two relation schemas R_1 and R_2, PS's $\mathcal{P}_1$ over R_1 and $\mathcal{P}_2$ over R_2, and their sets of models $\mathscr{M}_1$ and $\mathscr{M}_2$, the lexicographic composition $L(\mathcal{P}_1, \mathcal{P}_2)$ of $\mathcal{P}_1$ and $\mathcal{P}_2$ is a PS $\mathcal{P}_0$ over the Cartesian product $R_1 \times R_2$, whose set of models $\mathscr{M}_0$ is defined as:*

$$\forall \mathcal{M}_m = \langle W_1 \times W_2, \succeq_m \rangle \in \mathscr{M}_0,$$
$$\exists \mathcal{M}_k = \langle W_1, \succeq_k \rangle \in \mathscr{M}_1, \exists \mathcal{M}_l = \langle W_2, \succeq_l \rangle \in \mathscr{M}_2 \ s.t.$$
$$\forall w_1, w_1' \in W_1, \forall w_2, w_2' \in W_2 :$$
$$\succeq_m (w_1 \times w_2, w_1' \times w_2') \ \equiv \succ_k (w_1, w_1') \vee (=_k (w_1, w_1') \wedge \succeq_l (w_2, w_2')) \ .$$

4 Algebraic Optimization

As the PO extends RA, the optimization of queries with preferences can be realized as an extension of a classical relational query optimization. Most importantly, we can inherit all well known laws from RA, which, together with algebraic laws governing the commutativity and distributivity of the PO with respect to RA operations, constitute a formal foundation for rewriting queries with preferences using the standard strategies (*push selection*, *push projection*) aiming at reducing the sizes of intermediate relations.

Remark 2 (RA operators notation). In the following, RA selection and projection are generalized so that they can operate on set arguments, denoted by braces, e.g., $\sigma_\varphi(\{\omega_\mathcal{P}(I(R))\})$. The corresponding definitions are indicated by $\stackrel{\text{def}}{=}$.

4.1 Commuting with Selection

The following theorem identifies a sufficient condition under which the PO commutes with RA selection:

Theorem 3 (Commuting with selection). *Given a relation schema R, a PS $\mathcal{P}$ over R, the set of its PM's $\mathscr{M}$, and a selection condition φ over R, if*

$$\forall \mathcal{M}_k = \langle W, \succeq_k \rangle \in \mathscr{M}, \forall w, w' \in W : \quad \succ_k (w', w) \wedge w = \sigma_\varphi(w) \Rightarrow w' = \sigma_\varphi(w')$$

is a valid formula, then for any relation instance $I(R)$ of R:

$$\omega_\mathcal{P}(\sigma_\varphi(I(R))) = \sigma_\varphi(\{\omega_\mathcal{P}(I(R))\}) \stackrel{def}{=} \{w \in \omega_\mathcal{P}(I(R)) | \sigma_\varphi(w) = w\} .$$

Proof. Observe that:

$$w \in \omega_\mathcal{P}(\sigma_\varphi(I(R))) \equiv w \subseteq I(R) \wedge \sigma_\varphi(w) = w \wedge \neg(\forall \mathcal{M}_k \in \mathscr{M}, \exists w' \subseteq I(R) : (\sigma_\varphi(w') = w' \wedge \succ_k (w', w)) .$$

$$w \in \sigma_\varphi(\{\omega_\mathcal{P}(I(R))\}) \equiv w \subseteq I(R) \wedge \sigma_\varphi(w) = w \wedge \neg(\forall \mathcal{M}_k \in \mathscr{M}, \exists w' \subseteq I(R) : \succ_k (w', w)) ,$$

Obviously, the second formula implies the first. To see that the opposite implication also holds, we prove that $w \notin \sigma_\varphi(\{\omega_\mathcal{P}(I(R))\}) \Rightarrow w \notin \omega_\mathcal{P}(\sigma_\varphi(I(R)))$. There are three cases when $w \notin \sigma_\varphi(\{\omega_\mathcal{P}(I(R))\})$. If $w \nsubseteq I(R)$ or $\sigma_\varphi(w) \neq w$, it is immediately clear that $w \notin \omega_\mathcal{P}(\sigma_\varphi(I(R)))$. In the third case, $\forall \mathcal{M}_k \in \mathscr{M}, \exists w' \subseteq I(R) : \succ_k (w', w)$. However, due to the theorem assumption, $\forall \mathcal{M}_k \in \mathscr{M}, \exists w' \subseteq I(R) : \sigma_\varphi(w') = w' \wedge \succ_k (w', w)$, which completes the proof. □

4.2 Commuting with Projection

The following theorem identifies sufficient conditions under which the PO commutes with RA projection. To prepare the ground for the theorem, some definitions have to be introduced:

Definition 7 (Restriction of a preference relation). *Given a relation schema R, a set of attributes X of R, and a preference relation $\succeq$ over R, the restriction $\theta_X(\succeq)$ of $\succeq$ to X is a preference relation $\succeq_X$ over $\pi_X(R)$ defined using the following formula:*

$$\succeq_X (w_X, w'_X) \equiv \forall w, w' \in W : \pi_X(w) = w_X \wedge \pi_X(w') = w'_X \Rightarrow \succeq (w, w') .$$

Definition 8 (Restriction of the preference model). *Given a relation schema R, a set of relation attributes X of R, and a PM $\mathcal{M} = \langle W, \succeq \rangle$ over R, the restriction $\theta_X(\mathcal{M})$ of $\mathcal{M}$ to X is a PM $\mathcal{M}_X = \langle W_X, \succeq_X \rangle$ over $\pi_X(R)$ where $W_X = \{\pi_X(w) \mid w \in W\}$.*

Definition 9 (Restriction of the preference operator). *Given a relation schema R, a set of attributes X of R, a PS $\mathcal{P}$ over R, and the set $\mathscr{M}_X$ of its models restricted to X, the restriction $\theta_X(\omega_\mathcal{P})$ of the PO $\omega_\mathcal{P}$ to X is the PO $\omega^X_\mathcal{P}$ defined as follows:*

$$\omega^X_\mathcal{P}(\pi_X(I(R))) = \{w_X \subseteq \pi_X(I(R)) \mid \exists \mathcal{M}_X \in \mathscr{M}_X \text{ s.t. } \forall w'_X \subseteq \pi_X(I(R)) : \succeq_X (w'_X, w_X) \Rightarrow \succeq_X (w_X, w'_X)\} .$$

Theorem 4 (Commuting with projection). *Given a relation schema R, a set of attributes X of R, a PS $\mathcal{P}$ over R, and the set of its PM's $\mathscr{M}$, if the following formulae*

$$\forall \mathcal{M}_k \in \mathscr{M}, \forall w_1, w_2, w_3 \in W : \pi_X(w_1) = \pi_X(w_2) \wedge \pi_X(w_1) \neq \pi_X(w_3) \wedge \succeq_k (w_1, w_3) \Rightarrow \succeq_k (w_2, w_3) ,$$

$$\forall \mathcal{M}_k \in \mathscr{M}, \forall w_1, w_3, w_4 \in W : \pi_X(w_3) = \pi_X(w_4) \wedge \pi_X(w_1) \neq \pi_X(w_3) \wedge \succeq_k (w_1, w_3) \Rightarrow \succeq_k (w_1, w_4)$$

are valid, then for any relation instance $I(R)$ of R:

$$\omega^X_\mathcal{P}(\pi_X(I(R))) = \pi_X(\{\omega_\mathcal{P}(I(R))\}) \stackrel{def}{=} \{\pi_X(w) \mid w \in \omega_\mathcal{P}(I(R))\} .$$

Proof. We prove: $\pi_X(w) \notin \omega^X_\mathcal{P}(\pi_X(I(R))) \Longleftrightarrow \pi_X(w) \notin \pi_X(\{\omega_\mathcal{P}(I(R))\})$.

$\Rightarrow$: Assume $\pi_X(w_3) \notin \omega^X_\mathcal{P}(\pi_X(I(R)))$. The case $\pi_X(w_3) \nsubseteq \pi_X(I(R))$ is trivial. Otherwise, it must be the case that $\forall \mathcal{M}_X \in \mathscr{M}_X, \exists w_X \subseteq \pi_X(I(R)) : \succ_X (w_X, \pi_X(w_3))$, which implies $\forall \mathcal{M}_k \in \mathscr{M}, \forall w_1, w_4 \in W : \pi_X(w_1) = w_X \wedge \pi_X(w_4) = \pi_X(w_3) \Rightarrow \succ_k (w_1, w_4)$ and thus $\pi_X(w_3) \notin \pi_X(\{\omega_\mathcal{P}(I(R))\})$.

$\Leftarrow$: Assume $\pi_X(w_3) \notin \pi_X(\{\omega_\mathcal{P}(I(R))\})$. Then $\forall \mathcal{M}_k \in \mathscr{M}$ and $\forall w_4 \subseteq I(R)$ s.t. $\pi_X(w_4) = \pi_X(w_3)$, there is $w_1 \subseteq I(R)$ s.t. $\succ_k (w_1, w_4)$ and $\pi_X(w_1) \neq \pi_X(w_4)$. From the assumption of the theorem, it follows that $\forall w_2, w_4 \subseteq I(R) : \pi_X(w_2) = \pi_X(w_1) \wedge \pi_X(w_4) = \pi_X(w_3) \Rightarrow \succ_k (w_2, w_4)$, which implies $\theta_X(\succ_k)(\pi_X(w_1), \pi_X(w_3))$ and thus $\pi_X(w_3) \notin \omega^X_\mathcal{P}(\pi_X(I(R)))$. □

4.3 Distributing over Cartesian Product

For the PO to distribute over the Cartesian product of two relations, the PS, which is the parametr of the PO, needs to be decomposed into the PS's that will distribute into the argument relations. We obtain the same property for both Pareto and lexicographic composition:

Theorem 5 (Distributing over Cartesian product). *Given two relation schemas R_1 and R_2, and PS's $\mathcal{P}_1$ over R_1 and $\mathcal{P}_2$ over R_2, for any two relation instances $I(R_1)$ and $I(R_2)$ of R_1 and R_2:*

$$\omega_{\mathcal{P}_0}(I(R_1) \times I(R_2)) = \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2)) \stackrel{def}{=} \{w_1 \times w_2 \mid w_1 \in \omega_{\mathcal{P}_1}(I(R_1)) \wedge w_2 \in \omega_{\mathcal{P}_2}(I(R_2))\} ,$$

where $\mathcal{P}_0 = P(\mathcal{P}_1, \mathcal{P}_2)$ is a Pareto composition of $\mathcal{P}_1$ and $\mathcal{P}_2$.

Proof. We prove:

$$w_1 \times w_2 \notin \omega_{\mathcal{P}_0}(I(R_1) \times I(R_2)) \Longleftrightarrow w_1 \times w_2 \notin \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2)) .$$

$\Rightarrow$: Assume $w_1 \times w_2 \notin \omega_{\mathcal{P}_0}(I(R_1) \times I(R_2))$. Then $\forall \mathcal{M}_m \in \mathscr{M}_0$, models of $\mathcal{P}_0$, there are $w_1' \subseteq I(R_1), w_2' \subseteq I(R_2)$ s.t. $\succ_m (w_1' \times w_2', w_1 \times w_2)$. Consequently, $\forall \mathcal{M}_k \in \mathscr{M}_1, \forall \mathcal{M}_l \in \mathscr{M}_2$, models of $\mathcal{P}_1$ and $\mathcal{P}_2$, there are $w_1' \subseteq I(R_1), w_2' \subseteq I(R_2)$ s.t. $\succ_k (w_1', w_1)$ or $\succ_l (w_2', w_2)$, which implies $w_1 \notin \omega_{\mathcal{P}_1}(I(R_1))$ or $w_2 \notin \omega_{\mathcal{P}_2}(I(R_2))$ and thus $w_1 \times w_2 \notin \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2))$.

$\Leftarrow$: Assume $w_1 \times w_2 \notin \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2))$. Then $w_1 \notin \omega_{\mathcal{P}_1}(I(R_1))$ or $w_2 \notin \omega_{\mathcal{P}_2}(I(R_2))$. Assume the first. Then $\forall \mathcal{M}_k \in \mathscr{M}_1$, models of $\mathcal{P}_1$, there must be $w_1' \subseteq I(R_1)$ s.t. $\succ_k (w_1', w_1)$. Consequently, $\forall \mathcal{M}_m \in \mathscr{M}_0$, models of $\mathcal{P}_0$, $\exists w_1' \subseteq I(R_1) : \succ_m (w_1' \times w_2, w_1 \times w_2)$, which implies $w_1 \times w_2 \notin \omega_{\mathcal{P}_0}(I(R_1) \times I(R_2))$. The second case is symmetric. □

Theorem 6 (Distributing over Cartesian product). *Given two relation schemas R_1 and R_2, and PS's $\mathcal{P}_1$ over R_1 and $\mathcal{P}_2$ over R_2, for any two relation instances $I(R_1)$ and $I(R_2)$ of R_1 and R_2:*

$$\omega_{\mathcal{P}_0}(I(R_1) \times I(R_2)) = \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2)) \stackrel{def}{=} \{w_1 \times w_2 \mid w_1 \in \omega_{\mathcal{P}_1}(I(R_1)) \wedge w_2 \in \omega_{\mathcal{P}_2}(I(R_2))\} ,$$

where $\mathcal{P}_0 = L(\mathcal{P}_1, \mathcal{P}_2)$ is a lexicographic composition of $\mathcal{P}_1$ and $\mathcal{P}_2$.

Proof. We prove:

$$w_1 \times w_2 \notin \omega_{\mathcal{P}_0}(I(R_1) \times I(R_2)) \Longleftrightarrow w_1 \times w_2 \notin \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2)) .$$

$\Rightarrow$: Assume $w_1 \times w_2 \notin \omega_{\mathcal{P}_0}(I(R_1) \times I(R_2))$. Then $\forall \mathcal{M}_m \in \mathscr{M}_0$, models of $\mathcal{P}_0$, there are $w_1' \subseteq I(R_1), w_2' \subseteq I(R_2)$ s.t. $\succ_m (w_1' \times w_2', w_1 \times w_2)$. Consequently, $\forall \mathcal{M}_k \in \mathscr{M}_1, \forall \mathcal{M}_l \in \mathscr{M}_2$, models of $\mathcal{P}_1$ and $\mathcal{P}_2$, there are $w_1' \subseteq I(R_1), w_2' \subseteq I(R_2)$ s.t. $\succ_k (w_1', w_1)$ or $=_k (w_1', w_1) \wedge \succ_l (w_2', w_2)$, which implies $w_1 \notin \omega_{\mathcal{P}_1}(I(R_1))$ or $w_2 \notin \omega_{\mathcal{P}_2}(I(R_2))$ and thus $w_1 \times w_2 \notin \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2))$.

$\Leftarrow$: Assume $w_1 \times w_2 \notin \omega_{\mathcal{P}_1}(I(R_1)) \times \omega_{\mathcal{P}_2}(I(R_2))$. Then $w_1 \notin \omega_{\mathcal{P}_1}(I(R_1))$ or $w_2 \notin \omega_{\mathcal{P}_2}(I(R_2))$. Assume the first. Then $\forall \mathcal{M}_k \in \mathscr{M}_1$, models of $\mathcal{P}_1$, there must be $w_1' \subseteq I(R_1)$ s.t. $\succ_k (w_1', w_1)$. Consequently, $\forall \mathcal{M}_m \in \mathscr{M}_0$, models of $\mathcal{P}_0$, there must be w_1' s.t. $\succ_m (w_1' \times w_2, w_1 \times w_2)$, which implies $w_1 \times w_2 \notin \omega_{\mathcal{P}_0}(I(R_1) \times I(R_2))$. The second case is symmetric. □

The equality $\omega_{\mathcal{P}_{\text{empty}}}(I(R)) = 2^{I(R)}$ and both Theorem 5 and Theorem 6 make it possible to derive the transformation rule that pushes the PO with a one-dimensional PS down the appropriate argument of the Cartesian product:

Corollary 1. *Given two relation schemas R_1 and R_2, a PS's $\mathcal{P}_1$ over R_1, and an empty PS $\mathcal{P}_2$ over R_2, for any two relation instances $I(R_1)$ and $I(R_2)$ of R_1 and R_2, the following property holds:*

$$\omega_{\mathcal{P}_0}(I(R_1) \times I(R_2)) = \omega_{\mathcal{P}_1}(I(R_1)) \times 2^{I(R_2)} \stackrel{def}{=} \{w_1 \times w_2 \mid w_1 \in \omega_{\mathcal{P}_1}(I(R_1)) \wedge w_2 \subseteq I(R_2)\} ,$$

where $\mathcal{P}_0 = P(\mathcal{P}_1, \mathcal{P}_2)$ is a Pareto of lexicographic composition of $\mathcal{P}_1$ and $\mathcal{P}_2$.

4.4 Distributing over Union

The following theorem shows how the PO distributes over RA union:

Theorem 7 (Distributing over union). *Given two compatible relation schemas[4] R and S, and a PS $\mathcal{P}$ over R (and S), if the following formula*

$$\omega_{\mathcal{P}}(I(R) \cup I(S)) \subseteq 2^{I(R)} \cup 2^{I(S)}$$

is valid for relation instances $I(R)$ and $I(S)$ of R and S, then:

$$\omega_{\mathcal{P}}(I(R) \cup I(S)) = \omega_{\mathcal{P}}(\{\omega_{\mathcal{P}}(I(R)) \cup \omega_{\mathcal{P}}(I(S))\}) .$$

Proof. It follows from Proposition 1 that $\omega_{\mathcal{P}}(I(R)) \cup \omega_{\mathcal{P}}(I(S)) \subseteq 2^{I(R)} \cup 2^{I(S)} \subseteq 2^{I(R) \cup I(S)}$. If we show that $\omega_{\mathcal{P}}(I(R) \cup I(S)) \subseteq \omega_{\mathcal{P}}(I(R)) \cup \omega_{\mathcal{P}}(I(S))$, then the theorem follows from Theorem 1.

If $w \in \omega_{\mathcal{P}}(I(R) \cup I(S))$, then it follows from the definition of the PO $w \subseteq I(R) \cup I(S) \wedge \exists \mathcal{M}_k \in \mathscr{M}$ s.t. $\forall w' \subseteq I(R) \cup I(S) : \succeq_k (w', w) \Rightarrow \succeq_k (w, w')$. As $w \subseteq I(R) \vee w \subseteq I(S)$ from the assumption of the theorem and $2^{I(R)} \cup 2^{I(S)} \subseteq 2^{I(R) \cup I(S)}$, we can conclude $(w \subseteq I(R) \vee w \subseteq I(S)) \wedge \forall w' \in 2^{I(R)} \cup 2^{I(S)} : \succeq_k (w', w) \Rightarrow \succeq_k (w, w')$, implying $w \in \omega_{\mathcal{P}}(I(R)) \cup \omega_{\mathcal{P}}(I(S))$. □

4.5 Distributing over Difference

Only in the trivial case, the the distribution over RA difference is possible:

[4] We call two relation schemas *compatible* if they have the same number of attributes and the corresponding attributes have identical domains.

Theorem 8 (Distributing over difference). *Given two compatible relation schemas R and S, and a PS $\mathcal{P}$ over R (and S), if the following formula*

$$\omega_{\mathcal{P}}(I(R)) \subseteq 2^{I(R)-I(S)} \cup 2^{I(S)}$$

is valid for relation instances $I(R) \neq I(S)$ of R and S, then:

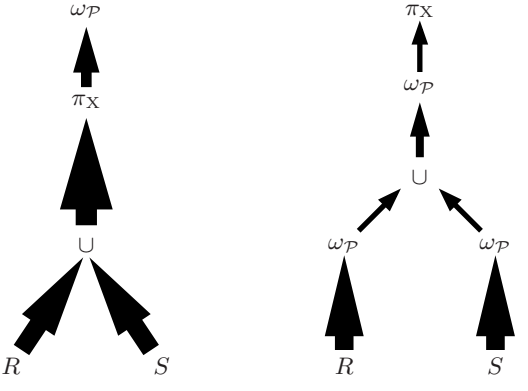
$$\omega_{\mathcal{P}}(I(R) - I(S)) = \omega_{\mathcal{P}}(I(R)) - \omega_{\mathcal{P}}(I(S))$$

iff the PS $\mathcal{P}$ is empty.

4.6 *Push Preference*

The question arises how to integrate the above algebraic laws into the classical, well-known hill-climbing algorithm. In particular, we want to add heuristic strategy of *push preference*, which is based on the assumption that early application of the PO reduces intermediate results. Indeed, the Theorem 1 provides a formal evidence that it is correct to pass exactly all the tuples that have been included in any world returned by the PO to the next operator in the operator tree. This leads to a better performance in subsequent operators.

Example 1. Consider a simple query expressed in RA as: $\omega_{\mathcal{P}}(\pi_X(R \cup S))$. After applying the preference strategy, we get: $\pi_X(\omega_{\mathcal{P}}(\{\omega_{\mathcal{P}}(R) \cup \omega_{\mathcal{P}}(S)\}))$. The corresponding expression trees are depicted in Fig. 1, where data flow between the computer's main memory and secondary storage is represented by line width.



(a) Before pushing (b) After pushing

Fig. 1. Improving the query plan by pushing PO down the expression tree

We have supposed that relations R and S are too big to fit into main memory. Using the number of the secondary storage I/O's as our measure of cost for an operation, it can be seen that the strategy of pushing the PO improves the performance in this case significantly.

5 Related Work

The study of preferences in the context of database queries has been originated by Lacroix and Lavency [8]. They, however, don't deal with algebraic optimization.

Following their work, *preference datalog* was introduced in [9], where it was shown that the concept of preference provides a modular and declarative means for formulating optimization and relaxation queries in deductive databases.

Nevertheless, only at the turn of the millennium this area attracted broader interest again. Kießling [1] and Chomicki et al. [3] have pursued independently a similar, *qualitative* approach within which preferences between tuples are specified directly, using binary *preference relations*. The embedding into RQL they have used is similar to ours: they have defined an operator returning only the best preference matches. However, they, by contrast to the approach presented in this paper, don't consider preferences between *sets* of elements and are concerned only with one type of preference. Moreover, the relation to a preference logic unfortunately is unclear. On the other hand, both Chomicki et. al. [3] and Kießling [2,10] have laid the foundation for preference query optimization that extends established query optimization techniques: preference queries can be evaluated by extended – preference RA. While some transformation laws for queries with preferences have been presented in [2,10], the results presented in [3] are mostly more general.

A special case of the same embedding represents *skyline operator* introduced by Börzsönyi et al. [4]. Some examples of possible rewritings for skyline queries are given but no general rewriting rules are formulated.

In [11], actual values of an arbitrary attribute were allowed to be partially ordered according to user preferences. Accordingly, RA operations, aggregation functions and arithmetic were redefined. However, some of their properties were lost, and the the query optimization issues were not discussed.

6 Conclusions

We build on the framework of embedding preferences into RQL through the PO that is parameterized by user preferences expressed in a declarative, logical language containing sixteen kinds of preferences and that returns the most preferred sets of tuples of its argument relation. Most importantly, the language is suitable for expressing preferences between sets of elements and its semantics allows for conflicting preferences.

The main contribution of the paper consists in presenting basic properties of the PO and a number of algebraic laws describing its interaction with other RA operators. Particularly, sufficient conditions for commuting the PO with RA selection or projection and for distributing over Cartesian product, set union, and set difference have been identified. Thus key rules for rewriting the preference queries using the standard algebraic optimization strategies like *push selection* or *push projection* have been established. Moreover, a new optimization strategy of *push preference* has been suggested.

Future work directions include identifying further algebraic properties and finding the best possible ordering of transformations to compose an effective hill-climbing algorithm for optimization of RA statements with the PO. Also, expressiveness of RA including the PO and complexity issues have to be addressed in detail.

References

1. Kießling, W.: Foundations of Preferences in Database Systems. In: Proceedings of the 28th VLDB Conference, Hong Kong, China, pp. 311–322 (2002)
2. Kießling, W., Hafenrichter, B.: Algebraic optimization of relational preference queries. Technical Report 2003-01, Institute of Computer Science, University of Augsburg (February 2003)
3. Chomicki, J.: Preference Formulas in Relational Queries. ACM Trans. Database Syst. 28(4), 427–466 (2003)
4. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering, pp. 421–430. IEEE Computer Society, Washington, DC, USA (2001)
5. von Wright, G.: The logic of preference. Edinburgh University Press, Edinburgh (1963)
6. Doyle, J., Wellman, M.P.: Representing preferences as ceteris paribus comparatives. In: Decision-Theoretic Planning: Papers from the 1994 Spring AAAI Symposium, pp. 69–75. AAAI Press, Menlo Park, California (1994)
7. Kaci, S., van der Torre, L.W.N.: Non-monotonic reasoning with various kinds of preferences. In: Brafman, R.I., Junker, U. (eds.) IJCAI 2005. Multidisciplinary Workshop on Advances in Preference Handling, pp. 112–117 (2005)
8. Lacroix, M., Lavency, P.: Preferences; Putting More Knowledge into Queries. In: Stocker, P.M., Kent, W., Hammersley, P. (eds.) VLDB, pp. 217–225. Morgan Kaufmann, San Francisco (1987)
9. Govindarajan, K., Jayaraman, B., Mantha, S.: Preference datalog. Technical Report 95-50 (January 1995)
10. Hafenrichter, B., Kießling, W.: Optimization of relational preference queries. In: CRPIT '39. Proceedings of the sixteenth Australasian conference on Database technologies, Darlinghurst, pp. 175–184. Australian Computer Society, Inc., Australia (2005)
11. Nedbal, R.: Relational Databases with Ordered Relations. Logic Journal of the IGPL 13(5), 587–597 (2005)

Mortality Problem for 2×2 Integer Matrices

C. Nuccio and E. Rodaro*

Dipartimento di Matematica, Politecnico di Milano, Piazza L.da Vinci, 32, 20133 Milano, Italy
claudia.nuccio@polimi.it, emanuele.rodaro@mate.polimi.it

Abstract. A given set F of $n \times n$ matrices is said to be mortal if the $n \times n$ null matrix belongs to the free semigroup generated by F. It is known that the mortality problem for 3×3 matrices with integer entries is undecidable [7],[3]. In this paper we prove that the mortality problem is decidable for any set of 2×2 integer matrices whose determinants assume the values $0, \pm 1$.

1 Introduction

Some very simple questions on semigroups of matrices with integer entries are undecidable even for low dimensions. In particular, in [7] it was proved that, given a set of 3×3 matrices over the integers $\mathbb{Z}$, the mortality problem is undecidable, i.e. it is undecidable whether or not the zero matrix belongs to the semigroup generated by this set. It is an open problem whether the mortality problem for integer matrices of dimension 2 is decidable and we refer to [1], [4] and [5] for some motivations, references and discussions. In this paper we prove that the mortality problem is decidable for each finite set of integer matrices of order 2 such that all non singular matrices are in $GL(2, \mathbb{Z})$, i.e. have determinant ± 1. We use an approach that closely follows the technique used by Choffrut and Karhumäki in [2], where it is proved that, for 2×2 integer matrices, it is decidable the membership of a given non singular matrix to a given finitely generated semigroup. We remark that in [6] it was proved that the mortality problem is decidable for any set of row-monomial matrices.

2 Preliminaries

Let $M(2, \mathbb{Z})$ be the set of 2×2 matrices with integer entries which is a monoid with respect to the ordinary matrix product. We will use the symbols I and $\underline{0}$ respectively for the identity and the null matrices of order 2. Moreover we denote by $GL(2, \mathbb{Z})$ the general linear group, by $SL(2, \mathbb{Z})$ the special linear group and by $PSL(2, \mathbb{Z})$ the projective special linear group of degree 2.

For each $A \in M(2, \mathbb{Z})$, let ι_A and κ_A be respectively the image and the kernel of the endomorphism of $\mathbb{R}^2$ associated to A. If $rank(A) = 1$, then ι_A and κ_A are

* This research was done with the partial support of GNSAGA, PRIN "Automi e Linguaggi Formali: aspetti matematici e applicativi" and ESF project AutoMathA.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 400–405, 2008.

1-dimensional $\mathbb{R}$-subspaces. Throughout the paper, $(x, y)^T$ stands for the column vector whose components are $x, y \in \mathbb{R}$ and e_1 for the column vector $(1, 0)^T$.

The following lemma gives a way to choose particular generators of ι_A and κ_A useful for the sequel:

Lemma 1. *Let $A \in M(2, \mathbb{Z})$ with $rank(A) = 1$. Then ι_A (resp. κ_A) contains a vector $(x, y) \in \mathbb{Z}^2$ such that $gcd(x, y) = 1$.*

Proof. The case of ι_A is trivial. In the case of κ_A, applying the Gauss elimination method to the system $Av^T = (0, 0)^T$, we get that $v \in \mathbb{Q}^2$. Hence, there exists $\lambda \in \mathbb{N}$ such that $\lambda v \in \mathbb{Z}^2$ and so, dividing λv by the greatest common divisor of its components, we obtain the vector (x, y) of the statement. □

Let $F = \{M_1, \ldots, M_m\} \subseteq M(2, \mathbb{Z})$ be a finite set and let $H = \langle M_1, \ldots, M_m \rangle$ be the subsemigroup of $M(2, \mathbb{Z})$ generated by F. The set F is called *mortal* if and only if $\underline{0} \in H$. We recall a lemma of [1]:

Lemma 2 ([1]). *A finite set $\{A_1, \ldots, A_m\}$ of 2×2 matrices is mortal if and only if there exist an integer k and integers $i_1, \ldots, i_k \in \{1, \ldots, m\}$ with $A_{i_1} \cdots A_{i_k} = \underline{0}$ and*

1. *$rank(A_{i_j}) = 2$ for $1 < j < k$,*
2. *$rank(A_{i_j}) < 2$ for $j \in \{1, k\}$.*

The set F can be divided into two disjoint sets $S = \{S_1, \ldots, S_k\}$ and $R = \{R_1, \ldots, R_n\}$ such that $rank(S_i) < 2$ for all $i \in \{1, \ldots, k\}$ and $rank(R_j) = 2$ for all $j \in \{1, \ldots, n\}$. We can assume, without loss of generality, that both S and R are non-empty. In fact, if $S = \emptyset$ then F is not mortal and, if $R = \emptyset$, then, by Lemma 2, we can decide whether the set is mortal simply considering the product of each pair of matrices in S. We can also assume, without loss of generality, that $rank(S_i) = 1$, otherwise $\underline{0} \in S$ and so F is trivially mortal. For each $S_i \in S$, we denote by ι_i and κ_i respectively the image and the kernel of S_i and by c_i (resp. k_i) the generator of ι_i (resp. κ_i) with integer and relative prime components (Lemma 1).

3 Reduction of the Problem

We have the following proposition which is a direct consequence of Lemma 2.

Proposition 1. *If $\underline{0} \notin F$, then F is mortal if and only if $K\iota_i = \kappa_j$ for some $i, j \in \{1, \ldots, k\}$ and $K \in \langle R_1, \ldots, R_n \rangle$.*

Proof. If there exist $i, j \in \{1, \ldots, k\}$ and $K \in \langle R_1, \ldots, R_n \rangle$ such that $K\iota_i = \kappa_j$, then $S_j K S_i = \underline{0}$. Conversely, suppose that F is mortal. Then, by lemma 2, $S_j K S_i = \underline{0}$ for some $K \in \langle R_1, \ldots, R_n \rangle$ and $i, j \in \{1, \ldots, k\}$ and so the statement follows. □

In the sequel we suppose that $R \subseteq GL(2,\mathbb{Z})$. With this assumption we can reduce the problem of seeing whether $K\iota_i = \kappa_j$ to the problem of checking whether $Kc_i^T = \pm k_j^T$, with $i,j \in \{1,\dots,k\}$ and $K \in \langle R_1,\dots,R_n\rangle$.

Lemma 3. *Let $K \in \langle R_1,\dots,R_n\rangle$. Then $K\iota_i = \kappa_j$ for $i,j \in \{1,\dots,k\}$ if and only if $Kc_i^T = \pm k_j^T$.*

Proof. Since $Kc_i^T = \lambda k_j^T$ for some $\lambda \in \mathbb{R}$, then $\lambda \in \mathbb{Z}$ because $k_j, c_i \in \mathbb{Z}^2$, $K \in M(2,\mathbb{Z})$ and $k_j = (z,t)$ with $gcd(z,t) = 1$. Therefore:

$$c_i^T = \lambda K^{-1} k_j^T$$

with $K^{-1} \in M(2,\mathbb{Z})$. If $c_i = (x,y)$ then λ divides x, λ divides y and so $\lambda = \pm 1$ since $gcd(x,y) = 1$. □

Lemma 4. *Let $c = (x,y) \in \mathbb{Z}^2$ with $gcd(x,y) = 1$. Then there exists $U \in GL(2,\mathbb{Z})$ such that $Ue_1 = c^T$.*

Proof. Since $gcd(x,y) = 1$, there are two integers $\lambda, \mu \in \mathbb{Z}$ such that $\lambda x + \mu y = 1$. If we put:

$$U = \begin{pmatrix} x & -\mu \\ y & \lambda \end{pmatrix}$$

then $Ue_1 = (x,y)^T$, $det(U) = 1$ and so $U \in GL(2,\mathbb{Z})$. □

Using lemma 4 we reduce the problem of checking whether $Kc_i^T = \pm k_j^T$, for some $K \in \langle R_1,\dots,R_n\rangle$ and for $i,j \in \{1,\dots,k\}$, to the problem of checking whether $U^{-1}KUe_1 = \pm U^{-1}k_j^T$ for some $U^{-1}KU \in \langle U^{-1}R_1U,\dots,U^{-1}R_nU\rangle$, where U is the matrix of lemma 4 when $c = c_i$.

Proposition 2. *Let $a,b \in \mathbb{Z}$ be relative prime. Then $K \in GL(2,\mathbb{Z})$ maps e_1 to $(a,b)^T$ if and only if K is of the form AT^λ where $\lambda \in \mathbb{Z}$,*

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

and

$$A = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \quad or \quad A = \begin{pmatrix} a & -c \\ b & -d \end{pmatrix}$$

with c,d integers satisfying $ad - bc = 1$.

Proof. From the hypothesis it follows trivially that $K = \begin{pmatrix} a & x \\ b & y \end{pmatrix}$, with $x,y \in \mathbb{Z}$ and $ay - bx = \pm 1$. Since $gcd(a,b) = 1$, there exist $c,d \in \mathbb{Z}$ such that $ad - bc = 1$. If $ay - bx = 1$, then a divides $(c - x)$ and b divides $(d - y)$. Let $\lambda \in \mathbb{Z}$ such that $c - x = \lambda a$ and $d - y = \lambda b$, hence

$$K = \begin{pmatrix} a & c \\ b & d \end{pmatrix} - \lambda \begin{pmatrix} 0 & a \\ 0 & b \end{pmatrix}.$$

Putting

$$A = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & a \\ 0 & b \end{pmatrix},$$

easy computations lead to $K = A(I - \lambda A^{-1}B) = AT^{-\lambda}$.

If $ay - bx = -1$, we prove the statement in an analogous way. □

4 Main Result

Now we are in position of demonstrating our main result. In fact, in order to prove whether the set F is mortal, we have to check if, for some integer $i, j \in \{1, 2, \ldots, k\}$, there is an integer λ such that $AT^{\lambda} \in \langle R_1, \ldots, R_n \rangle$, where A is a given matrix depending on i and j with $|\det A| = \pm 1$.

We strictly follow the technique used in [2], from which we recall for sake of completeness some notation and properties.

Let M be a finitely presented monoid with set of generators Σ. Then M is isomorphic to the quotient of a finitely generated free monoid Σ^* by some finitely generated congruence $\equiv$. A rational subset H of M is defined by some finite automaton $\mathcal{A}$ with input alphabet Σ in the following sense: $a \in H$ if and only if there exists a word w in the language recognized by $\mathcal{A}$ with $a = [w]_{\equiv}$.

Proposition 3. *Let G be a group such that $G \simeq \mathbb{Z}/p_1\mathbb{Z} * \ldots * \mathbb{Z}/p_n\mathbb{Z}$. Let $\mathcal{A}, \mathcal{B}$ be two automata that recognize respectively two rational subsets $H, K \subseteq G$. Then it is recursively decidable whether or not $H \cap K \neq \emptyset$*

Proof. The proof is a reformulation of the proof of Proposition 1 of [2]. As shown there, it is possible to construct the automata $\mathcal{A}_H, \mathcal{A}_K$ which recognize the sets of reduced words congruent to the words accepted by the automata defining respectively H and K. Then, since each word is equivalent to a unique reduced word, H and K have a nonempty intersection if and only if the languages $L(\mathcal{A}_H)$ and $L(\mathcal{A}_K)$ recognized respectively by $\mathcal{A}_H$ and $\mathcal{A}_K$ have nonempty intersection. □

Theorem 1. *Given a rational subset Q of matrices in $M(2, \mathbb{Z})$ and a matrix $A \in GL(2, \mathbb{Z})$, it is recursively decidable whether or not $\{AT^m \mid m \in \mathbb{Z}\} \cap Q \neq \emptyset$ where $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.*

Proof. This proof is analogous to the one of Theorem 1 in [2]. We reformulate the problem in the following way: we have a rational expression $\mathcal{R}(X_1, \ldots, X_n)$ over the set of symbols $X_i, i = 1, \ldots, n$ and a substitution φ assigning a matrix of $M(2, \mathbb{Z})$ to each symbol X_i. We want to establish, for the given rational subset $\{AT^m \mid m \in \mathbb{Z}\}$, whether or not:

$$\{AT^m \mid m \in \mathbb{Z}\} \cap \varphi(\mathcal{R}(X_1, \ldots, X_n)) = \emptyset \tag{1}$$

We prove the thesis by successive simplifications:

Claim 1. Without loss of generality we may assume that $A = I$. Indeed, if X is a new symbol and we extend φ so that $\varphi(X) = A^{-1}$, then the condition (1) is equivalent to the condition $\{T^m \mid m \in \mathbb{Z}\} \cap \varphi(X \cdot \mathcal{R}(X_1, \ldots, X_n)) = \emptyset$.

Claim 2. Without loss of generality we may assume that the determinant of all X_i's is equal to 1 or -1 because $det(T^m) = 1$ for all $m \in \mathbb{Z}$.

Claim 3. Without loss of generality we may assume that the determinant of all X_i's is equal to 1. The argument is the same of Claim 3 in Theorem 1 of [2] since $det(T^m) = 1$ for all $m \in \mathbb{Z}$.

The previous three claims prove that we can start from a rational expression $\mathcal{R}$ and a morphism φ assigning a matrix of $SL(2,\mathbb{Z})$ to each symbol X_i and so we can suppose that $\varphi(\mathcal{R})$ is a rational subset of $SL(2,\mathbb{Z})$. Since

$$SL(2,\mathbb{Z})/\{I,-I\} = PSL(2,\mathbb{Z}) \simeq \mathbb{Z}/2\mathbb{Z} * \mathbb{Z}/3\mathbb{Z},$$

by proposition 3 we can verify whether the image of $\{T^m \,|\, m \in \mathbb{Z}\} \cap \varphi(\mathcal{R})$ in $PSL(2,\mathbb{Z})$ is non-empty. Then, in order to prove if $\{T^m \,|\, m \in \mathbb{Z}\} \cap \varphi(\mathcal{R}) \neq \emptyset$ in $SL(2,\mathbb{Z})$, we have to lift the ambiguity between T^m and $-T^m$. An easy computation show that

$$T^m = \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix}$$

and so, denoting by ψ the morphism from $SL(2,\mathbb{Z})$ into $M(2,\mathbb{Z}/3\mathbb{Z})$ which puts the entries of a matrix module 3, we obtain that:

$$\{T^m \,|\, m \in \mathbb{Z}\} \subseteq H = \left\{ A \in SL(2,\mathbb{Z}) \,|\, \psi(A) = \begin{pmatrix} 1 & i \\ 0 & 1 \end{pmatrix} \bmod 3, i = 0,1,2 \right\}$$

while $-T^m \notin H$. The set:

$$Q = \left\{ W \in \{X_1,\ldots,X_n\}^* | \psi(\varphi(W)) = \begin{pmatrix} 1 & i \\ 0 & 1 \end{pmatrix} \bmod 3, i = 0,1,2 \right\}$$

is a rational set since it is the preimage by the morphism $\varphi \circ \psi$ of the finite set:

$$\left\{ \begin{pmatrix} 1 & i \\ 0 & 1 \end{pmatrix} \bmod 3 \,|\, i = 0,1,2 \right\}$$

Thus $\{T^m \,|\, m \in \mathbb{Z}\} \cap \varphi(\mathcal{R}(X_1,\ldots,X_n)) = \emptyset$ if and only if

$$\{T^m \,|\, m \in \mathbb{Z}\} \cap \varphi(\mathcal{R}(X_1,\ldots,X_n) \cap Q) = \emptyset,$$

where $\mathcal{R}(X_1,\ldots,X_n) \cap Q$ is rational since it is the intersection of two rational sets. □

It would be interesting to see if the mortality problem is decidable for any set of $n \times n$ integer matrices whose determinants are $0, \pm 1$.

Acknowledgment

The authors thank professor Stuart Margolis for suggesting the problem to the second named author during his stay at Bar Ilan University with the support of AutMathA. They are also grateful to professor Alessandra Cherubini for her precious advices.

References

1. Bournez, O., Branicky, M.: On the mortality problem for matrices of low dimensions. TCS 35(4), 433–488 (2002)
2. Choffrut, C., Karhumäki, J.: Some decision problems on integer matrices. RAIRO/ITA 39(1), 125–132 (2005)
3. Halava, V., Harju, T.: Mortality in matrix semigroups. Amer. Math. Monthly 108(7), 649–653 (2001)
4. Krom, M., Krom, M.: Recursive solvability of problems with matrices. zwitschr. f. math. Logik und Grundlagen d. Math. 35, 437–442 (1989)
5. Krom, M., Krom, M.: More on mortality. Amer. Math. Monthly 97, 37–38 (1990)
6. Lisitsa, A., Potapov, I.: Membership and reachibility problems for row-monomial transformations. In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 623–634. Springer, Heidelberg (2004)
7. Paterson, M.S.: Unsolvability in 3×3 matrices. Stud. Appl. Math. 49, 105–107 (1970)

Element Distinctness and Sorting on One-Tape Off-Line Turing Machines

Holger Petersen

Univ. Stuttgart, FMI
Universitätsstraße 38
D-70569 Stuttgart
petersen@informatik.uni-stuttgart.de

Abstract. We investigate off-line Turing machines equipped with a two-way input-tape and one work-tape.

It is shown that the Element Distinctness Problem (EDP) for m binary strings of length $\ell = O(m/\log^2 m)$ can be solved in time $O(m^{3/2}\ell^{1/2})$ and space $O(m^{1/2}\ell^{1/2})$ on a nondeterministic machine. This is faster than the best sorting algorithm on the computational model and optimal if time and space are considered simultaneously.

For deterministic machines we give an optimal algorithm that can sort m binary strings consisting of ℓ bits each in $O(m^{3/2}\ell)$ steps, provided that $\ell = O(m^{1/4})$. By modifying the solution we obtain the time bound $O(m^{3/2}\ell)$ and the space bound $O(m^{1/2}\ell^2)$ for the EDP.

1 Introduction

The Element Distinctness Problem (EDP) is a well known benchmark task for a variety of models of computation ranging from single-tape Turing machines [1,8,9,11] to Quantum Computers [3]. The input for the EDP (sometimes called Element Uniqueness Problem) is a multiset of m binary strings of length ℓ each. Here ℓ is determined by some function of m. Of course $\ell \geq \log m$ and typically $\ell = \Theta(\log m)$. The question to be answered is, whether every element in the input occurs exactly once.

The EDP is closely related to sorting, since on most computational models an efficient reduction from EDP to sorting is easily accomplished. Therefore lower bounds for EDP generalize to sorting, while EDP as a decision problem might be easier to analyze. For these reasons it is interesting to study upper and lower bounds for EDP and compare them to the most efficient sorting algorithms on different computational models.

On single-tape Turing machines (no separate input tape) several authors have investigated the complexity of the EDP [8,9,11] until it was completely determined in [1]. We continue this research by moving on to a more powerful model of computation, the off-line Turing machine. For this type of machine lower and upper bounds on the complexity of sorting are known in the deterministic and nondeterministic mode of operation [4,12]. In fact off-line Turing machines are among the most powerful models of computation for which non-trivial lower

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 406–417, 2008.

bounds on natural problems could be shown. Upper bounds indicate to what extent our techniques for establishing lower bounds can be improved.

This paper is organized as follows. In Section 2 the necessary definitions are given. We present a nondeterministic solution of EDP and a matching lower bound in Section 3. There are at least two reasons for considering these results first. A nondeterministic Turing machine can guess very useful information that a deterministic machine would have to tediously compute (if possible at all). Therefore the algorithm tends to be simpler than a deterministic counterpart. The second aspect is that a solution to EDP can reject immediately once it discovers that an element appears twice in the input, while a sorting algorithm has to handle multiple copies of an element. Surprisingly these copies make the sorting method from [12] as well as our deterministic solution rather complex.

It should be noted that our nondeterministic solution for EDP is faster than the best sorting algorithm [12] for a wide range of relations between m and ℓ.

Section 4 contains a time optimal deterministic sorting algorithm and a modification that solves the EDP. The algorithm settles the question raised by Wiedermann [12], whether there exists a deterministic sorting method of an efficiency comparable to his nondeterministic solution for $\ell > \log m$. The strategy of the algorithm from [12] is to nondeterministically extract groups of strings from the input which are close in the sorted sequence. In a second phase these strings a copied to their final positions, again making use of nondeterminism. We replace the nondeterministic steps by computing strings that partition the input in a suitable way and by finding their positions deterministically.

In Section 5 some open problems are outlined.

2 Preliminaries

Formally, the EDP can be defined as a language recognition problem:

$$\mathrm{EDP}(m,\ell) = \{x_1\#x_2\#\cdots\#x_m \mid x_i \in \{0,1\}^\ell, x_i \neq x_j \text{ for } i \neq j\}.$$

Notice that the length of an input for the EDP has size $O(m\ell)$.

Using an analogous notation, a solution of the sorting problem receives as input $x_1\#x_2\#\cdots\#x_m$ with $x_i \in \{0,1\}^\ell$ and has to produce as output $x_{\pi(1)}\#x_{\pi(2)}\#\cdots\#x_{\pi(m)}$ with the property that $x_{\pi(i)} \leq x_{\pi(i+1)}$ for $1 \leq i < m$. Here π is a permutation of the set $\{1,\ldots,m\}$.

The computational models investigated are deterministic and nondeterministic off-line Turing machines. These machines are equipped with one work-tape and a separate two-way input-tape. The machines cannot write on the input-tape or move the head on this tape off the input. This is the notion of off-line operation from [12]. We point out that other definitions exist in the literature.

The output of a computation is written onto the work-tape.

3 Nondeterministic Turing Machines

On most models of computation the EDP easily reduces to sorting. For Turing machines the reduction requires some care, since comparing m strings of length

ℓ written on the work-tape to an adjacent string bit by bit (in a deterministic way and without using the input-tape) requires time $\Omega(m\ell^2)$. Making use of nondeterminism and the storage capacity of the input-tape we can reduce this overhead to $O(m\ell)$, which is the time required to scan the input anyway.

Observation 1. *Let $t(m,\ell)$ be an upper time bound for sorting m binary strings of length ℓ on a nondeterministic off-line Turing machine with two-way input. Then EDP(m,ℓ) can be accepted in time $t(m,\ell)+O(m\ell)$ on the same machine model.*

Proof. In the first stage a machine M deciding the EDP applies the sorting algorithm. Then it verifies that neighboring strings are different making use of its input tape as a unary counter. The count 0 is represented by the input head being on the left end-marker. Increment and decrement correspond to right and left movements.

Let x_i and x_{i+1} differ at position j. Then M starts from the first symbol of x_i on the work-tape counting on the input tape until it guesses that it has reached symbol j. It records the bit read and moves on the work-tape to the first symbol of x_{i+1}. By decrementing it locates symbol j and verifies that it differs from the recorded bit.

Each comparison can be done on time $O(\ell)$, leading to the claimed additional $O(m\ell)$ steps. □

We can now derive from Wiedermann's bound $O(m^{3/2}\ell)$ on sorting that EDP is at most of the same complexity.

In the reduction above we made use of the fact that in order to determine inequality of strings a difference of one bit is enough. Now we exploit this observation a little further and show that by using a suitable technique we can get a better time complexity than by reducing EDP to sorting.

In the following we treat the strings in the input of the EDP as integers. We first adapt Lemma 6 of [1] to the notation of the present paper.

Lemma 1. *For every sufficiently large value of m, and for every set of m distinct binary strings of length $\ell \le m$ there is a prime $r \le m^4$, such that no two elements are congruent modulo r.*

Proof. By the Prime Number Theorem there are $\Omega(m^4/\log m)$ primes less or equal than m^4. For any pair of values i,j of distinct elements in the input every prime p such that i and j are congruent modulo p divides $|i-j|$. Hence there are at most $\ell \le m$ such primes for each pair. Omitting these at most m^3 numbers leaves at least one prime r with the desired property for sufficiently large m. □

Theorem 1. *The EDP(m,ℓ) for length $\ell = O(m/\log^2 m)$ can be accepted in time $O(m^{3/2}\ell^{1/2})$ and space $O(m^{1/2}\ell^{1/2})$ on a nondeterministic off-line Turing machine with two-way input.*

Proof. The strategy of the algorithm is to partition the elements from the input into disjoint blocks of similar size. For each block a space efficient perfect hashing

function is guessed and it is verified that all elements in one block are mapped to different hash values. This establishes that all elements are distinct.

We first describe the computation of the Turing machine M solving the EDP on a positive instance. Machine M guesses a partition of the elements into blocks of at most $q = \lfloor \sqrt{m\ell} \rfloor$ elements, such that each element from block $i+1$ is greater than each element from block i. In order to do this, it generates two elements such that at most q elements would lie between them in the *sorted* sequence of elements. In each iteration M selects these elements. From now on we will concentrate on one such set S of size q.

First M guesses a prime r of size polynomial in m according to Lemma 1. Observe that r can be represented in $O(\log m)$ bits. By reducing each element x to $x' = x \bmod r$ in time $O(\ell \log m)$ the machine can perform all following arithmetics with numbers consisting of $O(\log m)$ bits.

A second prime p with $r < p \leq 2r$ is guessed in binary notation. Now M uses the hashing technique based on the corollaries from [5] with universe $U = \{1, \ldots, p-1\}$. By $B(q, S, k, j)$ we denote the number of times j appears as the image of an $x \in S$ when the function $x \mapsto (kx \bmod p) \bmod q$ is applied. First M prepares a two-level hashing scheme by guessing a $k < p$ such that

$$\sum_{j=1}^{q} B(q, S, k, j)^2 < 3q.$$

A k with this property exists by Corollary 1 of [5]. Then M allocates q buckets consisting of $B(q, S, k, j)^2$ binary flags each, separated by markers (length $B(q, S, k, j)^2$ of bucket S_j is guessed).

For the second level the space efficient rehashing of Slot and van Emde Boas is employed [10], which we outline below.

Set $s_j = |S_j|$, call $k' \in U$ *good for bucket* j if $x \mapsto (k'x \bmod p) \bmod 2s_j^2$ is one-to-one on S_j. Let C be the set of buckets. By Corollary 4 from [5] at least half of the k's is good for each bucket in C, therefore there is a k'_1 that is good for at least half of the buckets. Some k'_2 is good for at least half of the remaining buckets etc. until all buckets are covered. From these $1 + \log q$ good elements a table $k'_1, \ldots, k'_{1+\log q}$ is formed in space $O(\log^2 m)$.

The k'_i are assigned to buckets by $1 + \log q$ tables of binary flags: in the first table all buckets are marked for which k'_1 is a good multiplier in the sense defined above, in the second table all remaining buckets are marked for which k'_2 is good etc. Observe that each table occupies at most half of the space of the previous one leading to a space bound $O(q)$.

Computation of the hash-values: For every element $x \in S$ hash function $h(x') = (kx' \bmod p) \bmod q$ is evaluated. Bucket $j = h(x')$ can be marked in $O(q)$ time with the help of the fast counting technique explained below.

A good k'_{i_j} for bucket j is found by first checking the flags for k'_1. If the j-th flag is false, M counts the number of buckets before j which are not covered by k'_1. This is again done with the help of the fast counting technique. Then

$h_j(x') = (k'_{i_j} x' \bmod p) \bmod 2s_j^2$ is computed, flag $h_j(x')$ of bucket j is tested (M rejects if it is true) and the flag is set.

The time required for arithmetics over all iterations is

$$O(m\ell \log m) = O(m\sqrt{\ell}\sqrt{m/\log^2 m}\log m) = O(m^{3/2}\ell^{1/2}).$$

This concludes the description of M's computation on a positive instance. It should be clear that by choosing suitable partitions and hashing functions each positive instance of EDP can be accepted.

If there are duplicates in the input, then it might be impossible to guess a partitioning into blocks of the intended size. This can be detected by M, since it can compare the number of selected elements and q. Notice that M does not have to establish that a partitioning is impossible, since an input is rejected if all guesses fail. If a block contains two identical elements, then after applying the transformations and hash-functions the same values will be computed. Thus M will detect the collision.

Fast counting technique: The goal is to use the input tape as an auxiliary counter without moving the input head to one of the end-markers, since this would take too much time.

Let the input head of M be positioned on the marker before element x. Copies of x are generated by M on a track parallel to the stored information on the work-tape in time $O(q)$. If x is closer to the left end of the input tape, then an increment operation is simulated by a move to the right and decrement is simulated by a move to the left. If x is close to the right, the directions are interchanged. M can guess which case to choose. In this way M can count up to $m\ell/2$, which is sufficient.

The test for zero is done in two stages: first M checks, whether it reads a marker on the input tape. If this is the case, it compares the element following the marker and a copy of x on the work-tape. If the work-tape head is not on the first symbol of such a copy, M marks its current position, moves its head to the first symbol, and after the comparison returns to the initial position. This test would lead to incorrect results if x would appear more than once on the input tape. Therefore whenever the counter is incremented and afterwards a marker is read on the input tape, M compares the element following that marker and x and immediately rejects if they are identical.

The time complexity of all counting processes is $O(q)$ in the algorithm above, since each comparison of complexity $O(\ell)$ is preceded by ℓ increment or decrement operations of constant complexity. For linear complexity it is important that x needs to be copied on the parallel track only once for all counting operations required by x. □

Remark 1. The result shows that on this model of computation EDP is easier concerning its time complexity than sorting at least for certain relations between m and ℓ.

Next we show that it is not possible to simultaneously improve the time and space requirements in the previous algorithm.

Theorem 2. *Any nondeterministic off-line Turing machine accepting the problem EDP(m,ℓ) for length $\ell \geq 2\log m$ simultaneously within space $s(m,\ell)$ and time $t(m,\ell)$ satisfies*

$$s(m,\ell)t(m,\ell) = \Omega(m^2\ell).$$

Proof. Recall that $N^1(f)$ is the nondeterministic two-party communication complexity of the two-place function f for the value 1. One can define this measure as the minimum cost in terms of bits exchanged in a protocol for two players evaluating f which are allowed to take nondeterministic steps. An equivalent definition based on covers can be found in [7]. By DISJ(n,u) we denote the boolean function of disjointness of two sets of n numbers each chosen from a universe of $u \geq 2n$ elements.

Theorem 2.11 of [7] shows that

$$N^1(\mathrm{DISJ}(n,u)) = \Omega\left(\frac{\log\binom{u}{n}}{\log u}\right).$$

A protocol for DISJ$(m/4, 2^\ell - m/2)$ can be obtained from a nondeterministic off-line Turing machine accepting the EDP in time $t(m,\ell)$ and space $s(m,\ell)$ by running it on an input consisting of Alice's $m/4$ numbers, then a "desert" of $m/2$ fixed numbers and finally Bob's $m/4$ numbers. As long as the machine does not enter Bob's numbers with its input head, Alice does the simulation. When this happens, she transmits all of the work-tape contents to Bob. He then continues until the machine enters Alice's region of the input tape and so on. If the machine accepts, then the protocol accepts. Since a transfer of the tape contents happens after at least $m\ell/2$ (length of the desert) steps, we have that the protocol transmits $O(t(m,\ell)s(m,\ell)/(m\ell))$ bits. This gives the claimed bound. □

We remark that Karchmer [6] has shown $t^2(m,\log m)s(m,\log m) = \Omega(m^3)$ for the stronger model of non-uniform nondeterministic Turing machines with several input-heads.

4 Deterministic Turing Machines

The main portion of this section is devoted to a deterministic sorting algorithm. Since it is considerably more complex than the nondeterministic solution of the EDP presented in the previous section, we first describe the computation of some auxiliary information.

Definition 1. *Let M be a multiset of strings of length ℓ and $m = |M|$. The k-th* radian x_k *is a string with the properties*

$$|\{y \in M \mid y < x_k\}| < k \cdot \lfloor\sqrt{m}\rfloor$$

and

$$|\{y \in M \mid y \leq x_k\}| \geq k \cdot \lfloor\sqrt{m}\rfloor,$$

where $\{y \in M \mid y < x_k\}$ and $\{y \in M \mid y \leq x_k\}$ are multisets.

Notice that $x_i = x_j$ is possible for $i \neq j$, since M is not required to be a set. Intuitively, x_k is an element of M whose position in the sorted sequence is $k \cdot \lfloor\sqrt{m}\rfloor$.

Lemma 2. *There are less than $\sqrt{m}+2$ radians and for each radian x_k holds $x_k \in M$.*

Proof. For $0 \leq m \leq 4$ the first statement clearly holds. If $m \geq 5$ we have

$$\begin{aligned}(\sqrt{m}+2) \cdot \lfloor\sqrt{m}\rfloor &\geq (\sqrt{m}+2) \cdot (\sqrt{m}-1) \\ &= m + \sqrt{m} - 2 \\ &> m\end{aligned}$$

and the inequality $|\{y \in M \mid y \leq x_k\}| \geq k \cdot \lfloor\sqrt{m}\rfloor$ cannot be satisfied for $k \geq \sqrt{m}+2$.

If $x_k \notin M$, then $|\{y \in M \mid y < x_k\}| = |\{y \in M \mid y \leq x_k\}|$ contradicting the definition of a radian. □

Lemma 3. *A deterministic off-line Turing machine with two-way input receiving m binary strings of length ℓ each can compute all radians and store them on its work-tape in time $O(m^{3/2}\ell + m\ell^3)$.*

Proof. The computation is split into phases, where in each phase ℓ radians are computed.

Assume that the machine T computing the radians has determined all radians up to $x_{(k-1)\ell}$ (this is vacuously true for $k = 1$). Now T starts phase k for the computation of radians $x_{(k-1)\ell+1}$ up to $x_{k\ell}$ with two variables $s = 0$ and $t = 2^\ell - 1$ and performs a binary search for $x_{k\ell}$ in the space of binary sequences of length ℓ (we identify these sequences with the numerical values in binary notation). It maintains the invariant $s \leq x_{k\ell} \leq t$, which clearly holds initially. Then it computes $z = \lfloor(s+t)/2\rfloor$, compares each string y in the input with z and counts those y for which $y \leq z$. In order to do this efficiently the counter is stored on a track along with z and the update is done when T resets its head to the first symbol of the binary encoding of z. If the multiset of $y \leq z$ satisfies $|\{y \mid y \leq z\}| < k\ell \cdot \lfloor\sqrt{m}\rfloor$ machine T replaces s with $z+1$, otherwise it replaces t with z. In the former case the radian $x_{k\ell}$ has to be greater than z, since $|\{y \in M \mid y \leq x_k\}| \geq k \cdot \lfloor\sqrt{m}\rfloor$. Therefore $z+1 \leq x_{k\ell} \leq t$ and the invariant is maintained by replacing s with $z+1$. In the other case $x_{k\ell}$ cannot be greater than z and so $s \leq x_{k\ell} \leq z$. The process stops when $s = t = x_{k\ell}$. The number of iterations is bounded by ℓ, since after the i-th iteration the i most significant bits of s and t are identical, the $\ell - i$ least significant bits of s are 0 and those of t are 1. These properties hold before the first iteration and are maintained since the $\ell - i$ least significant bits of z in iteration i are 1. The time complexity of this part of each phase is $O(m\ell^2)$ (the input of length $O(m\ell)$ is read ℓ times, the other operations can be done in time $O(\ell)$ per string).

Now in one pass over the input the machine computes $d = \min(|\{y \mid y \leq x_{(k-1)\ell}\}|, k\ell \cdot \lfloor\sqrt{m}\rfloor)$. Then it generates $d - (k-1)\ell \cdot \lfloor\sqrt{m}\rfloor$ copies of $x_{(k-1)\ell}$. In

another pass over the input, T copies all those y for which $x_{(k-1)\ell} < y < x_{k\ell}$ onto its work-tape. By definition of $x_{(k-1)\ell}$ and $x_{k\ell}$ these are less than $\ell \cdot \lfloor\sqrt{m}\rfloor$ strings. Finally T generates copies of $x_{k\ell}$ until $\ell \cdot \lfloor\sqrt{m}\rfloor$ strings are on the work-tape.

Performing a similar binary search as above, T determines $x_{(k-1)\ell+1}$ to $x_{k\ell-1}$. The main difference is that copies of s, t and z are stored on separate tracks underneath each y. The initialization of s and t is trivial. The computation of z can be done in one pass over the strings in time $O(\sqrt{m}\ell^2)$ ($\sqrt{m}\ell$ strings of ℓ bits each). Along with the computation, T can compare z with each string and count those y such that $y \leq z$ making use of its input-tape as a unary counter. If $d + |\{y \mid y \leq z\}| < ((k-1)\ell + j) \cdot \lfloor\sqrt{m}\rfloor$ then T replaces s with $z+1$, otherwise it replaces t with z. In this way T determines $x_{(k-1)\ell+j}$ for $1 \leq j < \ell$. The number of radians computed per phase is ℓ, each computation has to determine ℓ bits leading to a time bound $O(\sqrt{m}\ell^4)$ per phase.

There are $O(\sqrt{m}/\ell)$ phases of complexity $O(m\ell^2 + \sqrt{m}\ell^4)$ each leading to the claimed bound $O(m^{3/2}\ell + m\ell^3)$. □

Theorem 3. *Sorting m binary strings of length ℓ each separated by marker symbols can be done in time $O(m^{3/2}\ell)$ on a deterministic off-line Turing machine with two-way input if $\ell = O(m^{1/4})$.*

Proof. The strategy of the algorithm is to partition the elements into blocks using the radians computed according to Lemma 3. The elements from one block are then moved to their final position by a counting technique.

Without loss of generality we may assume $\ell \geq \log m$, since otherwise Wiedermann's procedure of complexity $O(m\ell 2^{\lceil \ell/2 \rceil})$ [12, Theorem 7] can be employed. The assumption $\ell \geq \log m$ justifies the use of binary encoded counters in the range from 0 to m stored on separate tracks along with copies of the input strings.

The machine T sorting the input first computes all radians according to Lemma 3. Then it sets up a sequence of m slots containing several fields, where slot j is composed of:

- t_j a field of ℓ bits which will eventually contain the j-th string of the sorted sequence,
- a_j an auxiliary field of ℓ bits,
- p_j a single bit,
- and g_j a boolean flag marking those slots containing the final value in field t_j.

Initially all g_j are set to false and all t_j to 0^ℓ.

Now T computes $k = \lfloor\sqrt{m}\rfloor$ in binary and uses its input-head as a unary counter in order to mark the positions of the radians in the sequence of slots by setting their fields t_j to 1^ℓ. Then T transfers each radian x_i to its position by finding x_i on the input-tape, moving its work-tape head to the first slot j with $t_j = 1^\ell$ and $g_j =$ false, and copying x_i from the input-tape onto t_j. Field g_j is set to true. A second copy of x_i is stored in the auxiliary field of the slot of the previous radian (this will not work for x_1, for which a second copy is placed before the sequence).

The radians define blocks of $k-1$ empty slots, where eventually strings between the two neighboring radians will be stored. The j-th remaining slot in block i has to accommodate a record of values, where the meaning of each entry after completion of the computation for the block will be:

- y_j is a string occurring in the input with $x_i \leq y_j \leq x_{i+1}$,
- c_j is the number of times y_j occurs in the input,
- d_j is the number of *different* strings smaller than y_j and larger than x_i in the input,
- e_j is the number of strings (counting duplicates) smaller than y_j and larger than x_i in the input,
- and f_j is a boolean flag.

Notice that the numbers c_j, d_j, and e_j can be stored in ℓ bits each, since $\ell \geq \log m$.

Stage i which determines strings properly between x_i and x_{i+1} (we set $x_0 = 0^\ell$) starts with an empty block, which is indicated by flag f_j being initialized to false for all j in the current block.

In one pass over the input, T compares every string y with x_i and x_{i+1} stored in the first slot of the block. If y falls within this range, T compares y to all strings already stored in the block. If y occurred previously as y_j, then the count c_j is incremented. If y occurs for the first time, then the new slot h is initialized by setting $y_h = y$, $c_h = 1$, and $f_h = \text{true}$.

After storing all strings of the block the values d_j and e_j are determined in two additional passes over the input. All counters are initialized with 0. For computing the d_j each y in the block is located on the input-tape in one scan from left to right and then all other entries of the block are compared to y. For those greater the counter d_j is incremented. When computing the e_j all strings in the input between x_i and x_{i+1} are taken into account and the counters are updated in the same way as for the d_j.

Now the fields t_j in the current block are filled with the appropriate values. Notice that the value $d = |\{y \mid y \leq x_i\}| - i \cdot k$ (which can be computed by T in another scan of the input) determines the number of copies of radian x_i that occur before the next larger string. Therefore T marks the first $\max(d, k)$ slots (e.g. by writing a special value into their a-fields) using its input-head as a counter and then copies x_i onto their t-fields. It marks them as final values by setting their g-fields to true.

Next T has to move the values y_j to their proper positions. It again uses its input-head as a counter and in turn for every non-empty slot j of the block marks the $e_j + 1$-st non-final field t_h. Then it stores c_j as the position of its input-head and copies the least significant bit of d_j into the fields $p_h, \ldots, p_{h+c_j-1}$.

Now T scans all y_j starting with the *last* one. It locates y_j on the input-tape searching *backwards*. If it has found y_j it uses the input-tape for copying y_j into all a-fields of the current block. This copying is a preparatory step for a deterministic variant of the fast counting technique described in Section 3. Then T positions its input-head on the marker symbol after y_j on the input-tape. It

starts to decrement d_j until it reaches zero and for every step moves its input-head one position to the left. If it encounters a marker, it checks whether the string on the input-tape ending at this marker is equal to y_j. If it is, it restores d_j (by reversing the counting process) and moves the input-head to the previous occurrence of y_j. This revision may be repeated several times. If the count is completely stored, T moves its work-head onto the first non-final t-field and starts to look for the correct position of y_j. For doing this it decrements the count stored as the position of its input-head for every change of the value of consecutive p-fields, not however counting sequences of equal p-fields belonging to final values. If the count reaches zero, T has found the proper position, copies y_j into the t-fields of all slots with the same p-value as the first one, and marks these slots as final. The test for zero is done with the help of the copies of y_j stored in the a-field of every slot. Then T decrements the values d_h for all $y_h > y_j$ to the left of y_j on the work-tape and continues the process until all y_j have been copied.

We have to argue, that there is some occurrence of y_j for which the counting process can be completed. Such an occurrence is the first one, since before the first occurrence of y_j there will be at least as many occurrences of other strings as there are different strings that still have to be moved to their proper positions. The latter number bounds the maximum count to be stored.

If after the distribution of the y_j there are still non-final t-fields, then T fills them with x_{i+1}.

We will now analyze the time complexity of the algorithm. Computing the radians according to Lemma 3 is possible in time $O(m^{3/2}\ell + m\ell^3)$. For $\ell = O(m^{1/4})$ this time bound is $O(m^{3/2}\ell)$.

The initialization of all fields can be completed in $O(m\ell)$ steps.

Copying a radian x_i is done by finding x_i on the input-tape in $O(m\ell)$ operations, moving the work-tape head to the appropriate slot over $O(m\ell)$ tape cells, and copying x_i from the input-tape in time $O(\ell)$. Transferring all $O(\sqrt{m})$ radians is therefore possible in time $O(m^{3/2}\ell)$.

Since each stage (except possibly the last) processes at least $\sqrt{m}$ strings from the input, the number of stages is $O(m/\sqrt{m}) = O(\sqrt{m})$. In order to derive the claimed time bound $O(m^{3/2}\ell)$ it therefore suffices to show that each stage can be completed in $O(m\ell)$ steps.

Comparing all m strings in the input with a fixed number of strings on the work-tape can be done in time $O(m\ell)$. Similarly comparing $O(\sqrt{m})$ strings to $O(\sqrt{m})$ strings stored in a block is possible in $O(m\ell)$ steps. This shows that filling the slots in one block and updating the counters does not exceed the time-bound.

Copying the radian into $O(\sqrt{m})$ slots takes $O(\sqrt{m}\ell^2)$ steps, which is $O(m\ell)$.

Preparing the p-fields requires $O(\sqrt{m})$ movements of the input-head over $k\ell = O(\sqrt{m}\ell)$ symbols. Here and in the sequel T has to count in binary. Notice that counting up to (or down from) some number n takes time $O(n)$.

Since T searches for the y_j stored in one block in one scan of the input-tape, it reads every symbol a finite number of times. The test for zero is carried out

only at marker symbols, therefore this comparison of time complexity $O(\ell)$ occurs only after ℓ steps with constant cost, resulting in constant amortized complexity. Updating the counters is possible in $O(\sqrt{m}\ell)$ steps per string, leading again to the bound $O(m\ell)$. □

The lower bound on matrix transposition from [4] yields the time-bound

$$\Omega\left(m^{3/2}\ell \Big/ \left\lceil \sqrt{(\log m)/\ell} \right\rceil\right)$$

on sorting for ℓ large enough to store binary addresses. Therefore Theorem 3 is asymptotically optimal for a wide range of lengths of strings to be sorted. If ℓ is small, Wiedermann's sorting algorithm [12, Theorem 7] with time bound $O(m\ell 2^{\lceil \ell/2 \rceil})$ is more efficient.

By modifying the sorting algorithm from the proof of Theorem 3 we obtain:

Theorem 4. *The problem EDP(m, ℓ) with $\ell = O(m^{1/4})$ can be solved by a deterministic off-line Turing machine with two-way input in time $O(m^{3/2}\ell)$ and space $O(m^{1/2}\ell^2)$.*

Proof sketch. The first stage of the algorithm showing Theorem 3 computes elements partitioning the input elements in time $O(m^{3/2}\ell)$ using $O(m^{1/2}\ell^2)$ space.

The second stage actually sorting the input works in $\Theta(m^{1/2})$ phases, where each phase generates a sorted subsequence consisting of $\Theta(m^{1/2})$ elements (space $\Theta(m^{1/2}\ell)$ per phase).

We modify the second stage so that each phase overwrites the elements stored by the previous one. This reduces the overall space usage to $\Theta(m^{1/2}\ell)$. Along with storing an element, the machine compares it to every other element previously selected and rejects if it discovers two equal elements. Since the time complexity of the second stage is $O(m^{3/2}\ell)$, we obtain the claimed bounds. □

In a similar way as in Theorem 2 we obtain:

Proposition 1. *Any deterministic off-line Turing machine accepting the problem EDP(m, ℓ) for length $\ell \geq 2 \log m$ simultaneously within space $s(m, \ell)$ and time $t(m, \ell)$ satisfies*

$$s(m, \ell)t(m, \ell) = \Omega(m^2\ell^2).$$

This product of time and space complexity can be achieved by copying each element onto the work-tape and comparing it with every other element in time $O(m^2\ell)$ and space $O(\ell)$. The time-efficient algorithm from Theorem 4 is however worse by a factor ℓ than the lower bound from Proposition 1.

5 Open Problems

It remains open whether our algorithms for EDP are time-optimal. For deterministic machines there is even a gap between upper and lower bound for the time-space product. An old problem related to the results of the present paper is, whether a variant of EDP can be solved in linear time by a nondeterministic Turing machine with several work-tapes [2].

The deterministic algorithm from Theorem 3 is asymptotically optimal for a wide range of lengths of strings to be sorted. For small ℓ Wiedermann's solution of time complexity $O(m\ell 2^{\lceil \ell/2 \rceil})$ is superior. In fact, it is optimal for constant ℓ, but it remains an open question whether it can be improved for slowly growing, non-constant ℓ.

Acknowledgments. The author is grateful to the referees for suggesting several corrections and improvements of the presentation.

References

1. Ben-Amram, A.M., Berkman, O., Petersen, H.: Element distinctness on one-tape Turing machines. Acta Informatica 40, 81–94 (2003)
2. Book, R.V., Greibach, S.: Quasi realtime languages. Mathematical Systems Theory 4, 97–111 (1970)
3. Buhrman, H., Dürr, C., Heiligman, M., Høyer, P., Magniez, F., Santha, M., de Wolf, R.: Quantum algorithms for element distinctness. SIAM Journal on Computing 34, 1324–1330 (2005)
4. Dietzfelbinger, M., Maass, W., Schnitger, G.: The complexity of matrix transposition on one-tape off-line Turing machines. Theoretical Computer Science 82, 113–129 (1991)
5. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with $O(1)$ worst case access time. Journal of the Association for Computing Machinery 31, 538–544 (1984)
6. Karchmer, M.: Two time-space tradeoffs for element distinctness. Theoretical Computer Science 47, 237–246 (1986)
7. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)
8. López-Ortiz, A.: New lower bounds for element distinctness on a one-tape Turing machine. Information Processing Letters 51, 311–314 (1994)
9. Petersen, H.: Bounds for the element distinctness problem on one-tape Turing machines. Information Processing Letters 81, 75–79 (2002)
10. Slot, C., van Emde Boas, P.: The problem of space invariance for sequential machines. Information and Computation 77, 93–122 (1988)
11. Szepietowski, A.: The element distinctness problem on one-tape Turing machines. Information Processing Letters 59, 203–206 (1996)
12. Wiedermann, J.: Optimal algorithms for sorting on single-tape Turing machines. In: van Leeuwen, J. (ed.) Algorithms, Software, Architecture, Proceedings of the IFIP 12th World Computer Congress, Madrid, Spain, vol. I, pp. 306–314. Elsevier Science Publishers, Amsterdam (1992)

Improved Bounds for Range Mode and Range Median Queries

Holger Petersen

Univ. Stuttgart
FMI
Universitätsstr. 38
D-70569 Stuttgart
petersen@informatik.uni-stuttgart.de

Abstract. We investigate the following problem: Given a list of n items and a function defined over lists of these items, generate a bounded amount of auxiliary information such that range queries asking for the value of the function on sub-lists can be answered within a certain time bound.

For the function "mode" we improve the previously known time bound $O(n^\varepsilon \log n)$ to $O(n^\varepsilon)$ with space $O(n^{2-2\varepsilon})$, where $0 \leq \varepsilon < 1/2$. We improve the space bound $O(n^2 \log\log n/\log n)$ for an $O(1)$ time bounded solution to $O(n^2/\log n)$.

For the function "median" the space bound $O(n^2 \log\log n/\log n)$ is improved to $O(n^2 \log^{(k)} n/\log n)$ for an $O(1)$ time solution, where k is an arbitrary constant and $\log^{(k)}$ is the iterated logarithm.

1 Introduction

In this work we investigate the complexity of the following problem: Let $A = (a_1, \ldots, a_n)$ be a list of elements chosen from some set S and let f be a function defined for lists over S. After possibly computing auxiliary information about A in advance, a sequence of *range queries* asking for $f(a_p, \ldots, a_q)$ with varying p and q has to be answered. We simultaneously bound the size of the additional data stored by the preprocessing (the space) and the time for each query. The computational model is a unit-cost RAM with $\Theta(\log n)$ word length.

The obvious solution of storing answers for all sub-lists uses $\Theta(n^2)$ space and has constant time complexity. Our goal is to improve the space-time product to sub-quadratic bounds.

The problem of range queries is particularly easy if S is a group with a constant time computable operation and $f(a_p, \ldots, a_q) = a_p \cdots a_q$ is the product of all elements in the range. In this case all partial products $m_i = a_1 \cdots a_i$ and their inverses can be precomputed and stored in $O(n)$ space. The computation of $f(a_p, \ldots, a_q) = m_{p-1}^{-1} m_q$ is possible in constant time. Since the list A itself requires space n these bounds are asymptotically optimal.

Among the operators of interest that do not admit the computation of inverses are min and max where S is an ordered set. Nevertheless an optimal $O(n)$ space

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 418–423, 2008.

and $O(1)$ query time solution has been given by Gabow, Bentley, and Tarjan [GBT84], which is based on the solution of the nearest common ancestor problem by Harel and Tarjan [HT84]. A simplified version is due to Bender and Farach-Colton [BFC00, BFCP+05].

A *median* is an element of a sorted multi-set dividing the higher half from the lower half. In comparison to the average (which usually is not a member of the multi-set), the median often more accurately captures the concept of a typical element. As an example of an area where median computations arise we mention the definition of poverty: In the EU a person with an income below 60 % of the median in a country is considered to be at risk of poverty [SCFF05, p. 125]. Another (not necessarily unique) parameter of a multi-set is the *mode*, a value with maximum frequency.

The problem of computing the median and mode for ranges of lists has been investigated by Krizanc, Morin and Smid [KMS05]. For mode we improve their time bound $O(n^\varepsilon \log n)$ with space $O(n^{2-2\varepsilon})$ by a logarithmic factor and for the $O(1)$ time solution we improve the space bound $O(n^2 \log\log n/\log n)$ to $O(n^2/\log n)$.

For the function median we improve the previous space bound $O(n^2 \log\log n/\log n)$ to $O(n^2 \log^{(k)} n/\log n)$ for the $O(1)$ time solution, where $k \geq 1$ is an arbitrary constant and $\log^{(k)}$ is the iterated logarithm.

Algorithms for approximate range mode and median queries are due to Bose, Kranakis, Morin and Tang [BKMT05]. For computing an approximate range mode they also present an $\Omega(n \log n)$ lower bound on the time necessary for preprocessing and answering the query, where the model is that of algebraic decision trees. Notice that the algorithms of [KMS05] and the present work are designed for random access machines.

The new bounds obtained and some previous results are summarized in the following tables (for restrictions on ε see the references):

Range Mode			
space	time	space × time	ref.
$O(n^{2-2\varepsilon})$	$O(n^\varepsilon)$	$O(n^{2-\varepsilon})$	Theorem 1
$O(n^2/\log n)$	$O(1)$	$O(n^2/\log n)$	Theorem 2

Range Median			
space	time	space × time	ref.
$O(n)$	$O(n^\varepsilon)$	$O(n^{1+\varepsilon})$	[KMS05]
$O(n \log^2 n/\log\log n)$	$O(\log n)$	$O(n \log^3 n/\log\log n)$	[KMS05]
$O(n^2 \log^{(k)} n/\log n)$	$O(1)$	$O(n^2 \log^{(k)} n/\log n)$	Theorem 3

2 Results

We need the following observation from [KMS05]:

Lemma 1. *Let A, B, C be multi-sets. If a mode of $A \cup B \cup C$ is not a member of $A \cup C$, then it is a mode of B.*

In the following proofs we often use non-integer values when an integer is required. We assume that these values are appropriately rounded.

Theorem 1. *For every $0 \leq \varepsilon < 1/2$ there is a data structure of size $O(n^{2-2\varepsilon})$ that can answer range mode queries in $O(n^{\varepsilon})$ time.*

Proof. We first notice, that each element can be represented by an integer from $\{1, \ldots, n\}$, since a translation table can be stored within the space bound. We will work only with these numbers and identify them with the original elements.

Let $r = \lceil \varepsilon/(1-2\varepsilon) \rceil$. Notice that r is a constant with $\varepsilon \leq r/(2r+1)$. We divide the list into nested intervals, where each level ℓ interval has length $n^{(r+\ell)\cdot\varepsilon/r}$ for $0 \leq \ell \leq r$. A level ℓ interval thus contains $n^{\varepsilon/r}$ level $\ell - 1$ intervals. For each level r interval i we pre-compute a table f_i of size n that stores the frequency of each element in the prefix of the list up to and including interval i. Notice that the frequency of element e in the range from interval j to k is $f_k[e] - f_{j-1}[e]$ (f_0 is always 0).

For each level ℓ interval i with $\ell > 0$ we choose a constant time computable hash function h_i that is perfect for the elements occurring in the interval and mapping into a set of size $O(n^{(r+\ell)\cdot\varepsilon/r})$ [FKS84]. We store each element e in a table s at position $h_i(e)$. All entries of s that are not mapped to by h_i are filled with an arbitrary element from the interval. In addition we form a table $t(g, j)$ of size $O(n^{(r+\ell+1)\cdot\varepsilon/r})$ that stores for every hash value g and every level $\ell - 1$ interval j the frequency of the hashed element in the prefix of the list up to and including that level $\ell - 1$ interval.

In addition a table of size $O(n^{2-2\varepsilon})$ is set up that stores for each sorted pair of (possibly identical) level 0 intervals the mode of the list between them (including both intervals) and the frequency of each of these modes.

Finally a table c with n entries will be initialized with zeroes. While processing a query this table will record the frequency of certain elements from the selected range. After a query has been processed, the table will be reset to its initial state. An easy way to do this without increasing the time complexity by more than a constant factor is to record all modified entries in a linked list. After processing the query all recorded entries are set to zero.

Suppose a range mode query for the list from position p to position q has to be answered. If p and q are in the same level 0 interval, then in one scan for every element e in the range $c[e]$ is incremented. In a second scan of the interval the element with the maximum count is selected.

In the following we assume that p and q are in different level 0 intervals. Let m be the mode of the pair of (possibly identical) level 0 intervals properly between p and q, or an arbitrary element of the range if p and q are in neighboring level 0 intervals. By Lemma 1 the mode of the sub-list from p to q is m or one of the $O(n^{\varepsilon})$ elements in the prefix resp. suffix of the level 0 intervals containing p resp. q.

By the remarks in the introduction it is sufficient to compute the frequency from the start of the list up to position p (resp. $q - 1$) for each element e which could be the mode in constant time. The frequency can be computed as the difference of the two values. We describe the frequency computation for position p. Let ℓ be the minimal level of intervals containing p and a position at which e

occurs, or $\ell = r+1$ if no such interval exists. This ℓ can be computed in constant time by performing the test $e = s(h_i(e))$ for each of the at most $r+1$ intervals involved. If $\ell = r+1$ the frequency up to the preceding level r interval i (which is the result) can be looked up in f_i. If $r \geq \ell > 0$, then the value $t(h_i(e), j)$ for the level ℓ interval i and the level $\ell - 1$ interval j containing p is the result. If finally $\ell = 0$, then an approximate frequency up to the preceding level 0 interval j can be fetched from $t(h_i(e), j)$, where i is the level 1 interval containing j. All these approximate frequencies are stored in table c. Also the frequency of m is stored in c. Now in a second pass for each occurrence of an element e in the prefix of the level 0 interval containing p entry $c[e]$ is incremented. This can be done in $O(n^{\varepsilon})$ steps, thus in constant time per element. In this way the frequency of each element in a prefix is computed in time $O(1)$. From the non-zero entries of c the maximum frequency is selected and returned as the answer to the query.

Each level r interval is of length $n^{2\varepsilon}$, therefore there are $n^{1-2\varepsilon}$ of them. Hence the tables f_i with n entries require space $O(n^{2-2\varepsilon})$ in total. The space used by each level is dominated by the tables t which are of size $O(n^{1+\varepsilon/r})$. We have $1 + \varepsilon/r \leq 1 + 1/(2r+1) = 2 - (2r/(2r+1)) \leq 2 - 2\varepsilon$ and therefore $O(n^{1+\varepsilon/r}) = O(n^{2-2\varepsilon})$. The table of modes of pairs of level 0 intervals can be stored in space $O((n^{1-\varepsilon})^2) = O(n^{2-2\varepsilon})$. □

Theorem 2. *There is a data structure of size $O(n^2/\log n)$ that can answer range mode queries in $O(1)$ time.*

Proof. In order to simplify computations we assume that the n elements are stored in $A[0, \ldots, n-1]$. For every interval of the list of the form $A[i, j \cdot \log n]$ for $0 \leq i \leq n-1$ and $i \leq j \log n \leq n-1$ its mode is stored in the array m at position $m[i, j]$. For each of the $\log n - 1$ positions $k = j \cdot \log n + 1, j \cdot \log n + 2, \ldots, (j+1) \cdot \log n - 1$ the mode of $A[i, k]$ is the mode of $A[i, k-1]$ or element $A[k]$ by Lemma 1 (notice that one of the sets is empty). This information can be encoded into a single bit and all $\log n - 1$ bits thus determined can be encoded into a number $r[i, j]$ stored in an array r. A systematic way to do this is to store the bit for $k = j \cdot \log n + 1$ as the least significant bit and proceed to more significant bit positions in $r[i, j]$.

Arrays m and r clearly require space $O(n^2/\log n)$. The information stored can be accessed with the help of an auxiliary array b. Entry $b[v, \ell]$ selects for value v the least significant ℓ bits and returns the position of the most significant 1 among those bits, or 0 if all selected bits are 0. Array b is of size $O(n \log n)$. If the mode of $A[i, p]$ has to be determined, then first $j = p \operatorname{div} \log n$ and $\ell = p - j \cdot \log n$ are computed. If $b[r[i, j], \ell] = 0$ then $m[i, j]$ is returned. Otherwise the result is element $A[j \cdot \log n + b[r[i, j], \ell]]$.

Since the number of array accesses is fixed the running time is $O(1)$. □

We define the iterated logarithm function by letting $\log^{(1)} n = \log n$ and $\log^{(k+1)} n = \log \log^{(k)} n$.

Theorem 3. *There is a data structure of size $O(n^2 \log^{(k)} n / \log n)$ that can answer range median queries in $O(1)$ time for every integer $k \geq 1$.*

Proof. For $k = 1$ the statement holds, since the trivial solution of storing answers to all possible queries suffices. In the following we assume that $k \geq 2$.

We construct k levels of data structures. The level 1 structure contains for every pair of blocks of length $b_1 = \log n$ of the list the at most $4b_1$ elements that can possibly be a median for pairs of indices within the blocks (each element in the blocks can be a median, and the $2b_1$ elements in the center of the sorted sequence of blocks properly in between, if any). The level 1 structure requires space $O(n^2/\log n)$.

Suppose level $\ell-1$ has been constructed for $2 \leq \ell \leq k-1$. Then level ℓ contains for every pair of blocks B_i^ℓ and B_j^ℓ of length $b_\ell = \log^{(\ell)} n$ pointers to the at most $4b_\ell$ elements that can be a median for every pair of indices within the blocks. These pointers are relative to the start of the data of the pair of level $\ell-1$ blocks $B_{i'}^{\ell-1}$ and $B_{j'}^{\ell-1}$ containing B_i^ℓ and B_j^ℓ. Therefore $O(\log^{(\ell)} n)$ bits per pointer are sufficient, leading again to a space complexity of $O(n^2/\log n)$.

The construction for level k stores a pointer to the level $k-1$ structure of length $O(\log^{(k)} n)$ for *every* pair of indices, thus the level k structure requires space $O(n^2 \log^{(k)} n/\log n)$.

In order to meet the space bounds, the pointers have to be packed as fields into numbers of $O(\log n)$ bits each. Accessing a field requires shift and bit-mask operations which can be implemented efficiently with the help of tables as explained below (if these operations are assumed not to be included into the instruction set of a RAM).

A query is answered by following pointers starting from level k. If the median of $(a_p, \ldots, a_q)$ has to be computed, the algorithm first fetches $\log^{(k)} n + 2$ bits from a three-dimensional array m_k with $n \times (n \operatorname{div} \log n) \times (\log^{(k)} n + 2)$ entries. It reads $m_k[p, (q \operatorname{div} \log n), ((q \bmod \log n) \cdot (\log^{(k)} n + 2)) \operatorname{div} \log n]$ and extracts $\log^{(k)} n + 2$ bits starting at position $((q \bmod \log n) \cdot (\log^{(k)} n + 2)) \bmod \log n$. Here we start counting at the least significant position and allow for at most $\log n + \log^{(k)} n + 1$ bits per stored word in order to have only one access to m_k. The bits are shifted to the least significant positions, which can be accomplished with the help of a table of size $O(n \log^{(k-1)} n \log n)$, since the number of bits stored is $O(\log n)$. Then the lower $\log^{(k)} n + 2$ bits are selected with the help of a table of size $O(n \log^{(k-1)} n)$. Now the algorithm has computed a pointer i with $0 \leq i < 4 \log^{(k-1)} n$. Suppose a pointer i into the data structure for level $\ell < k$ has been determined. If $\ell \geq 2$ then the algorithm computes $p' = p \operatorname{div} \log^{(\ell)} n$, $q' = q \operatorname{div} \log^{(\ell)} n$, reads $m_\ell[p', (q' \operatorname{div} \log n), ((q' \bmod \log n) \cdot (\log^{(\ell)} n + 2)^2 + i \cdot (\log^{(\ell)} n + 2)) \operatorname{div} \log n]$ and extracts $\log^{(\ell)} n + 2$ bits starting at position $((q' \bmod \log n) \cdot (\log^{(\ell)} n + 2)^2 + i \cdot (\log^{(\ell)} n + 2)) \bmod \log n$. Here the tables have sizes at most $O(n \log^2 n)$ and $O(n \log n)$, which is within the claimed space bound. For level $\ell = 1$ the access is to $m_1[(p \operatorname{div} \log n), (q \operatorname{div} \log n), i]$, where the index of an element is stored directly.

Since the number of levels is fixed and every level requires a constant number of operations or memory accesses, we get time complexity $O(1)$. □

Remark 1. By the method from the proof of Theorem 3 we can also obtain the space bound $O(n^2 \log^* n / \log n)$ with time bound $O(\log^* n)$.

3 Summary

We have improved previous bounds for range mode and range median queries on lists. No non-trivial lower bounds for these problems are known on the RAM. Thus it is not clear how far these solutions are from being optimal and algorithmic improvements as well as lower bounds are problems left open.

Acknowledgments. The author is grateful to Benjamin Hoffmann, Jürn Laun, and the referees for useful comments.

References

[BFC00] Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 88–94. Springer, Heidelberg (2000)

[BFCP+05] Bender, M.A., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P.: Lowest common ancestors in trees and directed acyclic graphs. J. Algorithms 57, 75–94 (2005)

[BKMT05] Bose, P., Kranakis, E., Morin, P., Tang, Y.: Approximate range mode and range median queries. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 377–388. Springer, Heidelberg (2005)

[FKS84] Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with $O(1)$ worst case access time. Journal of the ACM 31, 538–544 (1984)

[GBT84] Gabow, H.N., Bentley, J.L., Tarjan, R.E.: Scaling and related techniques for geometry problems. In: STOC 1984. Proceedings of the sixteenth annual ACM Symposium on Theory of Computing, pp. 135–143. ACM Press, New York (1984)

[HT84] Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM Journal on Computing 13, 338–355 (1984)

[KMS05] Krizanc, D., Morin, P., Smid, M.: Range mode and range median queries on lists and trees. Nordic Journal of Computing 12, 1–17 (2005)

[SCFF05] Schäfer, G., Cervellin, S., Feith, M., Fritz, M. (eds.): Europe in figures — Eurostat yearbook 2005. Office for Official Publications of the European Communities, Luxembourg (2005)

An Automata Theoretic Approach to Rational Tree Relations

Frank G. Radmacher

Lehrstuhl für Informatik 7, RWTH Aachen, Germany
radmacher@automata.rwth-aachen.de

Abstract. We investigate rational relations over trees. Our starting point is the definition of rational tree relations via rational expressions by Raoult (Bull. Belg. Math. Soc. 1997). We develop a new class of automata, called asynchronous tree automata, which recognize exactly these relations. The automata theoretic approach is convenient for the solution of algorithmic problems (like the emptiness problem). The second contribution of this paper is a new subclass of the rational tree relations, called separate-rational tree relations, defined via a natural restriction on asynchronous tree automata. These relations are closed under composition, preserve regular tree languages, and generate precisely the regular sets in the unary case (all these properties fail for the general model), and they are still more powerful than, for instance, the automatic tree relations.

1 Introduction

Automata definable relations over words are widely investigated. Recognizable, automatic, deterministic rational, and (non-deterministic) rational relations result in a well-known hierarchy [3]. Proper generalizations of these theories to trees have been established over the past years in the case of recognizable relations and automatic relations [2,4]. However, it is still debatable how to obtain a reasonable generalization of rational word relations to trees.

Rational relations over words can be introduced in several equivalent ways: First, they are definable via rational expressions (a generalization of regular expressions), which means that rational relations are generated from the finite relations by closure under union, componentwise concatenation, and Kleene star. On the other hand rational relations are recognized by a generalized model of finite automata, so-called *asynchronous automata* (sometimes also called *multi-tape automata*). The theory was developed in [14,7,8,6,1,3].

Generalizing rational relations to trees (resp. terms) is not straightforward. A survey focussing on binary relations (transductions) was given by Raoult in [17]. Attractive results on rational word relations which one would also like for rational tree relations are the following:

- Applied to unary trees, the rational word relations should be generated (also in the case of n-ary relations).
- A characterization via rational expressions should exist (this implies closure under union, some kind of componentwise concatenation, and Kleene star).

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 424–435, 2008.

- A natural automata theoretic characterization should exist.
- Restricted to unary relations the class of regular tree languages should be generated.
- Binary rational tree relations should be closed under composition.
- Binary rational tree relations (transductions) should preserve regular tree languages.

Our automata theoretic approach is a step towards the definition of *deterministic* rational tree relations (cf. [14,13,11] for the word case) and rational relations over *unranked* trees. These theories were started in [16].

Towards a generalization of rational relations to trees, Raoult suggests in [18] defining relations over trees by tree grammars in which non-terminals are represented by tuples of letters (called *multivariables*), so that a synchronization between the productions is possible. Raoult calls these relations *rational tree relations* and gives also a characterization in terms of rational expressions.

Complementary Raoult's grammars, the first contribution of this paper are so-called *asynchronous tree automata* which recognize exactly the rational tree relations. With our automata theoretic approach it is possible to address certain properties and (un-) decidability results of rational tree relations.

Rational tree relations in the mentioned format have a few drawbacks. They do not coincide with regular tree languages in the unary case, they are not closed under composition, and if considered as transductions they do not preserve regular tree languages. In [18] Raoult proposes a restriction of his tree grammars to so-called *transduction grammars* which resolve these problems. But these have the disadvantage that, when applied to unary trees, they can only be considered as a generalization of binary rational word relations, but not of the n-ary case. Furthermore, Raoult's restriction is difficult to adapt to tree automata, i. e. it misses a natural automata theoretic characterization. To take account of these problems the second contribution of this paper is such a natural restriction of rational tree relations (which semantically differs from Raoult's one). These so-called *separate-rational tree relations* meet all the properties demanded above and are still more powerful than automatic tree relations [2].

The remainder of this paper is structured as follows. First we fix a few notations in Sect. 2. In Sect. 3 we define rational tree relation introduced by Raoult, develop asynchronous tree automata, and show the equivalence. In Sect. 4 we introduce separate-rational relations and corresponding separate-asynchronous automata. Section 5 contains a conclusion and an outlook on further research.

2 Preliminaries

We assume the reader is familiar with the basics of tree automata [9,4] and with rational relations over words [1,6]. Here, we fix just a few notations and conventions used throughout this paper.

We consider trees and tuple of trees over *ranked alphabets* $\Sigma = \Sigma_0 \uplus \ldots \uplus \Sigma_m$ (where Σ_i contains exactly the symbols of rank i). Often we will state the rank of a symbol in parentheses as superscript. So, $f^{(2)}$ means that the symbol f has

rank 2. A tree t is represented as a pair $(\mathrm{dom}_t, \mathrm{val})$ where dom_t is the set of tree nodes and $\mathrm{val} : \mathrm{dom}_t \to \Sigma$ maps each node of rank k to a symbol in Σ_k. Similarly, a tuple $\bar{t} = (t_1, \ldots, t_n)$ of trees is represented as $(\mathrm{dom}_{\bar{t}}, \mathrm{val})$ where $\mathrm{dom}_{\bar{t}}$ is the disjoint union of the dom_{t_i}. We write trees as terms in the standard way. The *height* of a tree resp. a tuple of trees is defined as the number of nodes of a longest path from a root to a leaf. For example a tree which only consists of the root has a height of 1. With T_Σ we denote the set of all trees over Σ. A *tree language* resp. *tree relation* is a subset of T_Σ resp. $(T_\Sigma)^n$. In Sect. 4 we will also distinguish alphabets for each (projection to one) component of a relation.

3 Rational Tree Relations

In this section we present the theory of rational tree relations starting from Raoult's definition via rational expressions [18]. Then we define asynchronous tree automata, show the equivalence to Raoult's definition, and deal with some closure properties and (un-) decidability results of rational tree relations.

3.1 Definition of Rational Tree Relations Via Rational Expressions

Example 1. Consider the rational expression

$$(cx_1y_1, cbx_2y_2)^{*_{y_1y_2}} \cdot_{y_1y_2} (a, a) \cdot_{x_1x_2} (bz_1, bz_2)^{*_{z_1z_2}} \cdot_{z_1z_2} (a, a)$$

over the ranked alphabet $\Sigma = \{a^{(0)}, b^{(1)}, c^{(2)}\}$. In this example we use "multi-variables" x_1x_2, y_1y_2, z_1z_2 (written also as X, Y, Z) which are subject to simultaneous substitution. The form of tuples of the rational tree relation defined by above expression is depicted in Fig. 1. We see that the multivariable $X = x_1x_2$ occurs in distinct instances $x_1x_2, x_1'x_2', \ldots$ which have to be distinguished. Here, the number of possible instances of X cannot be bounded by a natural number. Each instance of the multivariable X becomes substituted with two unary trees of same height (see Fig. 3(a) on page 429 for a full example pair of trees).

Towards the formal definition, let $\mathcal{V}$ be a set of variables. A *multivariable* is a sequence in $\mathcal{V}^+$ containing at most one occurrence of any variable. For

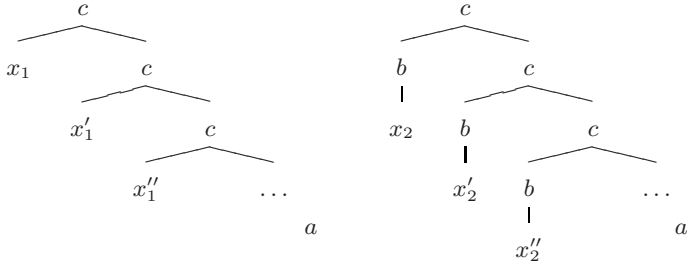


Fig. 1. Generating an unbounded number of instances of a multivariable

$X = x_1 \cdots x_n$ $(n > 0)$ we say that the multivariable X has length $|X| := n$. The set of all *instances* of variables resp. multivariables is the cartesian product $\mathcal{V} \times \mathbb{N}$ resp. $\mathcal{V}^+ \times \mathbb{N}$. We say (x, j) is the j-th instance of variable x, written x^j, and (X, j) is the j-th instance of multivariable X, written X^j. In order to avoid too many indices in the notation, we write instances $x_i^0, x_i^1, x_i^2, \ldots$ of a variable x_i also in the form $x_i, x_i', x_i'', \ldots$.

Instances of variables are nullary symbols which can only occur as leaves. Furthermore, each instance of a multivariable can occur in a tuple of trees at most once, and if an instance of a variable occurs, so all other variables of the same multivariable and same instance: Formally, let $\bar{t} \in T_\Sigma^m$, let $X = x_1 \cdots x_n$ be a multivariable where x_i^j occurs in $\bar{t}$; then each $x_{i'}^j$ occurs in $\bar{t}$ exactly once (and as leaf) for $1 \leq i, i' \leq n$.

Let $X = x_1 \cdots x_n$ be a multivariable of length n, R a relation over n-tuples of trees, S a relation over m-tuples of trees, and $\bar{t}$ a m-tuple of trees containing k instances of X. Then the concatenation of a tuple with a tree relation is defined as $\bar{t} \cdot_X R := \{\bar{t}' \mid \bar{t}' \text{ results from } \bar{t} \text{ by substituting each of the } k \text{ instances of } X \text{ with a tuple from } R\}$. The concatenation of two tree relations is defined as $S \cdot_X R := \{\bar{t} \cdot_X R \mid \bar{t} \in S\}$, and the iterated concatenation and the Kleene star for tree relations are defined as $R^{0X} := \{(x_1^j, \ldots, x_n^j)\}$, $R^{nX} := \{(x_1^j, \ldots, x_n^j)\} \cup R \cdot_X R^{(n-1)X}$, and $R^{*X} := \bigcup_{n \geq 0} R^{nX}$. In the case of the iterated concatenation the instance $j \in \mathbb{N}$ is chosen as a new instance, so that it occurs in the resulting relation only once.

Definition 1 ([18]). *The classes Rat_n of* rational tree relations *are defined inductively as follows:*

- *Each finite n-ary tree relation is in Rat_n.*
- $R \in Rat_n \wedge S \in Rat_n \Rightarrow R \cup S \in Rat_n$.
- $R \in Rat_n \wedge |X| = m \wedge S \in Rat_m \Rightarrow R \cdot_X S \in Rat_n$.
- $R \in Rat_n \wedge |X| = n \Rightarrow R^{*X} \in Rat_n$.

We denote the unary relations in the class Rat_1 as *rational tree languages*. Note that the class Rat_1 does not coincide with the class of regular tree languages:

Example 2. The rational expression $(fx_1x_2) \cdot_{x_1x_2} (gy_1, gy_2)^{*_{y_1y_2}} \cdot_{y_1y_2} (aa)$ over the ranked alphabet $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$ describes the tree language $T_{\mathrm{sim}} = \{f(g^n a, g^n a) \mid n \in \mathbb{N}\} \in \mathrm{Rat}_1$, but T_{sim} is not regular.

3.2 Asynchronous Tree Automata

Now we introduce a class of automata recognizing exactly the class Rat_n of rational tree relations. The above considered Examples 1 and 2 show that these automata basically have to provide the following three mechanisms:

- Certain transitions are supposed to be used simultaneously. We will achieve this by combining states to tuples of states which we will call *macro states*. In the runs of our automata all states of a macro state have to be reached and left simultaneously.

We write a finite set of *macro states* as $\mathfrak{Q} = \{\mathfrak{q}_1, \ldots, \mathfrak{q}_k\}$ where $\mathfrak{q}_1, \ldots, \mathfrak{q}_k$ are tuples of states taken from a finite set Q of states (Q contains all states that occur in some $\mathfrak{q} \in \mathfrak{Q}$). A macro state has the form $\mathfrak{q} = (q_1, \ldots, q_l)$ with $l \geq 1$. All macro states in $\mathfrak{Q}$ are "pairwise disjoint", i. e. $\{q_1, \ldots, q_l\} \cap \{p_1, \ldots, p_m\} = \emptyset$ for all macro states $\mathfrak{q} = (q_1, \ldots, q_l)$ and $\mathfrak{p} = (p_1, \ldots, p_m)$ in $\mathfrak{Q}$.

- In addition we require some mechanism to allow asynchronous moves. We will achieve this by the addition of ε-transitions. This enables the automaton to do a bottom-up step in one component and to stay in place in another component (possibly just changing the state).
- An unbounded number of instances of macro states has to be distinguished. In a run we have to distinguish whether states belong to the same or to different instances. We will achieve this by combining each state in a transition with a variable. States with same variables must belong to the same instance when these transitions are used. In a run of our automaton, variables will be instantiated with natural numbers to denote the different instances.

Example 3. Consider macro states $\mathfrak{p} = (p_1, p_2)$ and $\mathfrak{q} = (q_1, q_2)$. Then two transitions $((p_1, x), (p_1, y), (p_2, y), f, (q_1, z))$, $((p_2, x), \varepsilon, (q_2, z))$ enable a bottom-up computation step as depicted in Fig. 2.

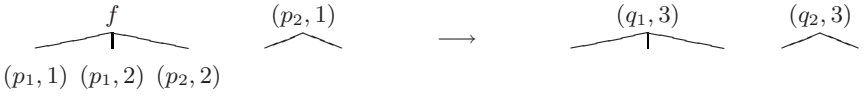


Fig. 2. A computation step of an asynchronous tree automaton

Before we give the formal definition of asynchronous tree automata, we start with a comprehensive example.

Example 4. Consider the rational relation from Example 1. We define an asynchronous tree automaton recognizing this relation. Formally, we will denote our automaton with $\mathcal{A}^{(2)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma, \Delta, \mathfrak{F} \rangle$ (the superscript indicates that the automaton runs on pairs of trees). $\Sigma = \{a^{(0)}, b^{(1)}, c^{(2)}\}$ is a ranked alphabet. The used macro state set $\mathfrak{Q} = \{(q_{a_1}, q_{a_2}), (q_{b_1}, q_{b_2}), (q_{c_1}, q_{c_2})\}$ consists of pairwise disjoint tuples of states in Q. We declare the macro states of the set $\mathfrak{F} \subseteq \mathfrak{Q}$ as final. In this example we declare only the macro state (q_{c_1}, q_{c_2}) as final. $\mathcal{A}^{(2)}$ has the following transitions in its transition relation Δ which employ variables of the set $\mathrm{Var} = \{x, y, z\}$:

$$
\begin{aligned}
&(a, (q_{a_1}, x)), && (a, (q_{c_1}, x)),\\
&(a, (q_{a_2}, x)), && (a, (q_{c_2}, x)),\\
&((q_{a_1}, x), b, (q_{a_1}, x)), && ((q_{a_1}, x), \varepsilon, (q_{b_1}, x)),\\
&((q_{a_2}, x), b, (q_{a_2}, x)), && ((q_{a_2}, x), b, (q_{b_2}, x)),\\
&((q_{b_1}, x), (q_{c_1}, y), c, (q_{c_1}, z)), && ((q_{b_2}, x), (q_{c_2}, y), c, (q_{c_2}, z)) \ .
\end{aligned}
$$

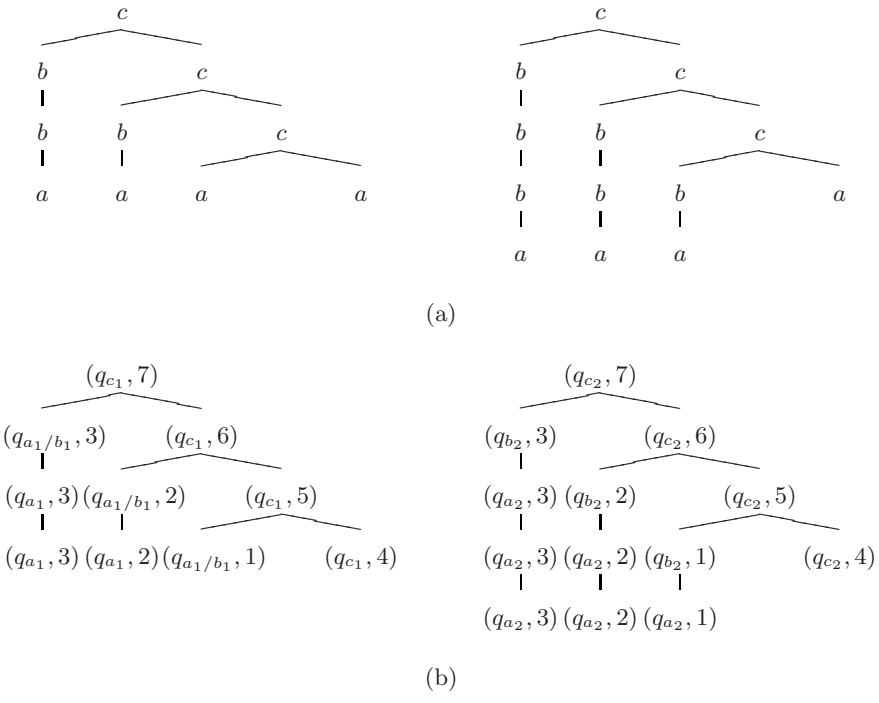


Fig. 3. (a) A pair of trees; (b) an accepting run of $\mathcal{A}^{(2)}$ on this pair of trees

Figure 3 shows a pair of trees and an accepting bottom-up run of $\mathcal{A}^{(2)}$ on this pair. For instance, in a first step of this accepting run the first instantiation of the macro state (q_{a_1}, q_{a_2}) is assigned to a pair of leaves resulting in the labelings $(q_{a_1}, 1)$ and $(q_{a_1}, 2)$. In a second step this macro state changes to (q_{b_1}, q_{b_2}) by application of the transitions $((q_{a_1}, x), \varepsilon, (q_{b_1}, x))$ and $((q_{a_2}, x), b, (q_{b_2}, x))$. Note that the numbering of instances is rather arbitrary as long as different instances of variables can be distinguished. Due to lack of space we illustrate all intermediate configurations of the run in one tree. If a node is part of two different cuts in the run (due to the use of ε-transitions), we label this node with both configurations in an abbreviated form, e. g. for a node v and two configurations $c_1(v) = (q_{a_1}, 3)$ and $c_2(v) = (q_{b_1}, 3)$ we label v with $(q_{a_1/b_1}, 3)$.

Now we give a formal definition of asynchronous tree automata.

Definition 2. *An asynchronous tree automaton over a ranked alphabet* $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_m$ *is a tuple* $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, Var, \Sigma, \Delta, \mathfrak{F} \rangle$ *with*

- *a finite set* Q *of states,*
- *a set* $\mathfrak{Q}$ *of macro states over* Q *(i. e. pairwise disjoint tuples of states in* Q*),*
- *a finite set* Var *of variables,*

– *a transition relation*

$$\Delta \subseteq \bigcup_{i=0}^{m} \big((Q \times Var)^i \times \Sigma_i \times Q \times Var\big) \ \cup \ (Q \times Var \times \{\varepsilon\} \times Q \times Var) \ ,$$

– *and a set* $\mathfrak{F} \subseteq \mathfrak{Q}$ *of final macro states.*

An instantiation *of a set* $\mathcal{V} \subseteq Var$ *of variables is an injective function* $I_{\mathcal{V}} : \mathcal{V} \to \mathbb{N}, x \mapsto \alpha$. *We also refer to* $\alpha \in \mathbb{N}$ *as the* instance α. *A* cut C *of an* n*-tuple* $(t_1, \ldots, t_n)$ *of trees is an antichain in* $dom_{(t_1,\ldots,t_n)}$ *(consisting of pairwise incomparable nodes w. r. t. the prefix ordering). The computation shifts the cut stepwise upwards until it reaches the antichain of the root nodes of* $t_1, \ldots, t_n$ *(if possible). A* configuration *is a mapping* $c : C \to Q \times \mathbb{N}$ *which associates an instantiated state to each node of* C *through the tuple* $(t_1, \ldots, t_n)$. *We require that the instances of states are the same within each macro state of a configuration; also different occurrences of a state in a configuration appear with different instances (formally* $c(v_1) \neq c(v_2)$ *for all* $v_1 \neq v_2$*).*

A makes a computation step $c_1 \to c_2$ *between two configurations* $c_1 : C_1 \to Q \times \mathbb{N}$ *and* $c_2 : C_2 \to Q \times \mathbb{N}$ *where* C_2 *contains the parents of* C_1*-nodes reached via a proper transition, those* C_1*-nodes which are only subject to state changes by* ε*-transitions, and those* C_1*-nodes which are not affected by any transitions in this step and hence stay unchanged. More precisely, we require that there exist nodes* $v_1, \ldots v_k$ *with children* $v_{1,1}, \ldots, v_{1,l}$ *of* v_1*, children* $v_{2,1}, \ldots, v_{2,l}$ *of* v_2*, ... and children* $v_{k,1}, \ldots, v_{k,l}$ *of* v_k *as well as nodes* $v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}$*, so that the following conditions are fulfilled:*

1. $\{v_{1,1}, \ldots, v_{k,l}, v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}\} \subseteq C_1$.
2. $C_2 = (C_1 \setminus \{v_{1,1}, \ldots, v_{k,l}\}) \cup \{v_1, \ldots, v_k\}$.
3. *There exist proper transitions*

   $$((q_{1,1}, x_{1,1}), \ldots, (q_{1,l}, x_{1,l}), val(v_1), (q_1, x)), \ \ldots \ , ((q_{k,1}, x_{k,1}), \ldots, (q_{k,l}, x_{k,l}), val(v_k), (q_k, x))$$

   and ε*-transitions*

   $$((q_{\varepsilon_1}, x_{\varepsilon_1}), \varepsilon, (q_{\varepsilon'_1}, x)), \ \ldots \ , ((q_{\varepsilon_j}, x_{\varepsilon_j}), \varepsilon, (q_{\varepsilon'_j}, x))$$

   in Δ*, so that*
   - $q_1, \ldots, q_l, q_{\varepsilon'_1}, \ldots, q_{\varepsilon'_j}$ *form exactly one macro state,*
   - $q_{1,1}, \ldots, q_{k,l}, q_{\varepsilon_1}, \ldots, q_{\varepsilon_j}$ *form a union of certain macro states, and all states belonging to the same macro state occur with the same variable,*
   - *there exist an instantiation* $I_{\mathcal{V}}$ *of a variable set* $\mathcal{V} \subseteq Var$*, so that these transitions with each variable* $x \in \mathcal{V}$ *replaced by* $I_{\mathcal{V}}(x)$ *match exactly the computation step* $c_1 \to c_2$.
4. c_2 *is identical to* c_1 *on* $(C_1 \cap C_2) \setminus \{v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}\}$.

The configuration $c : C \to Q \times \mathbb{N}$ *with* $C = \emptyset$ *is called* start configuration. *A configuration* $c : C \to Q \times \mathbb{N}$ *is* accepting *iff* $C = \{root_1, \ldots, root_n\}$ *with roots* $root_i$ *of* t_i *(*$1 \leq i \leq n$*), and there exist a final macro state* $(q_1 \ldots q_n) \in \mathfrak{F}$ *and an* $\alpha \in \mathbb{N}$*, so that* $c(root_1) = (q_1, \alpha), \ldots, c(root_n) = (q_n, \alpha)$. *A sequence of*

configurations is a run *iff* $c_1 \to \ldots \to c_m$ *and* c_1 *is the start configuration. Such a run is called* accepting *iff* c_m *is accepting.* $\mathcal{A}^{(n)}$ *recognizes the* n*-ary relation*

$$R(\mathcal{A}^{(n)}) = \{(t_1, \ldots, t_n) \mid \textit{there exists an accepting run of } \mathcal{A}^{(n)} \textit{ on } (t_1, \ldots, t_n)\} \ .$$

3.3 The Equivalence Theorem

The equivalence theorem is an adaption of the Kleene-Theorem for tree languages (see [9,4]). (For the detailed proof we refer to [15].)

Theorem 1. *A relation* R *of* n*-tuples of trees is rational if and only if there exists an asynchronous tree automaton* $\mathcal{A}^{(n)}$ *with* $R(\mathcal{A}^{(n)}) = R$.

Proof (Sketch). The $\Rightarrow$-direction of the proof goes by induction over rational expressions. For the induction start the construction of an asynchronous tree automaton for a singleton of a tuple of trees suffices. Here it is important to prepare the induction step by reading each instance of a multivariables at the leaves simultaneously. For the induction step asynchronous tree automata for the operations $\cup$, $\cdot_X$ and $*_X$ according to Definition 1 are easy to construct.

For the $\Leftarrow$-direction it can be shown for each asynchronous tree automaton $\mathcal{A}^{(n)}$ that its recognized relation is rational. The result can be shown by an induction over the set of "intermediate macro states" $\mathfrak{S}$ of the runs of $\mathcal{A}^{(n)}$. As intermediate macro states we count macro states which occur in other configurations than start configurations at the leaves or an end configuration at the root. For the induction start ($\mathfrak{S} = \emptyset$) we have to consider trees accepted by $\mathcal{A}^{(n)}$ without intermediate macro states. These are n-tuples of trees of height 1 or 2 only. Since these are only finitely many, they form a rational relation. For the induction step ($|\mathfrak{S}| > 0$) it suffices to give a rational expression which composes relations with $|\mathfrak{S}| - 1$ intermediate macro states to a relations with $|\mathfrak{S}|$ intermediate macro states and which is accepted by $\mathcal{A}^{(n)}$. □

3.4 Properties of Rational Tree Relations

Now we present some closure properties and (un-) decidability results, also recalling some "defects" of the rational tree relations which were noted in [18].

For a word relation R we define a tree relation $\mathrm{TRel}(R)$ by interpreting each word $u = a_1 a_2 \cdots a_n$ of a tuple of R as an unary tree $u\$ = a_1(a_2(\ldots(a_n(\$))\ldots))$. For an n-ary word relation $R \subseteq \Sigma_1^* \times \ldots \times \Sigma_n^*$ the tree relation $\mathrm{TRel}(R)$ over $\Sigma_1 \cup \{\$^{(0)}\}, \ldots, \Sigma_n \cup \{\$^{(0)}\}$ is defined as $\mathrm{TRel}(R) := \{(u_1\$, \ldots, u_n\$) \mid (u_1, \ldots, u_n) \in R\}$. The following results are easy to prove by construction of corresponding automata for each direction:

Lemma 1. *Let* R *be a word relation. Then* R *is rational iff* $TRel(R)$ *is rational.*

Due to Lemma 1 some elementary closure properties and all undecidability results of rational word relations can be extended to trees easily:

Proposition 1. *(a) The class* Rat_n *of* n*-ary rational tree relations is closed under union, not closed under intersection, and not closed under complementation.*

(b) For rational tree relations $R_1, R_2 \in Rat_n$ it is undecidable to determine whether $R_1 \cap R_2 = \emptyset$, $R_1 \subseteq R_2$, and $R_1 = R_2$.

The *membership problem* for asynchronous tree automata is decidable, i. e. it is decidable whether $(t_1, \ldots, t_n) \in R(\mathcal{A})$. Also the *emptiness problem*, i. e. the question whether $R(\mathcal{A}) = \emptyset$, and the *infinity problem*, i. e. the question whether $|R(\mathcal{A})|$ is infinite, are decidable. (The proofs can be found in [15].)

Theorem 2. *Given an asynchronous tree automata with macro state set $\mathfrak{Q}$ and transition relation Δ, and a tuple of trees with m nodes. The membership problem is decidable in $O(|\Delta|^m)$ time, and the emptiness and the infinity problem are decidable in $O(|\mathfrak{Q}|^2 \cdot |\Delta|)$ time.*

Unlike binary rational relations over words, the class Rat_2 of binary rational tree relations is not closed under composition:

Example 5. The binary tree relations $R_1 = \{(b^m a^n \$, f(a^n \$, b^m \$)) \mid m, n \in \mathbb{N}\}$ and $R_2 = \{(f(a^n \$, b^m \$), a^n b^m \$) \mid m, n \in \mathbb{N}\}$ are rational, but the composition $\{(b^m a^n \$, a^n b^m \$) \mid m, n \in \mathbb{N}\}$ is *not* rational.

Binary rational relations over words are also called *(rational) transductions*. They preserve regular and context-free languages, i. e. the image and the inverse image of a regular (resp. a context-free) language under a transduction is again a regular (resp. context-free) language [1]. Here we note that binary rational tree relations do not even preserve regularity:

Example 6 ([18]). Consider the rational tree relation $T_{\mathrm{sim}} = \{f(g^n a, g^n a) \mid n \in \mathbb{N}\}$ from Example 2. Clearly, $R := \Sigma^* \times T_{\mathrm{sim}}$ is rational. The image of a regular language under R is T_{sim} which is not regular. An analogous result for the inverse image can be proved with a relation $R' := T_{\mathrm{sim}} \times \Sigma^*$.

4 Separate-Rational Tree Relations

We have seen a few drawbacks of rational tree relations. They do not coincide with regular tree languages in the unary case, are not closed under composition, and do not preserve regular tree languages. In [18] Raoult proposes a restriction of rational tree relations, generated by so-called *transduction grammars*. These reestablish the demanded properties, but as mentioned in the introduction they have other drawbacks: They are not a proper generalization of rational word relations in the n-ary case, and the restriction is difficult to adapt to asynchronous tree automata. So, we define yet another restriction, both for rational expressions and asynchronous tree automata, resolving these issues.

The idea is to define a class of relations which can be computed by asynchronous tree automata which have all their macro states separated between the components, i. e. each state of a macro state can only occur in one component.

Definition 3. *The classes $SepRat_n$ of* separate-rational tree relations *are defined inductively as follows:*

- $\emptyset \in SepRat_n$.
- $\{(t_1, \ldots, t_n)\} \in SepRat_n$, *where* $t_1, \ldots, t_n$ *are only trees of height* 1 *or* 2 *and each component contains at most one variable of each multivariable.*
- $R \in SepRat_n \wedge S \in SepRat_n \Rightarrow R \cup S \in SepRat_n$.
- $R \in SepRat_n \wedge |X| = m \wedge S \in SepRat_m \Rightarrow R \cdot_X S \in SepRat_n$, $m \leq n$, *where each component of a tuple in* R *contains at most one variable of* X.
- $R \in SepRat_n \wedge |X| = n \Rightarrow R^{*_X} \in SepRat_n$, *where each component of a tuple in* R *contains exactly one variable of* X.

Example 7. (a) The relation from Example 1 is separate-rational. The rational expression can be rewritten as $(cx_1y_1, cx_2y_2)^{*_{y_1y_2}} \cdot_{y_1y_2} (a, a) \cdot_{x_1x_2} (x_1, bx_2) \cdot_{x_1x_2} (bz_1, bz_2)^{*_{z_1z_2}} \cdot_{z_1z_2} (a, a)$. It is generated by trees of height 2 at most, and all multivariables are separated between the components of the tuples.

(b) The rational relations R_1 and R_2 from Example 5 are *not* separate-rational, because multivariables of length 3 are are easily seen to be necessary in order to define these relations. So, at least two variables of one multivariable have to occur in the same component of a tuple.

We will restrict asynchronous tree automata, so that these recognize exactly the class of separate-rational relations. For the separate-asynchronous case we allow the automata to utilize a specific ranked alphabet for each component.

Definition 4. *A separate-asynchronous tree automaton* $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, Var, \Sigma_1, \ldots, \Sigma_n, \Delta, \mathfrak{F} \rangle$ *is an asynchronous tree automaton over* $\Sigma_1 \cup \ldots \cup \Sigma_n$ *(each* $\Sigma_j = \Sigma_{0_j} \cup \ldots \cup \Sigma_{m_j}$ *is a ranked alphabets) with the following restrictions:*

- *the set* Q *of states is partitioned in* $Q = Q_1 \uplus \ldots \uplus Q_n$,
- *for each macro state* $(q_1, \ldots, q_m) \in \mathfrak{Q}$ *and all* $q_k \neq q_l$, $1 \leq k, l \leq m$, $1 \leq j \leq n$ *holds:* $q_k \in Q_j \Rightarrow q_l \notin Q_j$,
- *the transition relation is partitioned in* $\Delta = \Delta_1 \uplus \ldots \uplus \Delta_n$ *with*

$$\Delta_j \subseteq \cup_{i=0}^{m} \big((Q_j \times Var)^i \times \Sigma_{i_j} \times Q_j \times Var \big) \cup (Q_j \times Var \times \{\varepsilon\} \times Q_j \times Var) \ ,$$

- *each final macro state* $\mathfrak{q} \in \mathfrak{F}$ *has the form* $\mathfrak{q} = (q_1, \ldots, q_n)$ *with* $q_i \in Q_i$ *for all* $1 \leq i \leq n$.

The Equivalence Theorem (Theorem 1) can be reformulated for separate-rational relations. Only slight modifications are necessary. It should be mentioned that the restriction to elementary trees of height 1 or 2 in Definition 4 is important for the "$\Rightarrow$"-direction of the proof in order to handle the induction start. Also, this condition is not a restriction for the "$\Leftarrow$"-direction, because in the original proof the induction start only results in trees of height 1 or 2.

Theorem 3. *A relation* R *of* n*-tuples of trees is separate-rational if and only if there exists a separate-asynchronous tree automaton* $\mathcal{A}^{(n)}$ *with* $R(\mathcal{A}^{(n)}) = R$.

Lemma 1 can be reformulated for separate-rational relations. So, we obtain the same undecidability results and closure properties which we derived for rational tree relations from Lemma 1. Beyond this, separate-rational relations resolve the issues raised in Sect. 3.

Theorem 4. *(a) The class $SepRat_1$ of separate-rational tree languages is the class of regular tree languages.*
(b) The class $SepRat_2$ of binary separate-rational tree relations is closed under composition.
(c) The image and the inverse image of a regular tree language under a binary separate-rational tree relation R are again regular tree languages.

Proof. (a) For $n = 1$ all multivariables have length 1 resp. all macro states have size 1, yielding regular tree languages.

(b) Construct a separate-asynchronous automaton recognizing $R \circledcirc S := \{(t, t', t'') \mid (t, t') \in R, (t', t'') \in S\}$ for separate-rational tree relations R and S by synchronization of the common component. The projection on the first and third component yields a separate-asynchronous automaton for $R \circ S$. (We refer to [15] for the detailed proof.)

(c) Due to symmetry of Definition 4, it suffices to show that the image of a regular tree language under a binary separate-rational relation is regular. Clearly, the identity $\mathrm{id}_T = \{(t, t) \mid t \in T\}$ of a regular tree language T is separate-rational. Thus, the image of T under a separate-rational relation R is the projection on the second component of $\mathrm{id}_T \circ R$. Due to Theorem 4(b) $\mathrm{id}_T \circ R$ is also a separate-rational. The projection on the second component yields a regular tree language (due to the closure of SepRat under projections [15] and Theorem 4(a)). □

If we consider rational relations over words, they also preserve context-free languages [1]. It is an open question whether separate-rational tree relations also preserve context-free tree languages as defined in [10].

5 Conclusion

We presented an automata theoretic approach to rational tree relations which now can be described by three equivalent formalisms: Rational expressions, tree grammars [18], and asynchronous tree automata. Separate-rational tree relations overcome some drawbacks of the rational tree relations. This restriction is natural, since it is easy to apply to all three formalisms (tree grammars were not discussed here, but can be restricted like rational expressions). Separate-rational tree relations are a proper generalization of rational word relations and are still more powerful than, for instance, automatic tree relations.

Outlook: Rational tree relations are more powerful than *linear tree transducers* (as defined in [4]) and some cases of *term rewriting systems* [12]. These results do not hold for the separate-rational restriction. More expressive extensions of separate-rational relations with such features need to be investigated.

Asynchronous tree automata allow the definition of rational relations over unranked trees and the definition of deterministic rational tree relations (both over ranked and unranked trees). For the deterministic top-down model see [5,16]. A deterministic bottom-up model seems to be more challenging (due to the non-deterministic grouping of nodes in a run for the instantiation with macro

states). A further restriction of separate-rational automata may yield a model which generalizes deterministic rational word relations on the one hand and includes recognizable and automatic tree relations on the other hand.

Acknowledgements. This work contains some results of my diploma thesis [16]. Special thanks go to Wolfgang Thomas for supervising this work and for his numerous helpful suggestions.

References

1. Berstel, J.: Transductions and Context-Free Languages. Leitfäden der angewandten Mathematik und Mechanik 38. Teubner, Stuttgart (1979)
2. Blumensath, A., Grädel, E.: Finite presentations of infinite structures: Automata and interpretations. Theory of Computing Systems 37, 641–674 (2004)
3. Carton, O., Choffrut, C., Grigorieff, S.: Decision problems among the main subfamilies of rational relations. Theor. Informat. Appl. 40(2), 255–275 (2006)
4. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications. Unpublished electronic book (1997), http://www.grappa.univ-lille3.fr/tata
5. Cristau, J., Löding, C., Thomas, W.: Deterministic automata on unranked trees. In: Liśkiewicz, M., Reischuk, R. (eds.) FCT 2005. LNCS, vol. 3623, pp. 68–79. Springer, Heidelberg (2005)
6. Eilenberg, S.: Automata, Languages and Machines, vol. A. Academic Press, New York (1974)
7. Elgot, C.C., Mezei, J.E.: On relations defined by generalized finite automata. IBM Journal of Research and Development 9(1), 47–68 (1965)
8. Fischer, P.C., Rosenberg, A.L.: Multitape one-way nonwriting automata. Journal of Computer and System Sciences 2(1), 88–101 (1968)
9. Gécseg, F., Steinby, M.: Tree Automata, Akadémiai Kiadó, Budapest (1984)
10. Gécseg, F., Steinby, M.: Tree Languages. In: Handbook of Formal Languages, Beyond Words, vol. 3, pp. 1–68. Springer, Heidelberg (1997)
11. Grigorieff, S.: Modelization of deterministic rational relations. Theoretical Computer Science 281(1-2), 423–453 (2002)
12. Meyer, A.: On term rewriting systems having a rational derivation. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 378–392. Springer, Heidelberg (2004)
13. Pelletier, M., Sakarovitch, J.: On the representation of finite deterministic 2-tape automata. Theoretical Computer Science 225(1-2), 1–63 (1999)
14. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM Journal of Research and Development 3(2), 115–125 (1959)
15. Radmacher, F.G.: An automata theoretic approach to the theory of rational tree relations. Tech. Rep. (2007), http://www.automata.rwth-aachen.de/~radmacher/
16. Radmacher, F.G.: Automatendefinierbare Relationen über Bäumen (Automata Definable Relations over Trees). Diploma thesis (revised version), RWTH Aachen (2007), http://www.automata.rwth-aachen.de/~radmacher/
17. Raoult, J.-C.: A survey of tree transductions. In: Nivat, M., Podelski, A. (eds.) Tree Automata and Languages, pp. 311–326. Elsevier, Amsterdam (1992) (also published as report 1410 INRIA-Rennes, 1991)
18. Raoult, J.-C.: Rational tree relations. Bulletin of the Belgian Mathematical Society 4(1), 149–176 (1997)

Slicing Petri Nets with an Application to Workflow Verification*

Astrid Rakow

Department für Informatik, Universität Oldenburg
astrid.rakow@informatik.uni-oldenburg.de

Abstract. We introduce the notion of net-slice to describe a subnet of a marked Petri net Σ that approximates Σ's temporal behaviour with respect to a set of places P. We consider slices Σ' whose set of places comprises the places referred to by a CTL* formula ϕ. If Σ is fair w.r.t. the transitions of a slice, $\Sigma \models \phi$ can be verified and falsified by examining the slice, given ϕ is a CTL* formula built without using the next-time operator. Verification of LTL$_{\text{-X}}$ formulas is thus possible under these very weak fairness assumptions, though LTL formulas using next-time can be falsified.

Keywords: Verification, Net Reduction, Slicing, CTL$^*_{\text{-X}}$, Workflow nets.

1 Introduction

Slicing is a technique to syntactically reduce a model in such a way that at best the reduced model contains only those parts that may influence the property the model is analysed for. Slicing has originally been developed for software analysis [1] to minimise the program size by "slicing away" bits of the program that are not relevant for the current analysis. It has successfully been applied to support software developers in tasks like program understanding, integration, maintenance and testing [2]. Since Mark Weiser in his original publication [1] introduced the first program slicing algorithm, the concept of slicing has been applied to formalisms other than programming languages such as attribute grammars [3], hierarchical state machines [4] and Z- and CSP-OZ-Specifications [5,6,7]. In [8] Chang and Wang present an algorithm on Petri nets, that slices out all sets of paths, called concurrency sets, such that all paths within the same set should be executed concurrently.

In this paper we introduce slicing as a method to alleviate the state explosion problem for model checking on Petri nets. Our slicing algorithm derives a slice Σ' from a net Σ for a set of places, P. Given P is the set of atomic propositions of a CTL* formula φ we examine under which conditions $\Sigma \models \varphi$ can be verified or falsified.

In the next section we give definitions of the basic terms used in this paper. We introduce our slicing algorithm and present the results concerning verification and falsification of a CTL* formula by means of a slice in Sect. 3. In Sect. 4

* This work is supported by the German Research Foundation (DFG), grant GRK 1076/1.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 436–447, 2008.

we apply our slicing method to a small example which shows that due to their normally non-strongly-connected nature, workflow systems are particularly apt to be reduced in this way. We give a short overview of related work before drawing the conclusions in Sect. 5.

2 Basic Definitions

We consider finite Petri nets only in this paper. We first introduce Petri net terminology, then define the logics we consider and interpret on Petri nets. Last, we give definitions for "$\models$", "$\models$ fairly" and related notions.

Petri Net Definitions
A *Petri net* N is a triple (S, T, W) where S and T are disjoint sets and $W : ((S \times T) \cup (T \times S)) \to \mathbb{N}$. An element s of S is called a *place* and $t \in T$ is called a *transition*. The function W defines weighted arcs between places and transitions. A Petri net is *finite* iff S and T are finite sets.

The *preset* of $s \in S$ is ${}^\bullet s = \{t \in T \mid W(t, s) > 0\}$ and the *postset* of s is $s^\bullet = \{t \in T \mid W(s, t) > 0\}$, analogously ${}^\bullet t$ and $t^\bullet$ are defined.

A *marking* of a net N is a function $M : S \to \mathbb{N}$, which assigns a number of *token* to each place. With a given order on the places, $s_1, ..., s_n$, M can be represented as a vector in $\mathbb{N}^{|S|}$, where the i-th component is $M(s_i)$.

A transition $t \in T$ is *enabled* at marking M, $M[t\rangle$, iff $\forall s \in {}^\bullet t : M(s) \geq W(s, t)$. If t is enabled it can *fire*. The firing of t generates a new marking M', $M[t\rangle M'$, which is determined by the *firing rule* as $M'(s) = M(s) - W(s, t) + W(t, s), \forall s \in S$. The definition of $[\rangle$ is extended to transition sequences σ as follows. A marking M always enables the empty firing sequence ε and its firing generates M. M enables a transition sequence σt, $M[\sigma t\rangle$, iff $M[\sigma\rangle M'$ and $M'[t\rangle$. If $M[\sigma\rangle$, the transition sequence σ is called a *firing sequence* from M. The function $\mathbf{Fs}_N$ associates with a marking M the set of firing sequences from M, $\mathbf{Fs}_N(M)$. Given a firing sequence $\sigma = t_1 t_2 ...$ with $M[t_1\rangle M_1[t_2\rangle M_2 ...$, the sequence $M M_1 M_2 ...$ is called the *marking sequence* of σ from M, $\mathcal{M}(M, \sigma)$. A marking M is called *final* iff there is no nonempty firing sequence from M. A firing sequence σ from M is *maximal* iff either σ is of infinite length or σ generates a final marking. The function $\mathbf{Fs}_{N,max}$ associates with a marking M the set of maximal firing sequences from M. A marking sequence $\mathcal{M}(M, \sigma)$ is *maximal* iff σ is a maximal firing sequence. By convention, we regard a finite maximal marking sequence μ as equivalent to the infinite marking sequence μ' that repeats the final marking of μ infinitely often.

We denote $X^* \cup X^\omega$ as X^∞ for a set X. For a finite sequence $\gamma = x_1 x_2 ... x_n \in X^\infty$, $|\gamma|$ is n, the length of γ. If γ is infinite, $|\gamma| = \omega$. $\gamma(i)$ denotes the i-th element and γ^i denotes the suffix of γ that truncates the first i positions of γ.

A Petri net $\Sigma = (N, M_0)$ with a designated initial marking M_0 is called a *marked Petri net*. A marking of Σ is *reachable* if there is a firing sequence from M_0 that generates M, $M_0[\sigma\rangle M$. The set of reachable markings of Σ is denoted as $[M_0\rangle$. In the following we will use N synonymous with (S, T, W) and Σ synonymous with (N, M_0).

The Logics

We study a Petri net and its slices for its behaviour with respect to temporal logics, namely CTL* and its sublogics CTL and LTL. To simplify proofs we define the semantics on Petri nets directly instead of considering transition systems. The following definition of CTL* is parameterised by the function **Fs**, which associates with a marking M a set of transition sequences. **Fs** depends on the net semantics (c.f. next section), e.g. $\mathbf{Fs}(M)$ may be the set of maximal or fair firing sequences from M.

We will refer to a path formula as ψ and to a state formula as ϕ, to formulas that may be either as φ.

Definition 1. *CTL^*, CTL, LTL*

Let Σ be a marked Petri net. Let $AP \subseteq S \times \mathbb{N}$ be the set of atomic propositions and $\mathbf{Fs} : \mathbb{N}^{|S|} \to 2^{(T^\infty)}$ a function that associates with a marking M a set of transition sequences. A CTL* *formula is a state formula of the following syntax:*

Every atomic proposition $(s, x) \in AP$ is a state formula.

If ϕ_1 and ϕ_2 are state formulas, then $\neg\phi_1$, $\phi_1 \vee \phi_2$ are state formulas.

If ψ is a path formula, $\mathbf{E}\psi$ is a state formula.

If ϕ is a state formula, $\mathbf{D}\phi$ is a path formula.

If ψ_1 and ψ_2 are path formulas, so are $\neg\psi_1$, $\mathbf{X}\psi_1$, $\psi_1 \vee \psi_2$ and $\psi_1\mathbf{U}\psi_2$.

CTL* Semantics:

$$
\begin{aligned}
\|(s,x)\| &= \{M \in [M_0\rangle \mid M(s) = x\}\\
\|\neg\phi_1\| &= [M_0\rangle \setminus \|\phi_1\|\\
\|\phi_1 \vee \phi_2\| &= \|\phi_1\| \cup \|\phi_2\|\\
\|\mathbf{E}\psi_1\| &= \{M \in [M_0\rangle \mid \exists\, \sigma \in \mathbf{Fs}(M) : M[\sigma\rangle \wedge \mathcal{M}(M,\sigma) \in \|\psi_1\|\}\\
\|\mathbf{D}\phi_1\| &= \{\mu \in (\mathbb{N}^{|S|})^\omega \mid \mu(1) \in \|\phi_1\|\}\\
\|\neg\psi_1\| &= (\mathbb{N}^{|S|})^\omega \setminus \|\psi_1\|\\
\|\psi_1 \vee \psi_2\| &= \|\psi_1\| \cup \|\psi_2\|\\
\|\mathbf{X}\psi_1\| &= \{\mu \in (\mathbb{N}^{|S|})^\omega \mid \mu^1 \in \|\psi_1\|\}\\
\|\psi_1\mathbf{U}\psi_2\| &= \{\mu \in (\mathbb{N}^{|S|})^\omega \mid \exists i,\, 0 \le i : \mu^i \in \|\psi_2\| \wedge \forall j,\, 0 \le j < i : \mu^j \in \|\psi_1\|\}
\end{aligned}
$$

We use the following abbreviations: true $\equiv (s,1) \vee \neg(s,1)$, $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\mathbf{F}\psi \equiv true\mathbf{U}\psi$, $\mathbf{A}\psi \equiv \neg\mathbf{E}(\neg\psi)$ and $\mathbf{G}\psi \equiv \neg\mathbf{F}(\neg\psi)$.

A CTL *formula is a state formula of the following syntax: If ϕ_1 and ϕ_2 are state formulas as defined above, then $\mathbf{X}(\mathbf{D}\phi_1)$, $\mathbf{G}(\mathbf{D}\phi_1)$, and $(\mathbf{D}\phi_1)\mathbf{U}(\mathbf{D}\phi_2)$ are path formulas.*

An LTL *formula is a path formula of the following syntax: If $(s, x) \in AP$, then $\mathbf{D}(s, x)$ is a path formula. If ψ_1 and ψ_2 are path formulas, then $\neg\psi_1$, $\psi_1 \wedge \psi_2$, $\mathbf{X}\psi_1$, and $\psi_1\mathbf{U}\psi_2$ are path formulas.*

A CTL$^*_{-\mathrm{X}}$ (*CTL$_{-\mathrm{X}}$*/LTL$_{-\mathrm{X}}$) *formula is a CTL* (CTL/LTL) formula built without using the* **X** *operator.*

For a path formula ψ, $\|\psi\|$ contains marking sequences that may or may not be generated by a firing sequence of the net. But since we only interpret state formulas on Σ (c.f. Def. 4) the path qualifier **E** makes sure that only realisable marking sequences are considered.

The function *scope* associates with every CTL* and LTL formula φ the set of places referred to by the atomic propositions used in φ.

Petri Net Semantics
In the following we define when a Petri net Σ satisfies an LTL or CTL$^*_{-X}$(and hence a CTL) formula. Intuitively, a Petri net satisfies a formula φ if its behaviour satisfies φ. We consider as behaviour of Σ its firing sequences that are maximal or fair w.r.t. T' (c.f. Def. 3).

Definition 2. *permanently enabled*
Let $\sigma = t_1 t_2 ...$ be a firing sequence of Σ with $M_i[t_{i+1}\rangle M_{i+1}, \forall i,\ 0 \leq i < |\sigma|$.
σ permanently enables *$t \in T$ iff*
either σ is finite and $M_{|\sigma|}[t\rangle$ or σ is infinite and $\exists i,\ 0 \leq i : \forall j,\ i \leq j : M_j[t\rangle$.

Definition 3. *Fairness with respect to T'*
Let T' be a subset of T, let $\sigma = t_1 t_2 t_3 ...$ be a maximal firing sequence of Σ and M_i be the markings with $M_i[t_{i+1}\rangle M_{i+1}, \forall i,\ 0 \leq i < |\sigma|$.
σ is fair w.r.t. *T' iff*
– either σ is finite
– or σ is infinite, and, if there is a $t \in T'$ it permanently enables, it then fires infinitely often some transition of T' (which may or may not be t itself).

We also say "Σ is fair w.r.t. T'" to express that we only consider firing sequences of Σ that are fair w.r.t. T'. The function $\mathbf{Fs}_{N,fair(T')}$ associates with a marking M the set of firing sequences σ that are fair w.r.t. T'. To define "$\Sigma \models \varphi$" we consider the set of all maximal firing sequences, which include the firing sequences that are fair w.r.t. T', for "$\Sigma \models \varphi$ fairly w.r.t. T'" we only consider firing sequences that are fair w.r.t. T':

Definition 4. *$\Sigma \models \varphi$ (fairly w.r.t. T')*
Let $T' \subseteq T$ be a set of transitions of Σ. Let ϕ be a CTL formula. $\Sigma \models \phi$ iff $M_0 \in \|\phi\|$ and* **Fs** *of Def. 1 is the function $\mathbf{Fs}_{N,max}$. $\Sigma \models \phi$* fairly w.r.t. *T' iff $M_0 \in \|\phi\|$ and* **Fs** *of Def. 1 is the function $\mathbf{Fs}_{N,fair(T')}$.*
Let ψ be an LTL formula. $\Sigma \models \psi$ iff $\Sigma \models \mathbf{A}\psi$. $\Sigma \models \psi$ fairly w.r.t. *T' iff $\Sigma \models \mathbf{A}\psi$ fairly w.r.t. T'.*

3 The Slicing Algorithm

The token count of a place p is determined by the firings of incoming and outgoing transitions of p. Whether such a transition can fire, depends on the token count of its input places. So given a set of places P, we can iteratively construct a subnet $\hat{\Sigma} = (\hat{S}, \hat{T}, \hat{W}, \hat{M}_0)$ of Σ by taking all incoming and outgoing transitions of a place $s \in \hat{S}$ together with their input places, starting with $\hat{S} = P$. The subnet $\hat{\Sigma}$ captures every token flow of Σ with an effect on the token count of a place $s \in P$. We extend this approach by distinguishing between *reading* and *non-reading* transitions. A reading transition with respect to a set of places R cannot change the token count of any place in R, i.e., t is a reading transition of R iff $\forall s \in R : W(s,t) = W(t,s)$. If t is not a reading transition of R,

we call t a non-reading transition of R. Let us now iteratively build a subnet $\Sigma' = (S', T', W', M'_0)$ by taking all non-reading transitions of a place $s \in S'$ together with their input places, starting with $S' = P$.

Definition 5. *slice, slicing criterion*
Let Σ be a marked Petri net and $P \subseteq S$ a non-empty set, called slicing criterion. *The following algorithm constructs $slice(\Sigma, P)$.*

```
generateSlice (Σ, P){
  T', S_done := ∅;
  S' := P;
  while ( ∃s ∈ (S' \ S_done) ) {
    while ( ∃t ∈ ((•s ∪ s•) \ T') : W(s,t) ≠ W(t,s) ) {
      S' := S' ∪ •t;
      T' := T' ∪ {t}; }
      S_done := S_done ∪ {s}; }
  W' := W|_{T'∪S'};
  M'_0 := M_0|_{S'};
  return (S', T', W', M'_0) }
```

Figure 1 illustrates the effect of `generateSlice`. The net $\Sigma' = slice(\Sigma, P)$ also captures every token flow with an effect on the token count of any $s \in P$ but Σ' may be smaller than $\hat{\Sigma}$, the subnet constructed without considering reading transitions. Even for certain strongly connected nets Σ' may be smaller than Σ, whereas $\hat{\Sigma}$ will equal Σ. As illustrated in Fig. 2, $slice(\Sigma, P)$ is a subnet that evolves without input of the remaining net: Reading transitions are the only incoming transitions.

Note, that given two slicing criteria $P_1 \subseteq P_2$, $slice(\Sigma, P_1)$ is a subnet of $slice(\Sigma, P_2)$, that means $slice(\Sigma, P_1)$ is the smallest *slice* that contains P_1. In the following we denote with Σ' the slice of a given net Σ for a slicing criterion

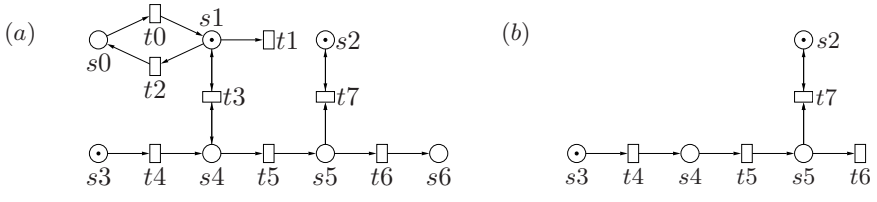


Fig. 1. (a) $\Sigma_1 = (N_1, M_1)$ (b) $slice(\Sigma_1, \{s5\}) = (N'_1, M'_1)$

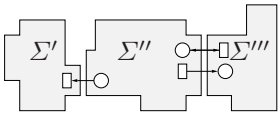


Fig. 2. Σ has (at least) four possible slices Σ'', $\Sigma'\Sigma''$, $\Sigma''\Sigma'''$ and Σ itself

$P \subseteq S$, and if we interpret a formula φ on a net Σ, we assume that $scope(\varphi) \subseteq S'$. So Σ' is not necessarily the smallest possible slice, $slice(\Sigma, scope(\varphi))$.

3.1 Firing Sequences of the *slice*

In this section we concentrate on the correspondence between Σ and Σ', which we define below by means of the function *slice*. We study the correspondence between (sets of) firing sequences of Σ and Σ'. From a correspondence between maximal firing sequences we can bridge to a correspondence between their marking sequences, as we shall see. In Sect. 3.2 we examine what this correspondence means for the set of formulas that Σ and Σ' satisfy.

The function *slice* helps us to describe the correspondences between Σ and Σ'. It projects markings, marking/firing sequences of a net Σ onto markings, marking/firing sequences of its slices.

Definition 6. $slice_{(N,N')}$
Let N and N' be two Petri nets with $T' \subseteq T$ and $S' \subseteq S$. We define $slice_{(N,N')} = slice^{Tr}_{(T,T')} \cup slice^{Mkg}_{(S,S')} \cup slice^{Msq}_{(S,S')}$, where

$slice^{Tr}_{(T,T')} \in [T^\infty \to T'^\infty]$ maps a sequence of transitions σ onto the transition sequence σ' where σ' is derived from σ by omitting every transition $t \in T \setminus T'$,

$slice^{Mkg}_{(S,S')} \in [\mathbb{N}^{|S|} \to \mathbb{N}^{|S'|}]$ maps a marking M of N onto the marking M' of N' with $M' = M|_{S'}$, and

$slice^{Msq}_{(S,S')} \in [(\mathbb{N}^{|S|})^\infty \to (\mathbb{N}^{|S'|})^\infty]$ is a function on sequences of markings. Let $M_0M_1M_2...$ be a sequence of markings. $slice^{Msq}_{(S,S')}(M_0M_1...)$ is defined as $unstutter(slice^{Mkg}_{(S,S')}(M_0)\ slice^{Mkg}_{(S,S')}(M_1)\ ...)$. The function $unstutter(\mu)$ replaces any finite sequence of a marking M in μ by a single occurrence of M but leaves an infinite sequence of M unchanged.

We omit the indices referring to the nets and simply denote the function as *slice*, if this does not cause ambiguities. We extend the function *slice* to sets in the usual way: $slice(X) = \bigcup_{x \in X} \{slice(x)\}$, $X \subseteq T^\infty \cup (\mathbb{N}^{|S|})^\infty$.

We start with two simple observations: The occurrence of a transition $t \in T \setminus T'$ cannot change the token count of any place in S' whereas the occurrence of a transition $t \in T'$ changes the token count of at least one place in S'. A marking M of Σ enables a transition $t \in T'$ if and only if a marking $M' = slice(M)$ of Σ' enables t, since transitions in T' have the same input places in Σ and Σ'.

Whatever one of the nets can do to a marking on S', the other can do as well by firing the same transitions in T' in the same order. So for firing sequences there is a correspondence between Σ and Σ', i.e. $slice(\mathbf{Fs}_N(M_0)) = \mathbf{Fs}_{N'}(M'_0)$.

Proposition 1. *Let σ be a firing sequence and M be a marking of Σ.*
(i) $M_0\ [\sigma\rangle\ M \Rightarrow M'_0\ [slice(\sigma)\rangle\ slice(M)$.
Let σ' be a firing sequence and M' a marking of Σ'.
(ii) $M'_0[\sigma'\rangle M' \Rightarrow \exists M \in \mathbb{N}^{|S|} : M' = slice(M) \wedge M_0[\sigma'\rangle M$.

Proof. We show Prop. 1 by induction on the length l of σ and σ', respectively.

$l = 0$: The initial marking of Σ and Σ' is generated by firing the empty firing sequence ε. By Def. 5 and Def. 6, $M'_0 = M_0|_{S'} = slice(M_0)$.

$l \rightarrow l+1$: First we show (i). Let σt be a firing sequence of Σ of length $l+1$, let M_l, M_{l+1} be markings of Σ with $M_0[\sigma\rangle M_l[t\rangle M_{l+1}$. By the ind. hyp., $M'_0[slice(\sigma)\rangle M'_k$ with $M'_k = slice(M_l)$. If t is an element of T', it follows from $M_l[t\rangle$ that M'_k enables t. By the firing rule and since $slice(M_l) = M_l|_{S'} = M'_k$, it follows that $slice(M_{l+1}) = M'_{k+1}$. If $t \in T \setminus T'$, $slice(\sigma) = slice(\sigma t)$ and thus $M'_0[slice(\sigma t)\rangle M'_k$. A transition in $T \setminus T'$ cannot change the token count of any place $s \in S'$, thus $slice(M_{l+1}) = slice(M_l)$.

For (ii) let $\sigma' t$ be a firing sequence of Σ' with length $l+1$. Let M'_l and M'_{l+1} be the markings of Σ' with $M'_0[\sigma'\rangle M'_l[t\rangle M'_{l+1}$. Let M_l be the marking of Σ with $M_0[\sigma'\rangle M_l$ and $slice(M_l) = M'_l$, which exists by the ind. hyp.. M_l enables t. Again by the firing rule, $slice(M_{l+1}) = M'_{l+1}$. □

Whatever maximal firing sequence the slice Σ' may fire, Σ can fire a corresponding maximal firing sequence, but not vice versa.

Proposition 2. *Let σ'_m be a maximal firing sequence of Σ'.*

There is a maximal firing sequence σ_m of Σ that starts with σ'_m and for which $slice(\sigma_m) = \sigma'_m$ holds.

Proof. By Prop. 1 (ii), σ'_m is a firing sequence of Σ. In case σ'_m is infinite, it is also a maximal firing sequence of Σ. So let σ'_m be finite. Let σ_m be a maximal firing sequence of Σ with $\sigma_m = \sigma'_m\sigma$ where $\sigma \in T^\infty$. Let σ' be the transition sequence with $\sigma' = slice(\sigma_m) = \sigma'_m slice(\sigma)$. By Prop. 1 (i), σ' is a firing sequence of Σ'. Since σ'_m is maximal, it follows that $slice(\sigma) = \varepsilon$. □

t_4 is not a maximal firing sequence of $slice(\Sigma_1, \{s5\})$ but it is the slice of Σ_1's maximal firing sequence $t4\ t2\ t0\ t2\ t0\ \ldots$. So for maximal firing sequences $slice(\mathbf{Fs}_{N,max}(M_0)) \supset \mathbf{Fs}_{N',max}(M'_0)$. We get a two-way correspondence if we assume that Σ is fair, i.e. $slice(\mathbf{Fs}_{N,fair(T')}(M_0)) = \mathbf{Fs}_{N',max}(M'_0)$.

Proposition 3. *Let σ' be a maximal firing sequence of Σ'.*

(i) There is a firing sequence σ of Σ that is fair w.r.t. T', starts with σ' and $slice(\sigma) = \sigma'$.

Let σ be a firing sequence of Σ, that is fair w.r.t. T'.

(ii) $slice(\sigma)$ is a maximal firing sequence of Σ'.

Proof. Let $M_i(M'_i)$ be the marking of $\Sigma(\Sigma')$ generated by firing the first i transitions of $\sigma(\sigma')$. We first show (i) : By Prop. 1 (ii), σ' is a firing sequence of Σ. If σ' is infinite, it is fair w.r.t. T'. So let σ' be finite. As σ' is maximal, $M'_{|\sigma'|}$ does not enable transitions of T', by Prop. 1, $M_{|\sigma'|}$ does not either. Let $\sigma_2 \in (T \setminus T')^\infty$ be such that $\sigma = \sigma'\sigma_2$ is a maximal firing sequence of Σ, which exists by Prop. 2. Transitions of σ_2 cannot change the token count of places in S', thus σ is slice-fair. We now show (ii): By Prop. 1 (i), $\sigma' = slice(\sigma)$ is a firing sequence of Σ'. If σ' is not maximal, there is a $t' \in T'$ with $M'_{|\sigma'|}[t'\rangle$. Let σ_1 be the smallest prefix of σ with $slice(\sigma_1)$ equals σ'. By Prop. 1 (i), $slice(M_{|\sigma_1|}) = M'_{|\sigma'|}$. So

$M_{|\sigma_1|}[t'\rangle$. After firing σ_1, σ does not fire any $t \in T'$ and thus the token count of places in S' is not changed and t' stays enabled. So σ is not fair w.r.t. T'. □

By the following proposition we can translate a correspondence between maximal firing sequences into a correspondence of their marking sequences.

Proposition 4. *Let M be a marking of Σ and σ be a maximal firing sequence from M such that $slice(\sigma)$ is a maximal firing sequence of Σ' from $slice(M)$.*
$slice(\mathcal{M}(M,\sigma)) = \mathcal{M}(slice(M), slice(\sigma))$

Proof. We give a brief sketch of proof. By Prop. 1, σ and $slice(\sigma)$ have the same effect on S'. The function *slice* removes from $\mathcal{M}(M,\sigma)$ all finite sequences of markings that are identical on S' and a $t \in T \backslash T'$ does not change the token count on S', but a transition $t \in T'$ changes the token count of at least one place $s \in S'$ and $slice(\sigma)$ removes all $t \in T \setminus T'$. Since $\mathcal{M}(M,\sigma)$ and $\mathcal{M}(slice(M), slice(\sigma))$ are maximal marking sequences, they are both infinite. □

We restate the main results regarding the correspondence between Σ and Σ': $slice(\mathbf{Fs}_N(M_0)) = \mathbf{Fs}_{N'}(M'_0)$, $slice(\mathbf{Fs}_{N,max}(M_0)) \supset \mathbf{Fs}_{N',max}(M'_0)$ and finally, $slice(\mathbf{Fs}_{N,fair(T')}(M_0)) = \mathbf{Fs}_{N',max}(M'_0)$. Also, marking sequences of corresponding maximal firing sequences on Σ and Σ', correspond as well.

3.2 Verification and Falsification Results

Now our main results: For Σ that is fair w.r.t. T' and a $\mathrm{CTL}^*_{-\mathrm{X}}$ formula ϕ, we can derive whether or not $\Sigma \models \phi$ by examining Σ'. For next-time, we can derive from $\Sigma' \not\models \psi$ that $\Sigma \not\models \psi$ fairly w.r.t. T' but only for LTL formulas. We show the contraposition in Theorem 2.

Theorem 1. *Let ϕ be a CTL^*_{-X} formula with $scope(\phi) \subseteq S'$.*
$\Sigma \models \phi$ *fairly w.r.t.* $T' \Leftrightarrow \Sigma' \models \phi$

We only sketch the proof of Theorem 1 very briefly due to the lack of space and since the rather lengthy proof can be omitted without loosing important insights about the relation of Σ and Σ'. First it is shown that markings and marking sequences with the same slice are equivalent w.r.t. $\mathrm{CTL}^*_{-\mathrm{X}}$ formulas. Therewith we can show that $slice(M)$ satisfies ϕ on Σ' iff M satisfies ϕ on Σ.

Consider Σ_1 and the CTL formula $\phi_1 = \mathbf{EX}(\mathbf{EX}(\mathbf{D}(s4,1)))$. Σ_1 may fire $t_2\,t_4$, to satisfy $\mathbf{XXD}(s4,1)$, then $t5\,t6$ for fairness w.r.t T', then infinitely often $t_0\,t_2$ for maximality. $slice(\Sigma_1, \{s5\})$ has to fire $t4\,t5$ first. Thus it does not satisfy ϕ_1. This shows, Theorem 1 cannot be extended for next-time for CTL (CTL^*).

Using the next-time operator it is possible to specify a condition to be true at a certain point in the net evolution as a position within a sequence of markings. Since slicing aims at building a reduced model, which should not reflect every state(=marking) change of the original system, the restriction to $\mathrm{CTL}^*_{-\mathrm{X}}$ formulas without next-time is intrinsic of slicing.

$\psi_1 = \mathbf{X}(\mathbf{D}(s4,1))$ shows that "$\Leftarrow$" of Theorem 1 cannot be extended for LTL with next-time: $slice(\Sigma_1, \{s5\}) \models \psi_1$ but $\Sigma_1 \not\models \psi_1$. We now show that "$\Rightarrow$" of Theorem 1 holds for LTL with next-time.

The marking sequence of a maximal firing sequence σ' satisfies ψ on Σ' iff the marking sequence of its corresponding maximal firing sequence σ satisfies ψ on Σ, given σ starts with σ'. As we consider next-time, this restriction is necessary.

Proposition 5. *Let ψ be an LTL formula such that $scope(\psi) \subseteq S'$. Let M be a marking of N. Let σ be a maximal firing sequence from M, such that $slice(\sigma)$ is a maximal firing sequence from $slice(M)$ and σ starts with $slice(\sigma)$.*

$\mathcal{M}(M,\sigma) \in \|\psi\| \Leftrightarrow slice(\mathcal{M}(M,\sigma)) \in \|\psi\|'$.

Proof. Let $\mathcal{M}(M,\sigma)$ be μ and $slice(\mathcal{M}(M,\sigma))$ be μ'. Note, a suffix σ^i is a maximal firing sequence from $\mu(i+1)$ which starts with $slice(\sigma^i)$, also $\mu^i = \mathcal{M}(\mu(i+1),\sigma^i)$, and by Prop. 4, $slice(\mu^i) = \mathcal{M}(slice(\mu(i+1)), slice(\sigma^i))$, $\forall i,\ 0 \leq i < |\sigma|+1$.

$\psi = \mathbf{D}(s,x)$: Since the satisfiability of ϕ depends on the initial marking of s only and $\{s\} = scope(\phi) \subseteq S' \subseteq S$, both directions hold.

$\psi = \neg\psi_1$ and $\psi = \psi_1 \wedge \psi_2$ follow directly by the ind. hyp..

$\psi = \mathbf{X}\psi_1$: Let $\mu = MM_1... \in \|\mathbf{X}\psi_1\|$. Hence $\mu^1 \in \|\psi_1\|$. By the ind. hyp., $slice(\mu^1) \in \|\psi_1\|'$. So we need to show that $slice(\mu) = slice(M)slice(\mu^1)$, i.e. that there is a marking preceeding $slice(\mu^1)$. If the first transition of σ is in T', $slice(\sigma) = \sigma(1)slice(\sigma^1)$ and $slice(M) \neq slice(M_1)$. Thus $slice(\mu) = slice(M)slice(\mu^1)$. If $\sigma(1) \in T \setminus T'$, $slice(\sigma) = \varepsilon$. Since $slice(\sigma)$ is a maximal firing sequence from $slice(M)$, $slice(\mu) = (slice(M))^\omega = slice(M)slice(\mu^1)$.

Let $\mu' \in \|\mathbf{X}\psi_1\|'$, so $\mu'^1 \in \|\psi_1\|'$. By the ind. hyp., $\mu^1 \in \|\psi_1\|$.

$\psi = \psi_1\mathbf{U}\psi_2$: If $\mu \in \|\psi\|$, then $\exists i \geq 0 : \mu^i \in \|\psi_2\| \wedge \forall j, 0 \leq j < i : \mu^j \in \|\psi_1\|$. By the ind. hyp., $slice(\mu^i) \in \|\psi_2\|'$ and $\forall j, 0 \leq j < i : slice(\mu^j) \in \|\psi_1\|'$. Since $slice(\mu) = \mu'$ there is an index i_2 with $slice(\mu^i) = \mu'^{i_2}$ and $\forall j, 0 \leq j < i_2 : \exists j_2 < i : \mu'^j = slice(\mu^{j_2})$.

$\mu' \in \|\psi\|'$ implies that $\exists i \geq 0 : \mu'^i \in \|\psi_2\|' \wedge \forall j, 0 \leq j < i : \mu'^j \in \|\psi_1\|'$. Let i_2 be the smallest index with $slice(\mu^{i_2}) = \mu'^i$. Since i_2 is minimal, $\forall j_2, 0 \leq j_2 < i_2 : \exists j, 0 \leq j < i : slice(\mu^{j_2}) = \mu'^j$. By the ind. hyp. follows that $\mu \in \|\psi_1\mathbf{U}\psi_2\|$. □

Theorem 2. *Let ψ be an LTL formula with $scope(\psi) \subseteq S'$.*

$\Sigma \models \psi$ *fairly w.r.t.* $T' \Rightarrow \Sigma' \models \psi$.

Proof. If $\Sigma' \not\models \psi$, there is a maximal firing sequence σ' with $\mathcal{M}(M'_0,\sigma') \notin \|\psi\|'$. By Prop. 3 (i), there is a firing sequence σ that is fair and starts with $slice(\sigma) = \sigma'$, so $\mathcal{M}(M_0,\sigma) \notin \|\psi\|$ by Prop. 5. Hence $\Sigma \not\models \psi$ fairly w.r.t. T'. □

If "$\Sigma \not\models \psi$ fairly w.r.t. T'" holds, then there is a firing sequence σ that does not satisfy ψ, is fair w.r.t. T' and hence also maximal. So "$\Sigma' \not\models \psi$" implies also that "$\Sigma \not\models \psi$".

Again we shortly summarise: For next-time, we can only falsify LTL formulas. We can verify and falsify $\mathrm{CTL}^*_{-\mathrm{X}}$ if the modelled system behaves fairly w.r.t. T'. In all scenarios it suffices to examine Σ' without fairness assumptions.

4 Example and Test Results

To illustrate our approach, we analyse the Petri net of Fig. 3, which models the workflow of a business process for dealing with insurance claims as in [10]. An incoming claim is recorded first. A claim may be accepted or rejected, depending on the insurance cover. For a rejected claim, a rejection letter is written. If the claim is accepted, emergency measures, if necessary, are provided. After an assessment -possibly done by an expert- a settlement is offered to the customer, who may either accept or reject. A rejected offer may be followed by legal proceedings or a revision. If a settlement is agreed upon, money is paid.

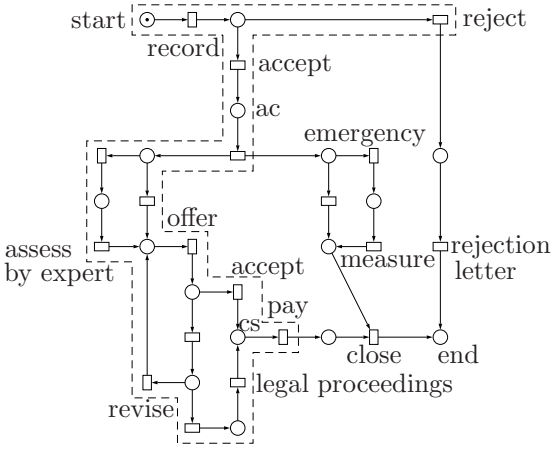


Fig. 3. The net Σ_2 modelling an insurance claim process

We want to verify that every accepted claim is settled, i.e. $\phi = \mathbf{AG}(\mathbf{D}(ac, 1) \Rightarrow \mathbf{F}\,\mathbf{D}(cs, 1))$. The *slice* of Σ_2 for $\{ac, cs\}$ is the subnet within the dashed borders. In Σ_2 29 states(=markings) via 60 state transitions are reachable, whereas $slice(\Sigma_2, \{ac, cs\})$ has 11 states and 14 state transitions only.

The example illustrates that the slicing algorithm works well for workflow nets, since they usually are not strongly-connected. As a benchmark for the effect of slicing on other nets, we used the set of Promela examples of [11], which is a collection of "standard examples from the concurrency analysis literature" [11]. The set consists of 75 examples, some of which are scaled up instances of the same problem type, e.g. 6, 8, 10 and 12 dining philosophers. All in all there are 23 different types of examples. For the experiments the biggest instance of each scalable type has been removed to reduce the overall computation time. For each place of the net a slice has been generated. Since we do not provide a meaningful slicing criterion, we filter the generated slices. Slices with less than 20 states or 10% of the states are considered to be uninteresting. We also consider slices with more than 85% of the states and the state transitions as too big. Of

the 23 types 6 had nice slices, i.e. slices with less than 85% of the states or the state transitions and more than 20 states and 10% of the states. In average they covered 67% of the places (median 69%). It is perhaps worth mentioning that only one type, the `dining philosophers (dp)`, has a strongly connected net and for the example `dartes` the time limit was exceeded.

5 Conclusions, Related Work and Future Work

This paper introduces Petri net slicing to reduce the size of a net in order to alleviate the state explosion problem for model checking Petri nets. The slicing algorithm is based on the observation that the token count of a place s is determined by the firings of non-reading transitions connected with s and these are determined by the token count of their input places. A *slice* allows falsification and verification of a $\mathrm{CTL}^*_{-\mathrm{X}}$ formula if $scope(\phi)$ is a subset of the slice's states. For verification we need to assume that Σ is fair w.r.t. the set T' of transitions of the slice. LTL formulas using next-time can be falsified. The present algorithm constructs smaller slices for certain strongly connected nets, but is expected to be most useful for nets, that are not strongly connected like workflow nets.

Our approach is quite general by imposing very weak restrictions on the formulas (CTL^* without next) and little restrictions on the net in terms of fairness assumptions. Nevertheless, our results show that slicing is a technique that can help to alleviate the state explosion problem for model checking Petri nets.

There are several groups working on slicing and on slicing as a method to tackle the state-explosion problem for model-checking in particular. For program slicing Dwyer and Hatcliff show in [9] that a program P and its program slice P' either both satisfy ϕ or do not satisfy ϕ given the set of atomic propositions of an $\mathrm{LTL}_{-\mathrm{X}}$ formula ϕ is the slicing criterion. Cone-of-influence reduction [12], a similar technique used in hardware verification, constructs a reduced model P' for the set of atomic propositions of a formula ϕ and guarantees that the reduced model P' satisfies ϕ if and only if the original model P does. In [6] Brückner develops a technique for slicing CSP-OZ specifications. Brückner and Wehrheim show in [7] the correctness of their approach to slicing Object-Z specifications.

We are investigating two ideas for the development of algorithms that allow for more aggressive slicing. One approach is the generalisation of reading transitions to reading subnets that may temporarily remove tokens. Another approach is to identify subnets that play the role of data and control flow, so that the concept of relevant variables, as defined in [1], is applicable.

Acknowledgements. I would like to thank Eike Best, Hans Fleischhack and Harro Wimmel for plenty of feedback. I am grateful to Javier Esparza and Maciej Koutny for reading and commenting an earlier version of this paper. Thanks also for the feedback I received while presenting a first version of this paper at the "Formal Approaches to Business Processes and Web Services" workshop [13].

References

1. Weiser, M.: Program slicing. In: Proceedings of the 5th international conference on Software engineering, pp. 439–449. IEEE Press, Piscataway (1981)
2. Tip, F.: A survey of program slicing techniques. Journal of programming languages 3, 121–189 (1995)
3. Sloane, A.M., Holdsworth, J.: Beyond traditional program slicing. In: International Symposium on Software Testing and Analysis, San Diego, CA, pp. 180–186. ACM Press, New York (1996)
4. Heimdahl, M.P.E., Whalen, M.W.: Reduction and slicing of hierarchical state machines. In: Jazayeri, M. (ed.) ESEC 1997 and ESEC-FSE 1997. LNCS, vol. 1301, pp. 450–467. Springer, Heidelberg (1997)
5. Chang, J., Richardson, D.J.: Static and dynamic specification slicing. In: Proceedings of the Fourth Irvine Software Symposium (1994)
6. Brückner, I.: Slicing CSP-OZ specifications. In: Nordic Workshop on Programming Theory (2004)
7. Brückner, I., Wehrheim, H.: Slicing Object-Z specifications for verification. In: Treharne, H., King, S., Henson, M.C., Schneider, S. (eds.) ZB 2005. LNCS, vol. 3455, pp. 414–433. Springer, Heidelberg (2005)
8. Chang, C.K., Wang, H.: A slicing algorithm of concurrency modeling based on Petri nets. In: Hwang, K., Jacobs, S.M., Swartzlander, E.E. (eds.) Proc. of the 1986 Int. Conf. on Parallel Processing, Washington, pp. 789–792. IEEE Computer Society Press, Los Alamitos (1987)
9. Hatcliff, J., Dwyer, M.B., Zheng, H.: Slicing software for model construction. Higher-Order and Symbolic Computation, 315–353 (2000)
10. Aalst, W.v.d., Hee, K.v.: Workflow Management - Models, Methods, and Systems, pp. 4–62. The MIT Press, Cambridge (2002)
11. Corbett, J.C.: Evaluating Deadlock Detection Methods for Concurrent Software. IEEE Transactions on Software Engineering 22(3), 161–180 (1996)
12. Berezin, S., Campos, S., Clarke, E.M.: Compositional reasoning in model checking. In: de Roever, W.-P., Langmaack, H., Pnueli, A. (eds.) COMPOS 1997. LNCS, vol. 1536, pp. 81–102. Springer, Heidelberg (1998)
13. Rakow, A.: Slicing Petri Nets. In: Proceedings of the Workshop on FABPWS 2007, Satellite Event, Siedlce, pp. 56–70 (2007)

Lower Bound for the Length of Synchronizing Words in Partially-Synchronizing Automata

Adam Roman and Wit Foryś

Institute of Computer Science, Jagiellonian University
Cracow, Poland

Abstract. We introduce the generalized notion of automata synchronization, so called partial synchronization, which holds for automata with partial transition function. We give a lower bound for the length of minimal synchronizing words for partial synchronizing automata. The difference, in comparison to the 'classical' synchronization, lies in the initial conditions: let $\mathcal{A} = (Q, A, \delta)$ be an automaton representing the dynamics of a particular system. In case of partial synchronization we assume that initial conditions (initial state of the system) can be represented by some particular states, that is by some $P \subset Q$, not necessarily by all possible states from Q. At first glance the above assumption limits our room for manoeuvre for constructing possibly long minimal synchronizing words (because of the lower number of states at the beginning). Unexpectedly this assumption allows us to construct longer minimal synchronizing words than in a standard case. In our proof we use Sperner's Theorem and some basic combinatorics.

1 Introduction

We define *an automaton* as a triple $\mathcal{A} = (Q, A, \delta)$, where Q is a finite set of states, A is a finite alphabet and $\delta : Q \times A \to Q$ is a partial function transforming states. It can be extended on the free monoid A^* and the set of subsets of Q:

$$\forall a \in A\ \forall w \in A^*\ \forall P \subseteq Q\ \delta(P, \varepsilon) = P,\ \delta(P, aw) = \bigcup_{p \in P} \{\delta(\delta(p, a), w)\},$$

where ε is an empty word of length 0. We say that $w \in A^*$ is a *carefully synchronizing word* (**csw**) for $\mathcal{A} = (Q, A, \delta)$ (or w *carefully synchronizes* $\mathcal{A}$) if $\delta(Q, w)$ is well defined (that is, if $\delta(q, w)$ is defined for all $q \in Q$) and $|\delta(Q, w)| = 1$. If w is the shortest word among all **csw**s, we call w a *minimal carefully synchronizing word* (**mcsw**). By $CSyn(\mathcal{A})$ we denote the set of all words which carefully synchronize $\mathcal{A}$. The following proposition is straightforward.

Proposition 1. *Let $\mathcal{A} = (Q, A, \delta)$, $w \in CSyn(\mathcal{A})$. Then $\forall u, v \in A^*$ $uwv \in CSyn(\mathcal{A})$, provided that $\delta(Q, uwv)$ is well defined.*

In the light of Proposition 1 a natural question arises: given n, what is the longest possible **mcsw** for n-state automaton? We denote it's length by $\omega(n)$:

$$\omega(n) = \max_{\mathcal{A}=(Q,A,\delta):|Q|=n} \{ \min_{u \in CSyn(\mathcal{A})} \{|u|\} \}\ .$$

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 448–459, 2008.

The notion of **csw** is a generalized version of well-known synchronizing words: let $\mathcal{A} = (Q, A, \delta)$ be a finite automaton with δ being a *total* function on $Q \times A$. We say that $w \in A^*$ *synchronizes* $\mathcal{A}$ iff $|\delta(Q, w)| = 1$. Such a word is called a *minimal synchronizing word* for $\mathcal{A}$, if there is no shorter one. We denote it's length by $m(\mathcal{A})$. The set of all n-state synchronizing automata is denoted by $Syn(n)$. Synchronizing automata are the object of intensive research because of the famous, unsolved Černý Conjecture (see [4,5]):

Conjecture 1 (Černý, 1964). $\mathcal{A} \in Syn(n) \Longrightarrow m(\mathcal{A}) \leq (n-1)^2$.

Synchronizing and carefully synchronizing automata have many possible applications, mainly in robotics ([1,9,10]), bioinformatics ([2,3]) and network theory ([7]).

Remark 1. Sometimes we will say that 'word w synchronizes P to R'. The word 'synchronization' does not refer here to the process of synchronizing the set of all possible states into a singleton, but to the process of transforming P into R with w such that $R = \delta(P, w)$ and $|R| < |P|$.

2 Previous Results and Generalization

In [6] M. Ito and K. Shikishima-Tsuji have considered the estimation of $\omega(n)$ (they denoted it by $d_3(n)$). They proved the following two theorems:

Theorem 1

$$\omega(n) \leq 2^n - 2^{n-2} - 1 \ .$$

Theorem 2

$$\omega(n) \geq 2^{\frac{n}{2}} + 1, \ \textit{if } n = 2k,$$
$$\omega(n) \geq 3 \cdot 2^{\frac{n-3}{2}} + 1, \ \textit{if } n = 2k+1 \ .$$

In [8] P. Martyugin improved the lower bound:

Theorem 3

$$\omega(n) \geq 3 \cdot 3^{\frac{n}{3}} - 2, \ \textit{if } n = 3k,$$
$$\omega(n) \geq 4 \cdot 3^{\frac{n-1}{3}} - 2, \ \textit{if } n = 3k+1,$$
$$\omega(n) \geq 6 \cdot 3^{\frac{n-2}{3}} - 2, \ \textit{if } n = 3k+2 \ .$$

We will now generalize the notion of synchronization and prove analogical lower bound for the generalized case.

Let $P(k, n)$, $0 < k \leq n$, denotes the set of all n-state automata $\mathcal{A} = (Q, A, \delta)$ for which there exists k-element subset S of Q and $w \in A^*$ such that $\delta(S, w)$ is well defined and $|\delta(S, w)| = 1$. Notice that $P(n+k, n) = \emptyset$ for $k > 0$.

Definition 1. *We say that an n-state automaton $\mathcal{A}$ is k-partially synchronizing if $\mathcal{A} \in P(k, n) \setminus \bigcup_{i=k+1}^{n} P(i, n)$.*

Definition 2. *We say that an n-state automaton $\mathcal{A}$ is partially synchronizing if there exists k such that $\mathcal{A} \in PSyn_k(n)$.*

By $PSyn_k(n)$ we denote the set of all k-partially synchronizing automata with n states. By $PSyn(n)$ we denote all n-state partially synchronizing automata.

The following propositions are straightforward.

Proposition 2

$$\mathcal{A} = (Q, A, \delta) \in Syn(n) \iff \mathcal{A} \in PSyn_n(n) \text{ and } \delta \text{ is a total function}.$$

Proposition 3. *Let $n > 1$ and $\mathcal{A} \in PSyn_k(n)$. Then $\forall 1 \leq l \leq k \;\; \mathcal{A} \in P(l, n)$.*

The motivation for introducing such a generalization arises from the paper of P. Martyugin (see [8], p. 2):

> "It is known ([9,10]) that synchronizing words play an important role in robotics or, more precise, robotic manipulation which deals with part handling problems in industrial automation such as part feeding, loading, assembly and packing. The idea of it is as following. Let some identical parts need to be oriented before assembly. There is a finite number of simple devices which can rotate the parts. The action of a device depends on the orientation of the part. A fragile part in several positions can be destroyed by some devices. We consider an automaton, where the state set is the set of all possible orientations of a part, the alphabet is the set of devices. The partial transition function is defined by non-destroying (or careful) actions of the devices. Then a carefully synchronizing word sets an order of devices use avoiding the destruction of fragile parts while orienting them".

Let $\mathcal{A} = (Q, A, \delta)$ be an automaton and assume that the initial parts orientation is represented by a proper subset P of all possible orientations represented by Q. If $|P| = k$ then the word synchronizing those k initial states into a single state is called a *k-partially synchronizing word* for P.

We can introduce the notion of minimal synchronizing word for partially synchronizing automaton in two ways:

Definition 3. *Let $\mathcal{A} = (Q, A, \delta) \in PSyn_k(n)$. We say that $w \in A^*$ is a m^+-minimal synchronizing word for $\mathcal{A}$ if $|w| = \max_{P \subset Q, |P|=k} \{|v| : \; |\delta(P, v)| = 1\}$.*

Definition 4. *Let $\mathcal{A} = (Q, A, \delta) \in PSyn_k(n)$. We say that $w \in A^*$ is a m_--minimal synchronizing word for $\mathcal{A}$ if $|w| = \min_{P \subset Q, |P|=k} \{|v| : \; |\delta(P, v)| = 1\}$.*

We will give the lower bounds for the length of m^+ and m_--synchronizing words for n-state partially synchronizing automata.

The reason for introducing two slightly different definitions of minimal synchronizing word is due to a specific nature of industrial automation tasks mentioned above. Speaking more precisely: if there is a *fixed set* of states representing

Fig. 1. A single detail

initial part orientations, then in order to find a lower bound for the length of minimal synchronizing word we need to use Definition 3. On the other side, if we know only the *maximal number* of possible initial orientations, we need to use Definition 4.

We illustrate this by the following example. Suppose that a factory produces identical details, like the one presented in Fig. 1. It consists of a single hexagon with four insets. Three of them (black) are tough and blow-resistant. The fourth one (white) is fragile. Hitting it may cause its damage. Details are put on the transmission belt. We assume a detail to be in one of 6 possible orientations. The task is to rotate all the details into one common orientation. To do this, we use 3 kinds of obstacles arranged along the transmission belt. The lower one (l) will rotate a detail 60° right only if there is an inset at the lower hexagon edge. The upper-small (u) will rotate a detail 60° left only if there is an inset at the upper hexagon edge. Finally, the upper-big (U) will rotate a detail 60° left in each case. An example situation is shown in Fig. 2. All possible transitions between orientations are shown in Fig. 3. Notice that transition function is a partial function. Workers, who put details on the transmission belt, can hold them only by one of the tough insets. The situation is shown in Fig 4: the initial orientations allowed are 1, 4, 6. The problem is to find a minimal synchronizing word over $A = \{l, u, U\}$ (representing the sequence of obstacles) for the automaton with $Q = \{1, 2, 3, 4, 5, 6\}$ (representing the set of all possible orientations), provided that the initial set of states is $\{1, 4, 6\}$.

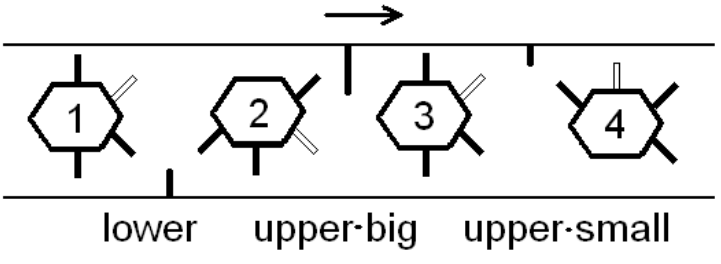


Fig. 2. Examplary transformations. Top arrow indicates the direction of transportation belt movement. Detail in position 1 is transformed by the lower upstacle into position 2. To transform it from position 2 into position 3 we cannot use the upper-small obstacle because there is no inset at the top hexagon edge. We must use the upper-big one. Next, to transform a detail from position 3 into position 4 we cannot use the upper-big obstacle again, because it would destroy the fragile inset; we have to use the upper-small one.

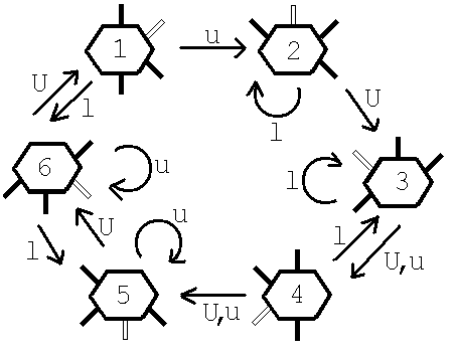


Fig. 3. Orientation transformations using 3 kinds of obstacles: lower (l), upper-small (u) and upper-big (U)

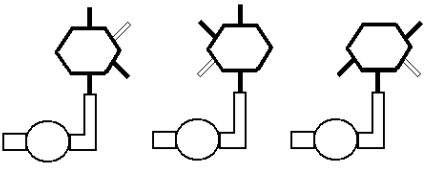


Fig. 4. Three possible ways to put a detail into transmission belt

At first glance the assumption on the cardinality of possible initial states limits our room for manoeuvre for constructing possibly long minimal synchronizing words (because of the lower number of states at the very beginning). Unexpectedly this assumption allows us to construct longer minimal synchronizing words than in standard case. The reason is that we don't need to use a letter a defined on the whole set Q, such that $|\delta(Q, a)| < |Q|$. This, in fact, *increases* our manoeuvre room.

In Fig. 5 all possible 'forward' transitions are shown. One can verify that the shortest word synchronizing $\{1, 4, 6\}$ into $\{5\}$ is $w = uUulUluu$, but $\mathcal{A} \in PSyn_5(6)$, so w is not a m^+-minimal synchronizing word. It is the word $v = lUlUw = lUlUuUulUluu$ and it synchronizes $\{1, 2, 3, 4, 6\}$ to $\{5\}$. In this case m_--minimal synchronizing word equals v.

3 Main Result

Since now, for the sake of simplicity, we assume that the number of states in automaton is an even number. Theorem 5 can be proved for automata with an odd number of states in a very similar way. Let $\omega^+(n)$ ($\omega_-(n)$ resp.) denotes the lower bound for the length of m^+- (resp. m_--) minimal synchronizing word for n-state partially synchronizing automaton. The main result is that $\omega^+(n) \geq \binom{n+1}{\frac{n}{2}} - \frac{n}{2} - 2$ and $\omega_-(n) \geq \omega^+(n) - \binom{n}{\frac{n}{2}} + 1$. These bounds are greater than

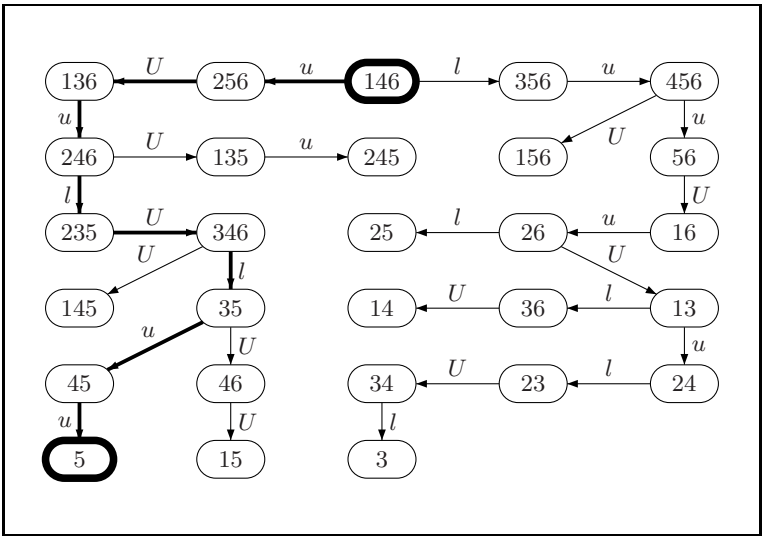


Fig. 5. All possible transitions. The initial subset is $\{1, 4, 6\}$.

all lower bounds for synchronizing and carefully synchronizing words' lengths previously found.

First let us recall the Sperner's Theorem.

Theorem 4 (Sperner). *The maximum cardinality of a collection of subsets of a n-element set X, none of which contains another, is the binomial coefficient* $\binom{n}{\lfloor \frac{n}{2} \rfloor}$.

The idea of our proof is to find possibly long sequence of subsets of Q such that no one of them synchronizes the one following it. We see that for n even the cardinality of the family of all $\frac{n}{2}$-element subsets of X satisfies the above assumption and realizes the maximum cardinality stated in Theorem 4. We will use this family in our main result, Theorem 5.

Theorem 5 (main result). *Let $|Q| = n$, where n is an even number. There exists $\frac{n}{2}$-partially synchronizing automaton $\mathcal{A} = (Q, A, \delta)$ and m^+-minimal synchronizing word $w \in A^*$ for $\mathcal{A}$ such that*

$$|w| = \binom{n+1}{\frac{n}{2}} - \frac{n}{2} - 2\,.$$

The immediate consequence of Theorem 5 are the following Propositions and Theorem.

Proposition 4. *For n even we have $\omega^+(n) \geq \binom{n+1}{\frac{n}{2}} - \frac{n}{2} - 2$.*

Theorem 6. *Let $|Q| = n$, where n is an even number. There exists $\frac{n}{2}$-partially synchronizing automaton $\mathcal{A} = (Q, A, \delta)$ and m_--minimal synchronizing word $w \in A^*$ for $\mathcal{A}$ such that*

$$|w| = \binom{n}{\frac{n}{2}-1} - \frac{n}{2} - 1 \,.$$

Proposition 5. *For n even we have $\omega_-(n) \geq \binom{n}{\frac{n}{2}-1} - \frac{n}{2} - 1$.*

4 Proof of Theorem 5

We construct n-state automaton $\mathcal{A} \in PSyn_{\frac{n}{2}}(n)$ with m^+-minimal synchronizing word of length $\binom{n+1}{\frac{n}{2}} - \frac{n}{2} - 2$. Let $\mathcal{A} = (Q, A, \delta)$, $Q = \{1, 2, ..., n\}$, $A = \{a_1, a_2, ..., a_N\}$, where $N = \binom{n}{\frac{n}{2}}$. Let $<$ be a natural order on Q. For each $P \subset Q$ and $p \in P$ we put $r_P(i) = p \Leftrightarrow i = 1 + |\{s \in P : s < p\}|$. This function returns the i-th element from the ordered sequence of elements from P.

From Theorem 4 we know that there exists a family $\mathfrak{F}$ of sets $\{F_1, ..., F_N\}$ such that $N = \binom{n}{\frac{n}{2}}$, $\forall i \leq N \; F_i \subset Q \wedge |F_i| = \frac{n}{2}$. Let us introduce the lexicographic order $\prec$ on $\mathfrak{F}$: $F_i \prec F_{i+1} \Leftrightarrow \sum_{j=1}^{\frac{n}{2}} n^{\frac{n}{2}-j} \cdot r_{F_i}(j) < \sum_{j=1}^{\frac{n}{2}} n^{\frac{n}{2}-j} \cdot r_{F_{i+1}}(j)$, $i < N$. We see that $\prec$ is a linear order on $\mathfrak{F}$. Let us now define the transition function δ:

$$\delta(r_{F_i}(j), a_i) = r_{F_{i+1}}(j) \text{ for } i < N,\ j \leq \tfrac{n}{2}, \tag{1}$$
$$\delta(r_{F_N}(1), a_N) = 1, \tag{2}$$
$$\delta(r_{F_N}(2), a_N) = 1, \tag{3}$$
$$\delta(r_{F_N}(j), a_N) = j - 1 \text{ for } 3 \leq j \leq \tfrac{n}{2}. \tag{4}$$

Let $P = \{1, 2, ..., \frac{n}{2}\} \subset Q$. From the properties of the Sperner family $\mathfrak{F}$ we have that in order to synchronize P to some $\frac{n}{2} - 1$-element subset of Q we need to use the word $w = a_1 a_2 ... a_N$. We have $\delta(P, w) = \{1, 2, ..., \frac{n}{2} - 1\}$. In order to finish the proof we need to use the following Theorem 7, which uses Lemmata 1 and 2.

Theorem 7. *Let $P = \{1, 2, ..., k\}$, where $k \leq \frac{n}{2}$. The shortest word w' synchronizing P to a subset of cardinality less then P has length $L = \binom{\frac{n}{2}+k}{k}$.*

Proof. Fix k and let $L = \binom{\frac{n}{2}+k}{k}$. First we show that there exists such a word. Let $\mathfrak{S} = (P_1, P_2, ..., P_L)$ be a sequence of subsets of Q such that

1. $\forall 1 \leq i \leq L \; P_i \subset Q$,
2. $P_1 = P$,
3. $P_L = \{\frac{n}{2} + 1, \frac{n}{2} + 2, ..., \frac{n}{2} + k\}$,
4. $\bigcup_{i=1}^{L} P_i = \{1, 2, ..., \frac{n}{2} + k\}$,
5. $\forall 1 < i < L \; \exists a_i \in A : \; \delta(P_i, a_i) = P_{i+1}$.

$\mathfrak{S}$ represents all k-element subsets, each consisting of some k different numbers chosen from the set $\{1, 2, ..., \frac{n}{2} + k\}$. For a given sequence $\{P_i\}_{i=1}^{L}$ and letters $a_1, ..., a_{L-1}$ there exists a corresponding sequence $F = \{F_{j_i}\}_{i=1}^{L}$ of sets from the family $\mathfrak{F}$, such that $\forall 1 < i \leq L\ P_i \subseteq F_{j_i}$ and $\delta(F_{j_i}, a_i) = F_{j_{i+1}}\ \forall i < L$. We say that $\{P_i\}$ is covered by $\{F_{j_i}\}$.

Notice that in order to synchronize the set $\{1, 2, ..., k\}$ into a smaller one, we need to transform it by some word to the set R such that $\{\frac{n}{2} + 1, \frac{n}{2} + 2\} \subset R$ and for all $r \in R$ we must have $r > \frac{n}{2}$, because state 1 must be transformed into state $\frac{n}{2} + 1$. Next, it is enough to transform R by the letter a_N. States $\frac{n}{2} + 1$ and $\frac{n}{2} + 2$ will be synchronized into a single state 1 (see definition of δ, equations (2), (3)).

The property 5. of $\mathfrak{S}$ is fulfilled for each $k \leq \frac{n}{2}$. It comes directly from the definition of δ. For example, if we have $n = 20$, $k = 4$ and $P_i = \{2, 4, 13, 14\}$, then its lexicographic successor in $\mathfrak{S}$ is $P_{i+1} = \{2, 5, 6, 7\}$, and indeed there exists $a \in A$ such that $\delta(P_i, a) = P_{i+1}$: for 10-element set $P = \{2, 4, 13, 14, 15, 16, 17, 18, 19, 20\}$ we have $\delta(P, a) = \{2, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$ and particularly $\delta(2, a) = 2$, $\delta(4, a) = 5$, $\delta(13, a) = 6$ and $\delta(14, a) = 7$.

Now we will show that $w' = a_1 a_2 ... a_L$ is the shortest word synchronizing P to a smaller set. This is a direct consequence of the two following lemmata, but first we need to define the notion of a sequence width.

Definition 5. *Let $P = \{P_i\}_{i=1}^{L}$ be a sequence of t-subsets covered by $F = \{F_{j_i}\}_{i=1}^{L}$. The value $d(P, F) = \max_i \max_{k=1...t} r_{F_{j_i}}^{-1}(k)$ is called the width of P covered by F.*

Lemma 1. *Let P be the shortest sequence of k-subsets covered by F, such that $P_1 = \{1, .., k\}$, $P_L = \{\frac{n}{2} + 1, ..., \frac{n}{2} + k\}$, where L is the length of P (that is, number of subsets P_i). If $d(P, F) = k$ then $L = \binom{\frac{n}{2}+k}{k} - 1$.*

Lemma 2. *Let $P = (P_1, ..., P_L)$ be the shortest sequence of t-subsets from P_1 to P_L covered by F with width $d(P, F) = d_P$, such that $P_1 = \{1, ..., t\}$, $P_L = \{\frac{n}{2} + 1, ..., \frac{n}{2} + t\}$. For each sequence $R = (R_1, ..., R_M)$ covered by G and of width $d(R, G) = d_R$, such that $R_1 = P_1$, $R_M = P_L$ and $d(R, G) > d(P, F)$, we have $M \geq L$.*

Lemma 2 shows that the shortest sequence S (corresponding to word w') synchroninzing $P = \{1, 2, ..., k\}$ to some smaller set should be covered by F such that $d(S, F)$ is possibly the smallest, that is - $d(S, F) = |P|$. Then, from Lemma 1 we have that $(|w'| = \binom{\frac{n}{2}+|P|}{|P|} - 1) + 1 = \binom{\frac{n}{2}+|P|}{|P|}$. We have to add one letter at the end of the sequence in order to synchronize $\{\frac{n}{2} + 1, \frac{n}{2} + 2, ..., \frac{n}{2} + |P|\}$ into $\{1, 2, ..., |P| - 1\}$.

Let us now return to the proof of Theorem 5. We start from $\frac{n}{2}$-element set $T_{\frac{n}{2}} = \{1, 2, ..., \frac{n}{2}\}$ and we want to synchronize it, through the sets $T_{\frac{n}{2}-1} = \{1, 2, ..., \frac{n}{2} - 1\}$, $T_{\frac{n}{2}-2} = \{1, 2, ..., \frac{n}{2} - 2\}$, ..., $T_2 = \{1, 2\}$ to the set $T_1 = \{1\}$. From Theorem 7 we have that the shortest word w_i which synchronizes T_i into T_{i-1} $(2 \leq i \leq \frac{n}{2})$ has length $\binom{\frac{n}{2}+i}{i}$.

The word $w = w_{\frac{n}{2}} w_{\frac{n}{2}-1} ... w_2$ synchronizes our automaton $\mathcal{A}$. It's length is

$$|w| = |w_{\frac{n}{2}} w_{\frac{n}{2}-1} ... w_2| = \sum_{i=2}^{\frac{n}{2}} |w_i| = \sum_{i=2}^{\frac{n}{2}} \binom{\frac{n}{2}+i}{i} = \sum_{i=2}^{\frac{n}{2}} \binom{\frac{n}{2}+i}{\frac{n}{2}}$$

$$= \sum_{i=0}^{\frac{n}{2}} \binom{\frac{n}{2}+i}{\frac{n}{2}} - 1 - \binom{\frac{n}{2}+1}{\frac{n}{2}} = \binom{n+1}{\frac{n}{2}} - \frac{n}{2} - 2.$$

This ends the proof of Theorem 5. Let us now bound the length of m_--minimal synchronizing word for the automaton defined above. It is obvious that for an $\frac{n}{2}$-element set we can take the one, which can be transformed into $\frac{n}{2}-1$-element set in one step (by one-letter word). Such a set is $T = \{\frac{n}{2}+1, ..., \frac{n}{2}+k\}$. Because

Table 1. Comparision between lower bounds on the lengths of minimal carefully synchronizing words and m^+-minimal synchronizing words

$\|Q\|$	$\omega(\|Q\|)$	$\omega^+(\|Q\|)$	$\|Q\|$	$\omega(\|Q\|)$	$\omega^+(\|Q\|)$
8	52	120	22	8746	1 352 065
10	106	455	24	19 681	5 200 286
12	241	1 708	26	39 364	20 058 285
14	484	6 424	28	78 730	77 558 744
16	970	24 300	30	177 145	300 540 178
18	2 185	92 367	32	354 292	1 166 803 092
20	4 372	352 704	34	708 586	4 537 567 631

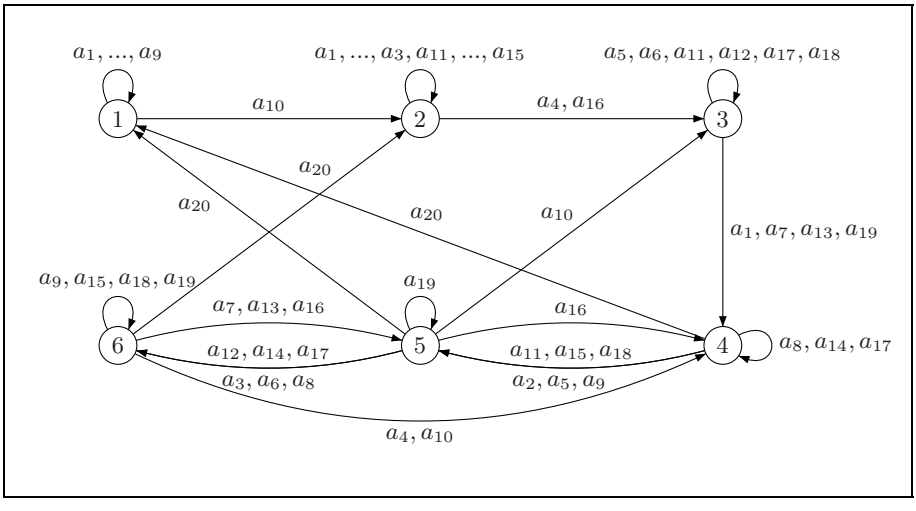


Fig. 6. 6-state automaton with m^+-minimal synchronizing word of length 30

we needed to use $\binom{n}{\frac{n}{2}} - 1$-letter word to transform $\{1, 2, ..., k\}$ into T, therefore from Theorem 6 and Proposition 5 we immediately obtain the following result:

$$\omega_-(n) \geq \omega^+(n) - (\binom{n}{\frac{n}{2}} - 1) = \binom{n}{\frac{n}{2} - 1} - \frac{n}{2} - 1 \, .$$

Table 1 presents a comparision between lower bounds on the length of minimal carefully synchronizing words and m^+-minimal synchronizing words for partially synchronizing automata.

5 Example

We illustrate now Theorem 5 by the following example. Let $n = 6$. From Proposition 4 we have $\omega^+(6) \geq \binom{7}{3} - \frac{6}{2} - 2 = 35 - 3 - 2 = 30$. We will construct the automaton with m^+-minimal synchronizing word of length 30.

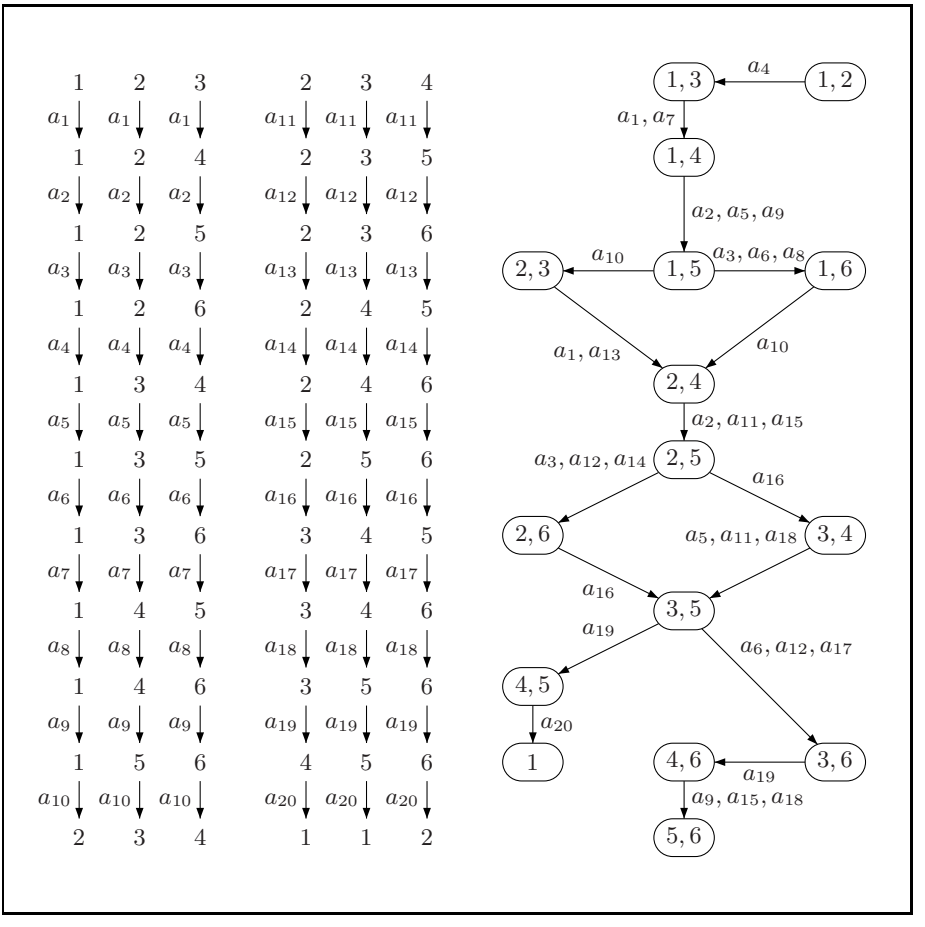


Fig. 7. Construction of δ and transformations for all 2-element subsets

Let $\mathcal{A} = (Q, A, \delta)$, where $Q = \{1, 2, 3, 4, 5, 6\}$ and $A = \{a_1 \cdots a_{20}\}$. The construction method of the transition function is presented at the left part of Fig. 7. Letter a_i transforms P_i into its very successor, P_{i+1}, according to (1). The last 3-element subset of Q, P_{20}, is synchronized to the set $\{1, 2\}$ by a_{20}, according to (2), (3), (4). In order to synchronize $\{1, 2, 3\}$ into $\{1, 2\}$ we need to use the word $w_3 = a_1 a_2 ... a_{19} a_{20}$. Now we have to synchronize $\{1, 2\}$ into $\{1\}$. The shortest such a word is the word labeling the shortest path from the state $\{1, 2\}$ to the state $\{1\}$ in the automaton illustrated on the right side of Fig. 7. Notice that one of the shortest words (denote it by w_2) transforms only the subsets of $\{1, 2, 3, 4, 5\}$: $w_2 = a_4 a_7 a_9 a_{10} a_{13} a_{15} a_{16} a_{18} a_{19} a_{20}$. Therefore an m^+-minimal synchronizing word for $\mathcal{A}$ is $w_3 w_2 = a_1 a_2 ... a_{19} a_{20} a_4 a_7 a_9 a_{10} a_{13} a_{15} a_{16} a_{18} a_{19} a_{20}$ and its length is 30. Notice, that there are also other m^+-minimal synchronizing words, for example $w_3 a_4 a_1 a_9 a_6 a_{10} a_{11} a_3 a_{16} a_{19} a_{20}$. In fact, a simple analysis of the right part of Fig. 7 shows that there are exactly 540 different m^+-minimal synchronizing words.

References

1. Ananichev, D., Volkov, M.: Synchronizing monotonic automata. In: Ésik, Z., Fülöp, Z. (eds.) DLT 2003. LNCS, vol. 2710, pp. 111–121. Springer, Heidelberg (2003)
2. Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, L., Shapiro, E.: DNA molecule provides a computing machine with both data and fuel. Proc. National Acad. Sci. USA 100, 2191–2196 (2003)
3. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machine of biomolecules. Nature 414, 430–434 1 (2001)
4. Černý, J.: Poznámka k. homogénnym experimentom s konecnymi automatmi. Mat. fyz. cas SAV 14, 208–215 (1964)
5. Černý, J., Pirická, A., Rosenauerova, B.: On directable automata. Kybernetica 7, 289–298 (1971)
6. Ito, M., Shikishima-Tsuji, K.: Some results on directable automata. In: Karhumäki, J., Maurer, H., Păun, G., Rozenberg, G. (eds.) Theory Is Forever. LNCS, vol. 3113, pp. 125–133. Springer, Heidelberg (2004)
7. Kari, J.: Synchronization and Stability of Finite Automata. JUCS 8, 2, 270–277 (2002)
8. Martyugin, P.V.: Lower bounds for length of carefully synchronizing words. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (eds.) CSR 2006. LNCS, vol. 3967, Springer, Heidelberg (2006)
9. Natarajan, B.K.: An algorithmic Approach to the Automated Design of Parts Orienters. In: Proc. 27th Annual Symp. Foundations of Computer Science, pp. 132–142. IEEE, Los Alamitos (1986)
10. Natarajan, B.K.: Some paradigms for the automated design of parts feeders. Int. J. Robotic Research 8, 6, 89–109 (1989)

A Proofs of Lemmata 1 and 2

Proof of Lemma 1 It is clear that for each $i < L$ and a such that $\delta(P_i, a)$ is defined, we have $\delta(P_i, a) = P_i$ or $\delta(P_i, a) = next(P_i)$, where $next(P)$ is the

set succeeding P_i in the lexicographic order under the alphabet $\{1, 2, ..., \frac{n}{2} + k\}$. So P must consist of all k-element subsets of $\frac{n}{2} + k$-element set, therefore $L = \binom{\frac{n}{2}+k}{k} - 1$.

Proof of Lemma 2 Denote by $max(P)$ the maximal element in P. Let us consider three possible cases:

Case 1. $max(\cup_{i=1}^{L} P_i) \geq \max(\cup_{i=1}^{M} R_i)$. In this case R_i can be covered by F and therefore be of width d_P. Because P was chosen as the shortest sequence from P_1 to P_L, we have $M \geq L$.

Case 2. $max(\cup_{i=1}^{L} P_i) < \max(\cup_{i=1}^{M} R_i) < d_R$. In this case R_i can be also covered by some F', such that $d(R_i, F') < d_R$. Therefore this case reduces to Case 1 (applying this construction at most $d_R - d_P + 1$ times).

Case 3. $max(\cup_{i=1}^{L} P_i) < max(\cup_{i=1}^{M} R_i) = d_R$. We will show that R_i can be covered by some F' such that $d(R_i, F') \leq d_R - 1$. If $d_P < d_R - 1$ we can apply this construction again until $d_P = d_R$ and then the problem reduces to the first case.

We will show how to transform R to R' of the same length, such that $R_1 = R'_1$ and $R_L = R'_L$ and $max(\cup_{i=1}^{L} R'_i) < \frac{n}{2} + d_R$. Let R_j be the first set in R such that $R_j \neq P_j$ and R_v be the first set such that $v > j$ and $R_v = P_v$. Notice that necessarily $v > j + 1$. The above implies that for each $l \in \{j, j + 1, ..., v\}$ the set G_l, which covers R_l has to be of the form $R_j \cup \{\frac{n}{2} + t + 2, \frac{n}{2} + t + 2, ..., n\}$. It is possible now to replace each R_l by $R'_l := P_l$ which will be covered by $F' = P_l \cup \{\frac{n}{2} + t + 2, \frac{n}{2} + t + 3, ..., n\}$. Such replacement should be done in all such subsequences $R_j, ..., R_v$. At the end we have the sequence R' with covering sequence F', such that $d(R', F') < d_R$. We can proceed this construction until $d(R', F') = d_P$. Then the problem reduces to Case 1. This ends the proof.

Verifying Parameterized taDOM+ Lock Managers

Antti Siirtola[1] and Michal Valenta[2]

[1] University of Oulu, Department of Information Processing Science, PL 3000, 90014 University of Oulu, Finland
antti.siirtola@oulu.fi
[2] Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Science and Engineering, Karlovo námesti 13, 121 35 Prague 2, Czech Republic
valenta@fel.cvut.cz

Abstract. taDOM* protocols are designed to provide lock-based approach to handle multiple access to XML databases. The notion of taDOM+ protocol is formalized and generalized and a formal model of taDOM+ lock manager that is parameterized in the number of transactions and in the size of database is represented. An important class of safety properties of taDOM+ lock managers were proven to be checked by examining just a small number of finite-state instances of the parameterized model. Our results were applied to prove a generalized mutual exclusion property, known as repeatable-read, of taDOM2+ and taDOM3+ lock managers by model-checking.

Keywords: Verification, model-checking, parameterized systems, case study, XML databases.

1 Introduction

Semi-structured data, like HTML and LaTeX documents, contain both content and structure information. One of the most popular and important formats for representing such data is XML (Extensible Markup Language) [27]. XML documents are not usually accessed directly, but through an interface that parses them.

DOM (Document Object Model) [27] takes an XML document basically as a tree and provides operations to manipulate them. DOM is applicable also when the documents are stored in *XML database*, database whose logical storage unit is an XML document. That is why we think the (XML) database as a *rooted tree* (V, E, r), where V is a set of *nodes*, $E \subseteq V \times V$ a set of *edges* and $r \in V$ a *root (node)* [5]. We use standard concepts of *parent*, *child*, *ancestor* and *descendant* when speaking about databases (rooted trees).

Transaction is the smallest operation a user can perform on a database. It consists of a sequence of smaller operations on nodes, such that either all the operations are completed successfully or all of them are cancelled. When the user

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 460–472, 2008.

wants to access the database, he/she first opens a transaction and then executes a sequence of operations, and finally, he/she either *commits* all operations as successfully completed, or *rolls back* the transaction, which means cancelling all the operations.

Unfortunately, DOM does not support concurrent transactional access to XML database, which is an issue of both availability and consistency. For this purpose, taDOM* protocols are introduced. They provide an efficient lock-based approach for the task [14]. The protocols are realized in the form of a transaction manager that implements DOM interface, provides transaction initialization and finalization, and invisibly takes care of locking (Fig. 1). Obviously, the role of taDOM* protocols is extremely important, but the correctness of the protocols is proved only in the form of test cases [14].

In this paper, we deal with the correctness of the lock manager part of taDOM+ protocols. taDOM+ protocols cover a half of the taDOM* protocols in practise and the lock manager is the most essential and critical part of these protocols. We have captured the behavior of the lock manager in a formal model, which is parameterized in the size of database and in the number of transactions, which means the maximum number of simultaneous accesses to the database. The correctness specification is given as an observer, which is comprised in the model. The formulation of the model allows us to study safety properties [1] related to two arbitrary transactions and one arbitrary node.

Due to the symmetry [4,12,17] of the model, to concentrate only on two fixed transactions is sufficient. We also show that an instance of the model with just two transactions enables to simulate bigger instances of the model with more transactions if the database parameter is unchanged. Then, we prove a similar bound to the size of the database. The bound depends on the correctness specification and on the taDOM+ protocol, but even in the worst case it is very small. Combining these results allows us to reduce the related parameterized verification problem to a finite verification task manageable by existing model-checkers. Finally, we apply our results to prove a generalized mutual exclusion property, known as as repeatable-read, of the lock manager of two real taDOM+ protocols, namely taDOM2+ and taDOM3+.

The parameterized verification problem is undecidable in general [2,8]. Therefore, all the methods proposed for the automatic verification of the parameterized systems are somehow restricted. Results similar to ours establishing an explicit bound for the number of replicated components have been previously proved for the systems composed of similar fixed-size processes [3,7], rings communicating through the token passing [11,10], rings of the Petri nets [21,24] and the cache coherence protocols [9,16].

In our case, the problem consists in dealing with a software system which is highly parameterized. In contrast to the standard structure of the system it is the state-space of the components which is parameterized as well. Actually, there exist only few methods that can handle the systems with parameterized components. One of the method is the method of Pyssysalo focussed on the rings of the Petri nets [24]. Obviously it is inapplicable to the systems with different topology,

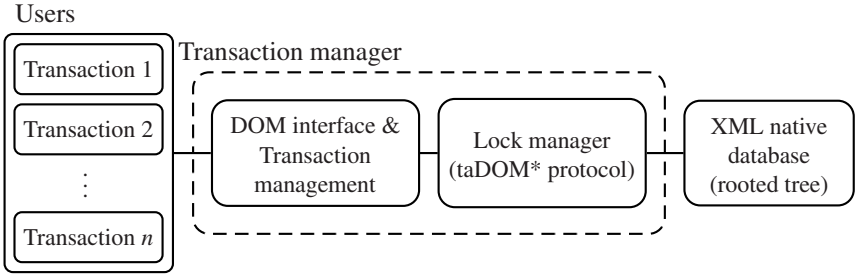


Fig. 1. Context of taDOM* protocol

like taDOM+ lock manager. Another method is an induction method of Creese [6] based on the network invariants [19,28] and data-independence [20]. Unfortunately, the method cannot be applied here, because the transactions must be able to traverse between consecutive nodes of the database. This involves incrementing and decrementing the node identifiers, which breaks data-independence of the system. We are not aware of other methods that could handle systems with parameterized components naturally.

Another problem that prevents us from using generic methods lies in the fact that the state-space of the components of our model is large. For example, a node of a taDOM3+ lock manager model has 190 states at minimum. As the complexity of generic methods is usually exponential in the number of states of a component [13,18,23] the only solution to our situation is to create the method tailor-made.

In the next section, the notation and the computation model which we build our results on are introduced. After that, we generalize and formalize the notion of a taDOM+ protocol. Section 4 presents our parameterized model of the taDOM+ lock manager, and the main result to bound the values of parameters are given in the following section. Empirical results are reported in Section 6, and finally the paper discusses the restrictions of our approach and recommendations for the future research.

2 Notation and Model of Computation

Let A and B be sets such that B has null element 0. The *Kernel* of function $f : A \mapsto B$, $\mathrm{Ker}(f)$, is set $\{a \in A \mid f(a) = 0\}$. The set of all the finite sequences of elements of A is denoted by A^*. Especially, the *empty sequence*, ϵ, is in A^*. If $w \in A^*$ and $A' \subseteq A$, then $w \setminus A'$ denotes a sequence obtained from w by removing all the elements in A'. If $i, j \in \mathbb{N}$, we write $[i, j]$ for set of natural numbers $\{n \in \mathbb{N} \mid i \leq n \leq j\}$.

We use *labelled transition system*, *LTS*, as our fundamental model of computation [26]. Intuitively, LTS is a state machine with labelled transitions. Formally, LTS $\mathcal{A}$ is a four-tuple (S, A, R, i), where (1) S is a non-empty set of *states*, (2) A is a set of *actions* such that $\tau \notin A$, (3) $R \subseteq S \times (A \cup \{\tau\}) \times S$ is a set of

transitions, and (4) $i \in S$ is *initial state*. A is the *alphabet* of $\mathcal{A}$ and it is denoted by $\Sigma(\mathcal{A})$. Actions in A are called *visible* and τ, which, by the definition, is not in A, is the *invisible action*.

When analyzing LTS's, we are usually interested in the sequences of visible actions reachable from the initial state. Let $\mathcal{A} = (S, A, R, i)$ be an LTS. An alternating sequence of states and visible and invisible actions, $s_1 a_1 s_2 \ldots a_{n-1} s_n$, of $\mathcal{A}$, is a *path (in $\mathcal{A}$) (from s_1) (to s_n)* if (s_i, a, s_{i+1}) is the transition of $\mathcal{A}$ for every $i \in \{1, 2, \ldots, n-1\}$. For a path c, $act(c)$ denotes the sequence of visible actions $c \setminus (S \cup \{\tau\})$. A state s_b is *reachable (in $\mathcal{A}$) (from s_a) (by a path c)* if c is a path in $\mathcal{A}$ from a state s_a to s_b. A state s is *reachable (in $\mathcal{A}$) (by a path c)* if it is reachable from the initial state of $\mathcal{A}$. A sequence t of visible actions of $\mathcal{A}$ is a *trace (of $\mathcal{A}$)* if there is path π in $\mathcal{A}$ such that $act(\pi) = t$. The set of all the traces of $\mathcal{A}$ is denoted by $\mathrm{tr}(\mathcal{A})$.

The systems are commonly built out of smaller LTS's by interleaving and a synchronous composition. Interleaved LTS's run independently whereas synchronously composed LTS's must take transitions labelled by the same visible action simultaneously. Finally, the actions that are irrelevant to the analysis, are hidden, i.e. they are replaced by the invisible action. Let $\mathcal{A}_1 = (S_1, A_1, R_1, i_1)$ and $\mathcal{A}_2 = (S_2, A_2, R_2, i_2)$ be LTS's and B a set of visible actions. A *parallel composition* of $\mathcal{A}_1$ and $\mathcal{A}_2$ *synchronized by* B is LTS $(S_1 \times S_2, A_1 \cup A_2, R, (i_1, i_2))$, denoted by $(\mathcal{A}_1 [| B |] \mathcal{A}_2)$, where

$$\begin{aligned} R = \{((s_1, s_2), a, (s_1', s_2')) \mid (s_1, a, s_1') \in R_1, (s_2, a, s_2') \in R_2, a \in B\} \cup \\ \{((s_1, s_2), a, (s_1', s_2)) \mid (s_1, a, s_1') \in R_1, a \notin B\} \cup \\ \{((s_1, s_2), a, (s_1, s_2')) \mid (s_2, a, s_2') \in R_2, a \notin B\} \,. \end{aligned}$$

The *Interleaving* of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted by $(\mathcal{A}_1 ||| \mathcal{A}_2)$, is a parallel composition of $\mathcal{A}_1$ and $\mathcal{A}_2$ synchronized by an empty set. *Synchronous composition* of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $(\mathcal{A}_1 \parallel \mathcal{A}_2)$, is a parallel composition of $\mathcal{A}_1$ and $\mathcal{A}_2$ synchronized by $\Sigma(\mathcal{A}_1) \cap \Sigma(\mathcal{A}_2)$. As the interleaving operator is associative, its replicated version, $|||_{i \in I} \mathcal{A}_i$, where I is a set $\{i_1, i_2, \ldots, i_k\}$ and $\mathcal{A}_i$ is an LTS for every $i \in I$, is naturally defined as LTS $(\mathcal{A}_{i_1} ||| \mathcal{A}_{i_2} \ldots ||| \mathcal{A}_{i_k})$. $\mathcal{A}_1$ *hiding* B, is LTS $(S_1, A_1 \setminus B, R', i_1)$, denoted by $(\mathcal{A}_1 \setminus B)$, where

$$R' = \{(s, a, s') \in R_1 \mid a \notin B\} \cup \{(s, \tau, s') \mid \exists a \in B.(s, a, s') \in R_1\} \,.$$

3 taDOM+ Protocols

taDOM+ protocols are described in [14] by giving a finite non-empty *set of lock modes* L, a *compatibility matrix*, $Cmp \subseteq L \times L$, and a *conversion function*, $cnv : L \times L \mapsto L$. For efficiency, at most one lock per transaction is stored on a node. A compatibility matrix tells which lock modes of different transactions can coexist on a node. We say that a lock mode l_1 *is compatible with* l_2 if $(l_1, l_2) \in Cmp$. If l_1 is compatible with l_2, then one transaction can have lock l_1 on a node while another transaction has lock l_2 on the same node. If a transaction requires lock mode l' on a node on which it already has lock l, then, as the result

of the request, the transaction will have lock $l'' := cnv(l', l)$ on the node, provided l'' is compatible with the locks other transactions have on the node.

However, these constructs are not sufficient enough to describe a taDOM+ protocol in detail; there are other constructs implicitly related to each of the protocols. A successful lock request on a node always leads to a lock request on the parent. The lock that will be requested on the parent node is determined by a *precedence function*, $prv : L \mapsto L$. Thus, if a transaction requests a lock on a node and, as the result, the transaction gets lock l on the node, then lock mode $prv(l)$ will be requested on the parent. It means that node n cannot be locked without locking all the nodes in the path from n to the root node. However, the first lock request in such a locking chain cannot be arbitrary. A set of *directly request-able lock modes* $D \subseteq L$ captures the lock modes which are allowed be requested on a node without the previous request on the one of its children.

Each lock mode is associated with a set of operations (on nodes) with scopes. The *set of operations (on nodes)* is denoted by O and typically consists of read and write operations. A *lock operation mapping* is a function, $lop : L \mapsto \mathbb{P}(O \times \mathbb{N})$, such that $lop(l)$ is a *downward-closed set* whenever $l \in L$, meaning that if $(o, n) \in lop(l)$ and $m < n$ then $(o, m) \in lop(l)$. A transaction can perform an operation o on a node n, if the transaction has, or successfully requests, a lock mode l on an ancestor n' of n such that $(o, d) \in lop(l)$, where d is the distance between n and n' in edges. Thus, lock l on node n allows to perform operation $o \in lop(l)$ on all the descendants of n within distance $\sup\{k \mid (o, k) \in lop(l)\}$ from n, including n itself.

As an example, we consider an XML library database which consists of books where it holds that for each of them specific data are assigned, e.g. author, type, availability etc. The database can be simultaneously accessed by many users, so its important to isolate write operations from simultaneous read or write operations on the same data.

For that purpose, we introduce a simple taDOM+ protocol with two self-explaining operations on the nodes, read and write, and five lock modes whose first two modes are directly request-able: SR — subtree read, SX — subtree read and write, IR — intention to read in a subtree, IX — intention to read or write in a subtree — and SRIX — subtree read with intention to write in a subtree.

Lock operation mapping corresponding to the intuitive explanation of the lock modes is a function mapping IR, IX and SRIX to the empty set, SR to $\{\text{read}\} \times \mathbb{N}$ and SX to $\{\text{read}, \text{write}\} \times \mathbb{N}$. The compatibility relation and the conversion function are given respectively, see Tables 1a and 1b. The first parameter of the conversion function denotes the row, the second one the column, and the result of the conversion can be read in the cell in the intersection. Similarly, lock modes are compatible if there is + in the corresponding cell of the first table. The precedence function maps SR and IR to IR and the rest of the lock modes to IX.

Now, let us assume that initially no one uses the database. Then appears a user who wants to see the particulars of a book. The system executes Transaction 1 for him/her and the read lock mode SR is requested on the node representing the book. Because there are no other locks on the node, it is successfully locked

Table 1. Compatibility and conversion matrices of taDOM+ protocol for library database

	IR	SR	IX	SRIX	SX
IR	+	+	+	+	
SR	+	+			
IX	+		+		
SRIX	+				
SX					

(a)

	IR	SR	IX	SRIX	SX
IR	IR	SR	IX	SRIX	SX
SR	SR	SR	SRIX	SRIX	SX
IX	IX	SRIX	IX	SRIX	SX
SRIX	SRIX	SRIX	SRIX	SRIX	SX
SX	SX	SX	SX	SX	SX

(b)

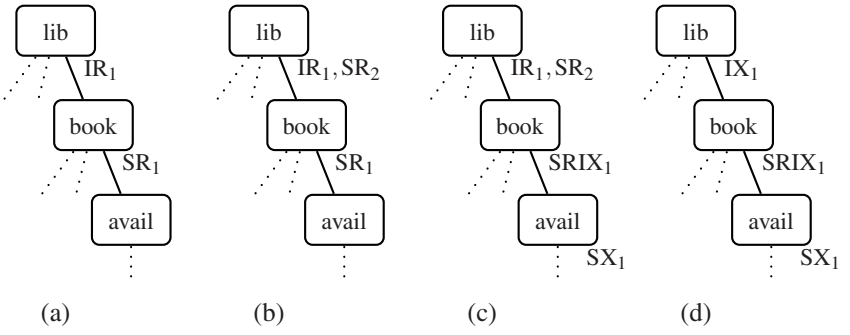


Fig. 2. (a) Transaction 1 accessing one book. (b) Transaction 1 accessing one book and Transaction 2 accessing all the books. (c) Transaction 2 accessing all the books and Transaction 1 trying to update availability information of a book. (d) Transaction 1 updating availability information of a book.

by SR and, according to the precedence function, IR is requested and granted on its parent (Fig. 2a).

Shortly after that appears another user who wants to list all the books in the database. The system executes Transaction 2 for him/her and the root gets locked by SR, because the transaction has no earlier lock on the node and SR is compatible with IR lock mode of Transaction 1 (Fig. 2b).

Next, the first user wants to borrow the book, and thus to change its information on availability. The write lock mode SX is requested on the availability node of the book by Transaction 1. However, before any changes are made, the whole path up to the root must be locked appropriately. It means request of IX and conversion of SR to SRIX on the book node and request of IX and conversion of IR to IX on the root node according to precedence function and conversion matrix. However, the conversion on the root node cannot be applied because IX is not compatible with the read lock of Transaction 2 which already exists there (see compatibility matrix). That is why Transaction 1 is blocked (Fig. 2c) until Transaction 2 finishes processing.

As Transaction 2 has generated the list of all the books for the second user, it ends and releases the locks it possesses. Now Transaction 1 can finish its lock request and the first user can borrow the book (Fig. 2d).

4 Modelling taDOM+ Lock Manager

Straightforward modelling of a taDOM+ lock manager leads to an infinite-state LTS, which cannot be model-checked in general. However, in practise, applications cannot reach infinitely many states due to limited memory and running time, and other limited resources. That is why we model the lock manager parameterized by the restrictions. The only problem with this approach is that parameters cannot usually be bound. That is why we instantiate the model for every possible value in the parameter domain, which results in an (infinite) family of similar finite-state systems. Although verifying the whole family automatically with limited effort is impossible in general [2], we will see that this is possible for the parameterized model of the taDOM+ lock manager we have created.

The number of transactions is a natural parameter, but parameterizing the size and the shape of the database is not that straightforward. We concentrate on an arbitrary node of the database, which we call *the context node*. We model only the path which leads from the root node to the context node, and we introduce abstract nodes representing a sequence of one or more nodes (Fig. 3). The nodes outside the path are not explicitly modelled. We model only the effect on the nodes on the path when a lock is requested on one of the other nodes. Abstract nodes do not exist as LTS's, but their identifiers are included in the model. When the transaction is proceeding on the path on either direction and enters an abstract node, it can stay there arbitrarily long, silently simulating the lock requests and lock reads.

We denote the set of transaction identifiers by $T_b = \{\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_b\}$ and the set of node identifiers by $N_c = \{\mathrm{n}_1, \mathrm{n}_2, \ldots, \mathrm{n}_c\}$. If $t \in T_b$ is a transaction identifier and $n \in N_c$ a node identifier, then the corresponding transaction and the node are referred to by respectively $\hat{t}$ and $\hat{n}$. *Database parameter* f is a mapping from N_c to $\{0, 1\}$ such that a sequence $f(\mathrm{n}_1)f(\mathrm{n}_2)\ldots f(\mathrm{n}_c)$ contains no consecutive 1-symbols. Now, if f maps a node identifier n to 1, then $\hat{n}$ is (modelled as) an abstract node, otherwise $\hat{n}$ is (modelled as) a *regular* node. The class of all the database parameters from N_c with the kernel of size c' is denoted by $\mathrm{F}_{c,c'}$. The LTS representing transaction $\hat{t}$ depends on database parameter f and the LTS of regular node $\hat{n}$ on the number of transactions b. The LTS's are denoted respectively by $Tr_f(t)$ and $Nd_b(n)$. Here, we present an intuitive view on the models, the formal definition is given in [29].

For modelling purposes, we assume that there is a special *empty lock mode* $\perp \in L \setminus D$, and initially every transaction has this lock on every node. We require that (1) $\{\perp\} \times L \cup L \times \{\perp\} \subseteq Cmp$, (2) $cnv(\perp, l) = \perp$ and $cnv(l, \perp) = l$ for every $l \in L$, (3) $prv(l) = \perp$ if and only if $l = \perp$ for all $l \in L$, and (4) $lop(\perp) = \emptyset$. Then $\perp$ represents the absence of the lock in a node and, on the other hand, it indicates a lock release request.

$Nd_b(n)$ just stores the lock for each transaction LTS, no data or other properties of the node are modelled. The locks are initialized to the empty lock mode. At any moment, the mode of the lock of any transaction can be read. The lock can also be updated anytime as long as the resulting lock mode is compatible

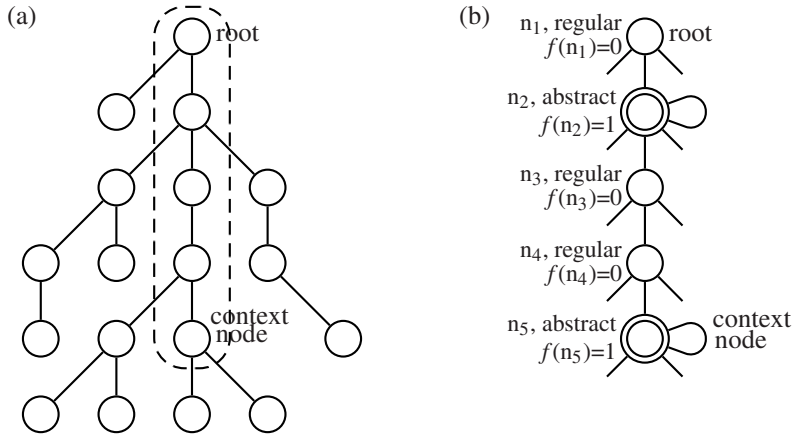


Fig. 3. Creation of a database instance: (a) The context node is chosen. The nodes in the path from the root to the context node are explicitly modelled. (b) Possible connections to the nodes outside the path are modelled as well, and the abstract nodes are introduced. The self-loops on the abstract nodes clarify the fact their representation of one or more consecutive real nodes.

with the other locks on the node. Locking a node is an atomic operation, just like in real implementations [15].

The transactions are modelled from the point of view of the lock manager, and the context node. $Tr_f(t)$ represents what the lock manager does for the transaction $\hat{t}$ and what operations the transaction can perform on the context node. $Tr_f(t)$ stores the set of operations $O' \subseteq O$ the transaction is entitled to perform on the context node. Naturally, O' is initially empty. We assume that once a transaction has obtained the rights to an operation, it cannot lose them. This modelling choice will not result in false negative verification results unless the conversion function allows the locks to be degraded. Hence, this will not be a problem with the models of real the taDOM+ protocols but they can be analyzed only in *repeatable-read* mode, which presumes that the locks of a transaction are not released until the transaction ends. However, in the case of taDOM+ protocols, repeatable-read is the most important operating mode.

In the model, the creation of transaction $\hat{t}$ is indicated by action t.beg. After that, $Tr_f(t)$ enters the main loop. Now, three kinds of things can be repeatedly done until the transaction ends.

(1) $Tr_f(t)$ can perform any operation $o \in O'$ on the context node. The operation is modelled using two actions, $t.o$.beg and $t.o$.end, representing the moments when $\hat{t}$ respectively starts and finishes execution of the operation.

(2) $Tr_f(t)$ can search for locks on the path that entitle to perform a set of operations U on the context node such that U is disjoint from O'. Searching always starts from the root node and proceeds towards the context node until U becomes empty or the context node is passed. During the search, the locks l the transaction has on the regular nodes $Nd_b(\mathrm{n}_i)$ are being read and U is cut down

and O extended by $\{o|(o, c-i) \in lop(l)\} \cap U$. Visiting abstract nodes causes only U to be reduced non-deterministically but O' is not affected, because otherwise all the transactions could get rights to every operation simultaneously, which would make the model unusable for the verification of any reasonable property. Correctness of this modelling choice will be established in the next section.

(3) $Tr_f(t)$ can (directly) request any directly request-able lock mode on any regular node on the path. It is also possible that $\hat{t}$ makes the first (direct) lock request on the node outside the path or on the node modelled as an abstract one. That is why $Tr_f(t)$ can (indirectly) request l' on any node on the path, where l' is any lock mode that can be obtained from a directly request-able lock mode by repeated (zero or more times) application of functions $g_{l''}(l) := prv(cnv(l, l'')), l'' \in L$.

If a direct request on regular node $Nd_b(\mathrm{n}_i)$ is successful, which means the resulting lock l' is compatible with the other locks on $Nd_b(\mathrm{n}_i)$, O' is extended by $\{o|(o, c-i) \in lop(l)\}$ and lock mode $l'' := prv(l')$ is (indirectly) requested on the parent node. Lock requests on abstract nodes are always successful, because abstract nodes do not store locks. That is why requesting a lock mode l on an abstract node just means that the lock mode that will be requested on the parent is non-deterministically selected from $\{prv(l, l'') \mid l'' \in L\}$. For the reasons explained earlier, O' is not affected. Additionally, because an abstract node represents any non-empty sequence of real nodes also the parent is non-deterministically chosen from two alternatives, which are the node itself and its actual parent. Hence, it is possible to stay within one abstract node arbitrarily long. Locking proceeds like this until the root node is passed.

Finally, when in the main loop, the transaction can always end. This is indicated by action t.end. After that all the locks of $Tr_f(t)$ are released, replaced by $\perp$, starting from the context node.

To build an instance $AS_{b,f}$ of the lock manager model with b transactions and database f, the transaction LTS's are mutually interleaved. It means that our lock manager model can serve the transactions simultaneously. In real implementations, not every interleaving is possible, which makes our model more general. Also the regular nodes are interleaved, because they do not communicate directly with each other. As the transactions and nodes communicate through the lock requests and lock reads, the resulting two LTS's are synchronously composed to create the lock manager model. Thus,

$$AS_{b,f} = \left(\;|||_{t \in T_b} \; Tr_f(t) \right) \,||\, \left(\;|||_{n \in \mathrm{Ker}(f)} \; Nd_b(n) \right) .$$

We are interested in safety properties [1] related to the operations of two arbitrary transactions, t_1 and t_2 performed on the context node. We feel that this kind of properties cover a large portion of the safety properties of practical interest. As taDOM+ protocols can deadlock, there is no point to reason liveness properties [1] and, in real implementations, there are special mechanisms for recovering from the deadlocks.

Normally, the correctness specification is formulated directly as an LTS, too. However, we assume that the specification is given indirectly as an *observer*. An

observer is such LTS that captures the correct behavior and provided it sees the behavior which is not in accordance with the property, executes a self-explaining action error [22].

We write $Obs(t_1, t_2)$ for an observer capturing a safety property related to operations transactions t_1 an t_2 perform on the context node. Thus, $Obs(t_1, t_2)$ is an LTS with alphabet $\{t.\text{beg}, t.\text{end}, t.o.\text{beg}, t.o.\text{end}, \text{error} \mid t \in \{t_1, t_2\}, o \in O\}$. If π is a path in $Obs(t_1, t_2)$, then $\Omega(\pi) = \{(t, o) \mid t.o.\text{beg or } t.o.\text{end occurs in } \pi\}$ denotes the set of transaction-operation pairs occurring in the actions of π. We write $|\Omega(Obs(t_1, t_2))|$ for the maximum of $|\Omega(\pi)|$ taken over paths π in $Obs(t_1, t_2)$ such that π does not contain error actions.

As we can now concentrate only on error actions, all the other actions can be hidden. Thus, the problem we are addressing is whether equation

$$\text{tr}((AS_{b,f} \mid\mid Obs(t_1, t_2)) \setminus \Sigma(AS_{b,f})) = \{\epsilon\} \tag{1}$$

is true for all positive integers c, c' and $b \geq 2$, any $f \in \text{F}_{c,c'}$ and every distinct $t_1, t_2 \in T_b$. If the equation is true for all the instances of the lock manager and every pair of transaction identifiers, then the observer cannot perform the error action and thus the lock manager is in compliance with the specification.

However, because of the abstractions, the model-checks within our framework may give false negative answers. It means that the actual lock manager can be correct even if the verification of the model gives the opposite result. Fortunately, this probably happens only if our framework is utilized in more innovative ways. On the other hand, if our approach proves the protocol correct then it must be correct.

5 Bounding the Parameters

We can automatically check Equation 1 for small b and f and any $t_1, t_2 \in T_b$. However, if we want to prove the correctness of the entire lock manager, the results of model-checking have to be generalized in three ways. We need to prove that Equation 1 holds independent of the number of transactions b, the structure of database f and the choice of t_1 and t_2.

First, let us fix b and f. To establish the correctness of $AS_{b,f}$ we should still check Equation 1 for each pair of transaction identifiers. Fortunately, it is not difficult to generalize observations made by two fixed transaction LTS's to any pair of transaction LTS's. By the construction of the model, it is easy to see that the resulting system is symmetric under the permutations of transaction identifiers. The symmetry on the model checking [4,12,17] is a well-known topic and we do not claim any originality here. As a consequence, we may assume that $t_1 = \text{t}_1$ and $t_2 = \text{t}_2$ in Equation 1 from now on.

It is also quite easy to show that adding transaction LTS's and hiding their actions cannot introduce new traces, that means new ways to violate the property. Therefore it is sufficient to study instances of the lock manager model having just two transaction LTS's. Thus, b in Equation 1 can be set to 2.

Once transaction LTS obtains the rights to perform an operation on the context node, the rights exist as long as it ends, because its locks are not released earlier. Getting the rights necessitates a suitable lock somewhere in the path, which means at most $|O|$ regular nodes per transaction LTS are effectively needed. Therefore, to check all the instances of the model having at most $2|O|$ regular nodes is sufficient. It means that we can also bound the size of the database.

However, in some cases we can do better than that. If in every trace t of an observer, such that no error action occurs in t, at most n different transaction-operation pairs occur in the actions of t, then it is sufficient to check all the instances of the lock manager model having at most n regular nodes. This is because every trace containing error actions has a maximal prefix that does not contain them, which means that using a model of n regular nodes it is possible to run the observer to a state from which the error action can be executed, if it can be executed at all.

To summarize, it is enough to check all the instances of the lock manager model having two transaction LTS's, at most $2|\Omega(Obs(t_1, t_2))|+1$ abstract nodes and at most $|\Omega(Obs(t_1, t_2))|$ regular nodes. The result is captured in the following theorem the proof of which can be found at [29].

Theorem 1. *Equation 1 holds for all positive integers c, c' and $b \geq 2$, every $f \in \mathrm{F}_{c,c'}$ and all distinct $t_1, t_2 \in T_b$ if and only if equation*

$$\mathrm{tr}((AS_{2,f} \mid\mid Obs(\mathrm{t}_1, \mathrm{t}_2)) \setminus \Sigma(AS_{2,f})) = \{\epsilon\}$$

is true for all positive integers c and $c' \leq |\Omega(Obs(\mathrm{t}_1, \mathrm{t}_2))|$ and every $f \in \mathrm{F}_{c,c'}$.

6 Verification Results

We have applied our result to prove a generalized mutual exclusion property, known as repeatable-read [25] in database world, for the lock managers at taDOM2+ and taDOM3+ protocols. Repeatable-read property states that reading a node should always give the same result within a transaction unless the transaction itself changed the contents of the node. In other words, if a transaction is writing to or has written to a node, no other transaction should be able to access the node until the transaction has ended, and if a transaction is reading or has read a node, no other transaction should be able to write to the node until the transaction has ended. The property was formulated as an observer with $|\Omega(Obs(t_1, t_2))| = 3$. Therefore it was sufficient to check all the instances of the protocol model having at most 3 regular nodes.

Model-checking was done using CSP model-checker FDR2 [26]. To verify a protocol, 18 model-checking runs were needed, one for each database parameter $f \in \{g \in \mathrm{F}_{m',m} \mid 1 \leq m' \leq m \leq 7, m' \leq 3\}$. The largest instance of the taDOM2+ lock manager model that needed to be checked had 25 million states and 330 million transitions. It took 30 minutes and 720 MB of memory to verify it using Sun V40z computation server with 2 GHz dual core Opteron 64 processors running on Solaris 10.

In order to verify taDOM3+ we had to make an assumption that it is always $Tr_f(\mathrm{t}_1)$ that first accesses the context node. It was achieved by composing the model with a process capturing the behavior. Without the restriction, the model was too large for FDR2 to handle. Since both the model and the property are symmetric under the permutations of T_2, the restriction does not affect the universality of model-checking results. Under this assumption, the largest model-check needed for proving the correctness of a taDOM3+ lock manager model took about 3.3 GB of memory and 180 minutes to complete. There were over 120 million states and 1.4 billion transitions to check.

7 Discussion

There are two obvious possible ways how to extend the results. Even though the lock manager model is proved correct independent of the number of transactions and the size of database, we have assumed the structure of the database to be static. Hence, modelling node inserts and removals and still keeping the model small enough for automatic verification is the first challenging task. The second task is to generalize the results in such a way to support systems with non-predefined structure and components.

Acknowledgments. The research is partially funded by the Ministry of Education of Finland through Infotech Oulu Graduate School, the Grant GA201/06/0756, and research program MSM 6840770014 of Ministry of Education, Czech Republic. We thank Juha Kortelainen for his comments on the paper and Pavel Strnad for discussions on taDOM* protocols.

References

1. Alpern, B., Schneider, F.B.: Defining Liveness. Inf. Process. Lett. 21, 181–185 (1985)
2. Apt, K.R., Kozen, D.C.: Limits for automatic verification of finite-state concurrent systems. Inform. Process. Lett. 22, 307–309 (1986)
3. Attie, P.C., Emerson, E.A.: Synthesis of concurrent systems with many similar processes. ACM T. Progr. Lang. Sys., 51–115 (1998)
4. Clarke, E.M., Enders, R., Filkorn, T., Jha, S.: Exploiting symmetry in temporal logic model checking. Form. Method. Syst. Des. 9, 77–104 (1996)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2001)
6. Creese, S.J.: Data Independent Induction: CSP Model Checking of Arbitrary Sized Networks. Ph.D. thesis, Oxford University (2001)
7. Emerson, E.A., Kahlon, V.: Reducing model checking of the many to the few. In: McAllester, D. (ed.) Automated Deduction - CADE-17. LNCS, vol. 1831, pp. 236–254. Springer, Heidelberg (2000)
8. Emerson, E. A., Kahlon, V.: Model checking guarded protocols. In: Proc. LICS 2003, Ottawa, pp. 361–370 (2003)

9. Emerson, E.A., Kahlon, V.: Exact and efficient verification of parameterized cache coherence protocols. In: Geist, D., Tronci, E. (eds.) CHARME 2003. LNCS, vol. 2860, pp. 247–262. Springer, Heidelberg (2003)
10. Emerson, E.A., Kahlon, V.: Parameterized model checking of ring-based message passing systems. In: Marcinkowski, J., Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 325–339. Springer, Heidelberg (2004)
11. Emerson, E.A., Namjoshi, K.S.: Reasoning about rings. In: Proc. POPL 1995, San Francisco, pp. 85–94 (1995)
12. Emerson, E.A., Sistla, A.P.: Symmetry and model checking. Form. Method. Syst. Des. 9, 105–131 (1996)
13. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM 39, 675–735 (1992)
14. Haustein, M., Härder, T.: Optimizing concurrent XML processing. Internal report, Kaiserslautern University of Technology (2005), http://wwwlgis.informatik.uni-kl.de/archiv/wwwdvs.informatik.uni-kl.de/pubs/papers/HH05.Int-Report.pdf
15. Haustein, M., Härder, T.: An efficient infrastructure for native transactional XML processing. Data Knowl. Eng. 61, 500–523 (2007)
16. Henzinger, T.A., Qadeer, S., Rajamani, S.K.: Verifying sequential consistency on shared-memory multiprocessor systems. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 301–315. Springer, Heidelberg (1999)
17. Ip, N., Dill, D.: Better verification through symmetry. Form. Method. Syst. Des. 9, 41–75 (1996)
18. Ip, N., Dill, D.: Verifying systems with replicated components in Murφ. Form. Method. Syst. Des. 14, 273–310 (1999)
19. Kurshan, R.P., McMillan, K.: A structural induction theorem for processes. Inf. Comp. 117, 1–11 (1995)
20. Lazić, R.S.: A Semantic Study of Data Independence with Applications to Model Checking. Ph.D. thesis. Oxford University (2001)
21. Li, J., Suzuki, I., Yamashita, M.: A new structural induction theorem for rings of temporal Petri nets. IEEE T. Software Eng. 20, 115–126 (1994)
22. Lesens, D., Halbwachs, N., Raymond, P.: Automatic verification of parameterized networks of processes. Theor. Comput. Sci. 256, 113–144 (2001)
23. Lubachevsky, B.D.: An approach to automating the verifcation of compact parallel coordination programs I. Acta Inform. 21, 125–169 (1984)
24. Pyssysalo, T.: An Induction Theorem for Ring Protocols of Processes Described with Predicate/Transition Nets. Research Reports, Helsinki University of Technology A37 (1996)
25. Ramakrishnan, R., Gehrke, J.: Database Management Systems, 3rd edn. McGraw Hill, New York (2002)
26. Roscoe, A.W.: The Theory and Practice of Concurrency. Prentice Hall, Englewood Cliffs (1997)
27. World Wide Web Consortium, http://www.w3c.org/
28. Wolper, P., Lovinfosse, V.: Verifying properties of large sets of processes with network invariants. In: Sifakis, J. (ed.) Automatic Verification Methods for Finite State Systems. LNCS, vol. 407, pp. 68–80. Springer, Heidelberg (1990)
29. Appendix, http://www.tol.oulu.fi/~santti/papers/tadom_appendix.pdf

Untangling a Planar Graph

Andreas Spillner[1] and Alexander Wolff[2]

[1] School of Computing Sciences, University of East Anglia, Norwich, UK
aspillner@cmp.uea.ac.uk
[2] Faculteit Wiskunde en Informatica, TU Eindhoven, The Netherlands
http://www.win.tue.nl/~awolff

Abstract. In John Tantalo's on-line game *Planarity* the player is given a non-plane straight-line drawing of a planar graph. The aim is to make the drawing plane as quickly as possible by moving vertices. Pach and Tardos have posed a related problem: can any straight-line drawing of any planar graph with n vertices be made plane by vertex moves while keeping $\Omega(n^\varepsilon)$ vertices fixed for some absolute constant $\varepsilon > 0$? It is known that three vertices can always be kept (if $n \geq 5$).

We still do not solve the problem of Pach and Tardos, but we report some progress. We prove that the number of vertices that can be kept actually grows with the size of the graph. More specifically, we give a lower bound of $\Omega(\sqrt{\log n / \log\log n})$ on this number. By the same technique we show that in the case of outerplanar graphs we can keep a lot more, namely $\Omega(\sqrt{n})$ vertices. We also construct a family of outerplanar graphs for which this bound is asymptotically tight.

1 Introduction

At the 5th Czech-Slovak Symposium on Combinatorics in Prague in 1998, Mamoru Watanabe asked the following question. Is it true that every polygon P with n vertices can be untangled, i.e., turned into a non-crossing polygon, by moving at most εn of its vertices for some absolute constant $\varepsilon < 1$? Pach and Tardos [8] have answered this question in the negative by showing that there must be polygons where at most $O((n \log n)^{2/3})$ of the vertices can be kept fixed. In their paper, Pach and Tardos in turn asked the following question: can any straight-line drawing of any planar graph with n vertices be made plane by vertex moves while keeping $\Omega(n^\varepsilon)$ vertices fixed for some absolute constant $\varepsilon > 0$? It is known [14,4] that at least three vertices can always be kept (assuming $n \geq 5$). We still do not know the answer to the question of Pach and Tardos, but we report further progress. We show that $\Omega(\sqrt{\log n / \log\log n})$ vertices can always be kept. For outerplanar graphs our method keeps a lot more, namely $\Omega(\sqrt{n})$ vertices, and we show that there are drawings of outerplanar graphs where only $O(\sqrt{n})$ vertices can be kept fixed, i.e., our bound is asymptotically tight.

There is a popular on-line game that is related to the problem of Pach and Tardos. In John Tantalo's game *Planarity* [12] the player is given a non-plane

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 473–484, 2008.

straight-line drawing of a planar graph. The player can move vertices, which always keep straight-line connections to their neighbors. The aim is to make the drawing plane as quickly as possible.

Let's formalize our problem. Given a planar graph $G = (V, E)$ a straight-line drawing of G in the plane is uniquely defined by an injective map $\delta : V \to \mathbb{R}^2$ of the vertices of G into the plane. It will be convenient to identify the map δ with the straight-line drawing of G that is defined by δ. A drawing of G is plane if no two edges in the drawing cross each other, that is, they only share points which are endpoints of both edges. Given a drawing δ of G let

$$\mathrm{fix}(G, \delta) = \max_{\delta' \text{ plane drawing of } G} |\{v \in V \mid \delta(v) = \delta'(v)\}|,$$

denote the maximum number of vertices of G that can be kept fixed when making δ plane. Let $\mathrm{fix}(G) = \min_{\delta \text{ drawing of } G} \mathrm{fix}(G, \delta)$ denote the maximum number of vertices of G that can be kept fixed when starting with the worst-possible drawing of G. In this paper we show $\mathrm{fix}(G) \in \Omega(\sqrt{\log n / \log\log n})$, where n is the number of vertices of G.

Our approach is as follows. Our main theorem (see Section 4) guarantees that $\mathrm{fix}(G) \in \Omega(\sqrt{l})$ for all triangulated planar graphs G that contain a simple length-l path of a special structure. In terms of the diameter d and the maximum degree Δ of G our main theorem yields bounds of $\Omega(\sqrt{d})$ and $\Omega(\sqrt{\Delta})$, respectively, for $\mathrm{fix}(G)$. The former is achieved with the help of so-called *Schnyder woods*. Moore's bound—a trade-off between d and Δ—then yields the bound $\Omega(\sqrt{\log n / \log\log n})$ for $\mathrm{fix}(G)$ in terms of n, see Section 5. The bound $\Omega(\sqrt{\Delta})$ immediately yields a lower bound of $\Omega(\sqrt{n})$ for outerplanar graphs. We complement this result by an asymptotically tight upper bound in Section 6. We start by reviewing previous work in Section 2 and outlining our method in Section 3.

2 Previous and Related Work

Pach and Tardos [8] have shown that $\sqrt{n} < \mathrm{fix}(C_n) \leq c(n \log n)^{2/3}$ where C_n is the cycle with n vertices and c is some positive constant. They used a probabilistic method based on the crossing lemma.

Verbitsky [14] has considered two graph parameters; the *obfuscation complexity* $\mathrm{obf}(G)$ of a graph G, which is the maximum number of edge crossings in any drawing of $G = (V, E)$, and the *shift complexity* $\mathrm{shift}(G) = |V| - \mathrm{fix}(G)$ of G. Concerning the shift complexity he observed that $\mathrm{fix}(G) \geq 3$ for planar graphs with $n \geq 5$ vertices. Further he gave two linear lower bounds on $\mathrm{shift}(G)$ depending on the connectivity of G. By reduction from independent set in line-segment intersection graphs he showed that computing the shift complexity $\mathrm{shift}(G, \delta)$ of a fixed drawing is NP-hard even if the given graph is restricted to a matching. This explains why Tantalo's game Planarity is difficult and shows that computing $\mathrm{fix}(G, \delta)$ is hard, too.

Independently, Goaoc et al. [4] have also shown that $\mathrm{fix}(G) \geq 3$ for any planar graph G with $n \geq 5$ vertices and that computing the shift complexity is NP-hard. Their (more complicated) reduction is from planar 3-SAT. A

variant of their reduction also shows that $\text{shift}(G, \delta)$ is hard to approximate. More precisely, if $\mathcal{P} \neq \mathcal{NP}$ then for no $\varepsilon \in (0, 1]$ there is a polynomial-time $(n^{1-\varepsilon})$-approximation algorithm for $\text{shift}(G, \delta) + 1$. Note that this does *not* imply hardness of approximation for computing $\text{fix}(G, \delta)$. On the combinatorial side, Goaoc et al. showed that $\text{fix}(T) \geq \sqrt{n/3}$ for any tree T with n vertices and that there exist planar graphs G with an arbitrary large number n of vertices such that $\text{fix}(G) \leq \lceil \sqrt{n-2} \rceil + 1$. Note that the graphs in their construction are not outerplanar and, therefore, this does not imply our result presented in Section 6.

Kang et al. [7] have investigated an interesting related problem. They start with a plane drawing of a graph and want to make it straight-line, again by vertex moves. For any positive integers s and k they construct a graph $G_{s,k}$ with $n = k(s + k)$ vertices and a plane drawing $\delta_{s,k}$ of $G_{s,k}$ such that $M \geq s(k - 1)$ moves are needed to make $\delta_{s,k}$ straight-line. The bound on M is maximized for $k \in O(n^{1/3})$ and thus shows that $\text{fix}(G, \delta_{s,k}) \in O(n^{2/3})$, which is weaker than the upper bound of $2\sqrt{n}$ proved by Goaoc et al. Note, however, that the drawings of Goaoc et al. are not plane.

Very recently—after submission of this paper and its long version [11]—Bose et al. [2] answered the question of Pach and Tardos [8] in the affirmative by showing that for any planar graph with n vertices at least $\sqrt[4]{n/9}$ vertices can be kept, thus improving our bound. They also showed that the $\Omega(\sqrt{n})$ lower bound of Goaoc et al. for trees is asymptotically tight.

3 Preliminaries and Overview

Definitions and notation. A(n abstract) *plane embedding* of a planar graph is given by the circular order of the edges around each vertex and by the choice of the outer face. A plane embedding of a planar graph can be computed in linear time [6]. If G is triangulated, a plane embedding of G is determined by the choice of the outer face. Recall that an edge of a graph is called *chord* with respect to a path Π if the edge does not lie on Π but both its endpoints are vertices of Π.

For a point $p \in \mathbb{R}^2$ let $x(p)$ and $y(p)$ be the x- and y-coordinates of p, respectively. We say that p lies *vertically below* $q \in \mathbb{R}^2$ if $x(p) = x(q)$ and $y(p) \leq y(q)$. For a polygonal path $\Pi = v_1, \ldots, v_k$, we denote by $V_\Pi = \{v_1, \ldots, v_k\}$ the set of vertices of Π and by $E_\Pi = \{v_1v_2, \ldots, v_{k-1}v_k\}$ the set of edges of Π. We call a polygonal path $\Pi = v_1, \ldots, v_k$ *x-monotone* if $x(v_1) < \cdots < x(v_k)$. In addition, we say that a point $p \in \mathbb{R}^2$ lies *below* an x-monotone path Π if p lies vertically below a point p' (not necessarily a vertex!) on Π. Analogously, a line segment $\overline{pq}$ lies below Π if every point $r \in \overline{pq}$ lies below Π. We do not always strictly distinguish between a vertex v of G and the point $\delta(v)$ to which this vertex is mapped in a particular drawing δ of G. Similarly, we write vw both for the edge $\{v, w\}$ of G and the straight-line segment connecting $\delta(v)$ with $\delta(w)$.

The basic idea. Note that in order to establish a lower bound on $\text{fix}(G)$ we can assume that the given graph G is triangulated. Otherwise we can triangulate G arbitrarily (by fixing an embedding of G and adding edges until all faces are

3-cycles) and work with the resulting triangulated planar graph. A plane drawing of the latter trivially yields a plane drawing of G. So let G be a triangulated planar graph, and let δ_0 be a drawing of G, e.g., one with $\mathrm{fix}(G, \delta_0) = \mathrm{fix}(G)$.

The basic idea of our algorithm is to find a plane embedding β of G such that there exists a long simple path Π connecting two vertices s and t of the outer triangle stu with the property that all chords of Π lie on one side of Π (with respect to β) and u lies on the other. For an example of such an embedding β, see Fig. 1(b). We describe how to find β and Π depending on the maximum degree and the diameter of G in Section 5. For the time being, let's assume they are given. Now our goal is to produce a drawing of G according to the embedding β and at the same time keep many of the vertices of Π at their positions in δ_0. Having all chords on one side is the crucial property of Π we use to achieve this. We allow ourselves to move all other vertices of G to any location we like. This gives us a lower bound on $\mathrm{fix}(G, \delta)$ in terms of the number l of vertices of Π. Our method is illustrated in Fig. 1.

Algorithm outline. Now we sketch our three-step method. Let C denote the set of chords of Π. We assume that these chords lie to the right of Π in the embedding β. (Note that "below" is not defined in an embedding.) Let V_{bot} denote the set of vertices of G that lie to the right of Π in β and let $V_{\mathrm{top}} = V \setminus (V_\Pi \cup V_{\mathrm{bot}})$. Note that u lies in V_{top}.

In step 1 of our algorithm we bring the vertices in V_Π from the position they have in δ_0 into the same ordering according to increasing x-coordinates as they appear along Π in β. This yields a new (usually non-plane) drawing δ_1 of G that maps Π on an x-monotone polygonal path Π_1. Now we can apply the Erdős–Szekeres theorem [3] that basically says that a sequence of l distinct integers always contains a monotone (increasing or decreasing) subsequence of length at least $\sqrt{l-1}+1 \geq \lceil \sqrt{l}\, \rceil$. Thus we can choose δ_1 such that at least $\sqrt{l}$ vertices of Π remain fixed. Let $F \subseteq V_\Pi$ be the set of the fixed vertices. Note that $\delta_1|_{V \setminus V_\Pi} = \delta_0$, see Fig. 1(c).

Once we have constructed Π_1 we have to find suitable positions for the vertices in $V_{\mathrm{top}} \cup V_{\mathrm{bot}}$. This is simple for the vertices in V_{top}: if we move vertex u, which lies on the outer face, far enough above Π_1, then the polygon P_1 bounded by Π_1 and by the edges us and ut will be *star-shaped*. Recall that a polygon P is called *star-shaped* if the interior of its *kernel* is non-empty, and the *kernel* of a clockwise-oriented polygon P is the intersection of the right half-planes induced by the edges of P. Now if P_1 is star-shaped, we have fulfilled one of the assumptions of the following result of Hong and Nagamochi [5] for drawing *triconnected* graphs, i.e., graphs that cannot be decomposed by removing two vertices. We would like to use their result in order to draw into P_1 the subgraph G^+_{top} of G induced by $V_{\mathrm{top}} \cup V_\Pi$ *excluding* the chords in C.

Theorem 1 ([5]). *Given a triconnected plane graph H, every drawing δ^* of the outer facial cycle of H on a star-shaped polygon P can be extended in linear time to a plane drawing of H (even one where all inner faces are convex).*

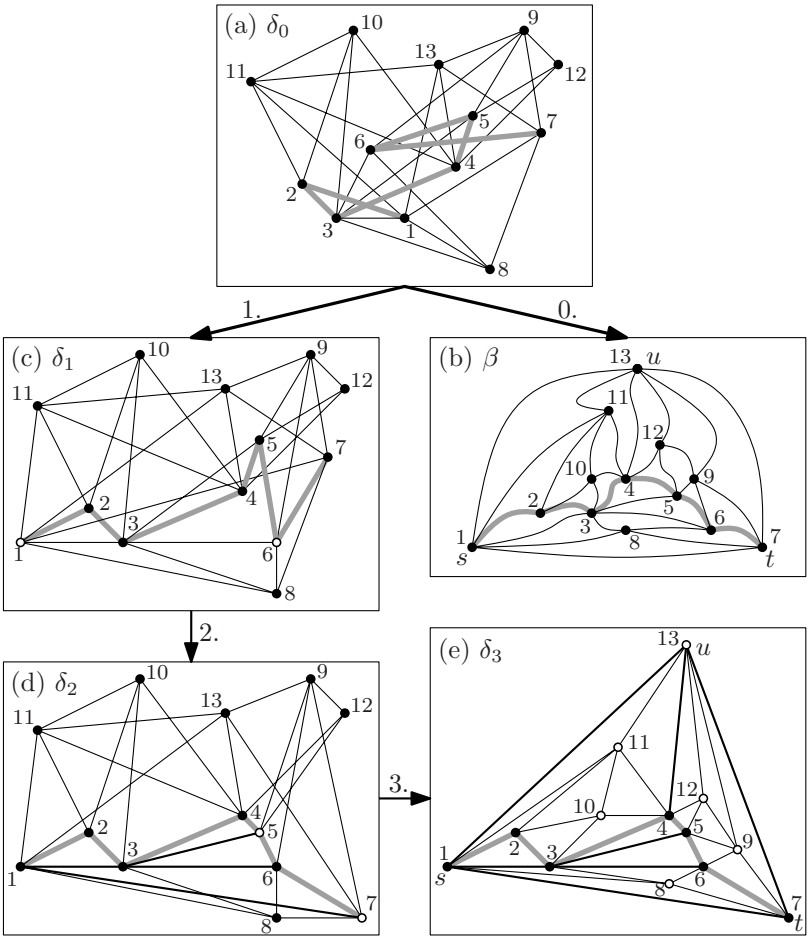


Fig. 1. An example run of our algorithm: (a) input: the given non-planar drawing δ_0 of a triangulated planar graph G. (b) Plane embedding β of G with path Π (drawn gray) that connects two vertices on the outer face. To make δ_0 plane we first make Π x-monotone (c), then we bring all chords (bold segments) to one side of Π (d), move u to a position where u sees all vertices in V_Π, and finally move the vertices in $V \setminus (V_\Pi \cup \{u\})$ to suitable positions within the faces bounded by the bold gray and black edges (e). Vertices that (do not) move from δ_{i-1} to δ_i are marked by circles (disks).

Observe that G^+_{top} is *not* necessarily triconnected: vertex u may be adjacent to vertices on Π other than s and t. But what about the subgraphs of G^+_{top} bounded (in β) by Π and edges of type uw_i, where $(s =)w_1, w_2, \ldots, w_l(= t)$ is the sequence of vertices of Π? Recall that a planar graph H is called a *rooted triangulation* [1] if in every plane drawing of H there exists at most one facial cycle with more than three vertices. According to Avis [1] the result stated in the

following lemma is well known. It can be shown using Tutte's characterization of triconnected graphs [13].

Lemma 1 ([1]). *A rooted triangulation is triconnected if and only if no facial cycle has a chord.*

Now it is clear that we can apply Theorem 1 to draw each subgraph of G^+_{top} bounded by Π and by the edges of type uw_i. By the placement of u, each drawing region is star-shaped, and by construction, each subgraph is chordless and thus triconnected. However, to draw the graph G^+_{bot} induced by $V_{\text{bot}} \cup V_\Pi$ (*including* the chords in C) we must work a little harder.

In step 2 of our algorithm we once more change the embedding of Π. We first carefully pick a subset V^* of vertices of Π. On the one hand V^* contains at least one endpoint of each chord in C. On the other hand V^* contains only a fixed fraction of the vertices in F, the subset of V_Π that δ_1 leaves fixed. Then we go through the vertices in V^* in a certain order, moving each vertex vertically down as far as necessary (see vertices 5 and 7 in Fig. 1(d)) to achieve two goals: (a) all chords in C move below the resulting polygonal path Π_2, and (b) the faces bounded by Π_2, the edge st, and the chords become star-shaped polygons. This defines a new drawing δ_2, which leaves a third of the vertices in F and all vertices in $V \setminus V_\Pi$ fixed.

In step 3 we use that Π_2 is still x-monotone. This allows us to move vertex u to a location above Π_2 where it can see every vertex of Π_2. Now Π_2, the edges of type uw_i (with $1 < i < l$) and the chords in C partition the triangle ust into star-shaped polygons with the property that the subgraphs of G that have to be drawn into these polygons are all rooted triangulations, and thus triconnected. This means that we can apply Theorem 1 to each of them. The result is our final—and plane—drawing δ_3 of G, see Fig. 1(e).

4 The Main Theorem

Recall that F is the set of vertices in V_Π we kept fixed in the step 1, i.e., in the construction of the x-monotone polygonal path Π_1. Our goal is to keep a constant fraction of the vertices in F fixed when we construct Π_2, which also is an x-monotone polygonal path, but has two additional properties: (a) all chords in C lie below Π_2 and (b) the faces induced by Π, w_1w_l, and the chords in C are star-shaped polygons. The following lemmas form the basis for the proof of our main theorem (Theorem 2), which shows that this can be achieved. For the proofs refer to the long version [11].

Lemma 2. *Let $\Pi = v_1, \ldots, v_k$ be an x-monotone polygonal path such that (i) the segment v_1v_k lies below Π and (ii) the polygon P bounded by Π and v_1v_k is star-shaped. Let v'_k be any point vertically below v_k. Then the polygon $P' = v_1, \ldots, v_{k-1}, v'_k$ is also star-shaped.*

Lemma 3. *Let $\Pi = v_1, \ldots, v_k$ be an x-monotone polygonal path and let D be a set of pairwise non-crossing straight-line segments with endpoints in V_Π that all*

lie below Π. Let v'_k be a point vertically below v_k, let $\Pi' = v_1, \ldots, v_{k-1}, v'_k$, and finally let D' be a copy of D with each segment $wv_k \in D$ replaced by wv'_k.

Then the segments in D' are pairwise non-crossing and all lie below Π'.

Lemma 4. *Let $\Pi = v_1, \ldots, v_k$ be an x-monotone polygonal path. Let C_Π be a set of chords of Π that can be drawn as non-crossing curved lines below Π. Let G_Π be the graph with vertex set V_Π and edge set $E_\Pi \cup C_\Pi$. Let V^* be a vertex cover of the edges in C_Π. Then there is a way to modify Π by decreasing the y-coordinates of the vertices in V^* such that the resulting straight-line drawing δ^* of G_Π is plane, the bounded faces of δ^* are star-shaped, and all edges in C_Π lie below the modified polygonal path Π.*

Note that the vertices in the complement of V^* remain fixed and that the modified polygonal path Π is x-monotone, too.

Proof. We use induction on the number m of chords. If $m = 0$, we need not modify Π. So, suppose $m > 0$. We first choose a chord $vw \in C_\Pi$ with $x(v) < x(w)$ such that there is no other edge $v'w' \in C_\Pi$ with the property that $x(v') \leq x(v)$ and $x(w') \geq x(w)$. Clearly, such an edge always exists. Then we apply the induction hypothesis to $C_\Pi \setminus \{vw\}$. This yields a modified path Π' of Π such that all edges in the resulting straight-line drawing of $G_\Pi - vw$ lie below Π' and all bounded faces in this drawing are star-shaped. Now consider the chord vw and let f be the new bounded face that results from adding vw. Without loss of generality we assume that $v \in V^*$. According to Lemmas 2 and 3 we can move v downwards from its position in Π' as far as we like. Hence, we can make the face f star-shaped without destroying this property for the other faces. □

Now suppose we have modified the x-monotone path Π_1 according to Lemma 4. Then the resulting x-monotone path Π_2 admits a straight-line drawing of the chords in C below Π_2 such that the bounded faces are star-shaped polygons, see for the example in Fig. 1(d). Recall that $u \in V_{\text{top}}$ is the vertex of the outer triangle in β that does not lie on Π. We now move vertex u to a position above Π_2 such that all edges $uw \in E$ with $w \in V_\Pi$ can be drawn without crossing Π_2 and such that the resulting faces are star-shaped polygons. Since Π_2 is x-monotone, this can be done. As an intermediate result we obtain a plane straight-line drawing of a subgraph of G where all bounded faces are star-shaped. It remains to find suitable positions for the vertices in $(V_{\text{top}} \setminus \{u\}) \cup V_{\text{bot}}$. For every star-shaped face f there is a unique subgraph G_f of G that must be drawn inside this face. Note that by our construction every edge of G_f that has both endpoints on the boundary of f must actually be an edge of the boundary. Therefore, G_f is a rooted triangulation where no facial cycle has a chord. Now Lemma 1 yields that G_f is triconnected. Finally, we can use the result of Hong and Nagamochi [5] (see Theorem 1) to draw each subgraph of type G_f and thus finish our construction of a plane straight-line drawing of G, see the example in Fig. 1(e). Let's summarize.

Theorem 2. *Let G be a triangulated planar graph that contains a simple path $\Pi = w_1, \ldots, w_l$ and a face uw_1w_l. If G has an embedding β such that uw_1w_l is*

the outer face, u lies on one side of Π, and all chords of Π lie on the other side, then $\mathrm{fix}(G) \geq \sqrt{l}/3$.

Proof. We continue to use the notation introduced earlier in this section. Recall that F is the set of vertices that we kept fixed in the first step, that is in the construction of the x-monotone path Π_1. It follows from [8, Proposition 1] that we can make sure that $|F| \geq \sqrt{l}$, where l is the number of vertices of the path Π we started with. Further recall that C is the set of chords of Π. Now let C' be the subset of those chords in C that have both endpoints in F. Consider the graph H induced by the edges in C' on F. For example, in Fig. 1(c), $C = \{17, 13, 35, 36\}$, $F = \{2, 3, 4, 5, 7\}$, $C' = \{35\}$ and $H = (\{3, 5\}, C')$. Since H is outerplanar, it is easy to color H with three colors: the dual of H without the vertex for the outer face consists of trees each of which can be processed by, say, breadth-first search. The union U of the smallest two color classes is a vertex cover of H of size at most $2|F|/3$. Now let $V^* = (V_\Pi \setminus F) \cup U$. Then every chord in C has at least one of its endpoints in V^* and $|V^* \cap F| = |U| \leq 2/3|F|$. Hence, by Lemma 4 at least a third of the vertices in F remain fixed when we construct the x-monotone path Π_2. In the remaining steps of our construction, i.e., when placing the vertices in V_{top} and V_{bot}, none of the vertices in $F \setminus U$ is moved. Hence, $\mathrm{fix}(G) \geq |F \setminus U| \geq |F|/3 \geq \sqrt{l}/3$. □

5 Finding a Suitable Path

In this section we present two strategies for finding a suitable path Π. They both do not depend on the geometry of the given drawing δ_0 of G. Instead, they exploit the graph structure of G. The first strategy works well if G has a vertex of large degree and, even though it is very simple, yields asymptotically tight bounds for outerplanar graphs.

Lemma 5. *Let G be a triangulated planar graph with maximum degree Δ. Then* $\mathrm{fix}(G) \geq \sqrt{\Delta}/3$.

Proof. Let u be a vertex of degree Δ and consider a plane embedding β of G where vertex u lies on the outer face. Since G is planar, such an embedding exists. Let $W = \{w_1, \ldots, w_\Delta\}$ be the neighbors of u in β sorted clockwise around u. This gives us the desired polygonal path $\Pi = w_1, \ldots, w_\Delta$ that has no chords on the side that contains u. Thus Theorem 2 yields $\mathrm{fix}(G) \geq \sqrt{\Delta}/3$. □

Lemma 5 yields a lower bound for outerplanar graphs that is asymptotically tight as we will see in the next section.

Corollary 1. *Let G be an outerplanar graph with n vertices. Then* $\mathrm{fix}(G) \geq \sqrt{n-1}/3$.

Proof. We select an arbitrary vertex u of G. Since G is outerplanar, we can triangulate G in such a way that in the resulting triangulated planar graph G' vertex u is adjacent to every other vertex in G'. Thus the maximum degree of a vertex in G' is $n - 1$. □

Our second strategy works well if the diameter d of G is large.

Lemma 6. *Let G be a triangulated planar graph of diameter d. Then* $\mathrm{fix}(G) \geq \sqrt{2d-1}/3$.

Proof. We choose two vertices s and v such that a shortest s–v path has length d. We compute any plane embedding of G that has s on its outer face. Let t and u be the neighbors of s on the outer face. Recall that a *Schnyder wood* (or *realizer*) [10] of a triangulated plane graph is a (special) partition of the edge set into three spanning trees each rooted at a different vertex of the outer face. Edges can be viewed as being directed to the corresponding roots. The partition is special in that the cyclic pattern in which the spanning trees enter and leave a vertex is the same for all inner vertices. Schnyder [10] showed that this cyclic pattern ensures that the three unique paths from a vertex to the three roots are vertex-disjoint and chordless. Let π_s, π_t, and π_u be the "Schnyder paths" from v to s, t, and u, respectively. Note that the length of π_s is at least d, and the lengths of π_t and π_u are both at least $d-1$. Let Π be the path that goes from s along π_s to v and from v along π_t to t. The length of Π is at least $2d-1$. Note that due to the existence of π_u the path Π has no chords on the side that contains u. Thus, Theorem 2 yields $\mathrm{fix}(G,\delta) \geq \sqrt{2d-1}/3$. □

Next we determine the trade-off between the two strategies above.

Theorem 3. *Let G be a planar graph with $n \geq 4$ vertices. Then* $\mathrm{fix}(G) \geq \frac{1}{3}\sqrt{\frac{2(\log n)-2}{\log\log n} - 1}$.

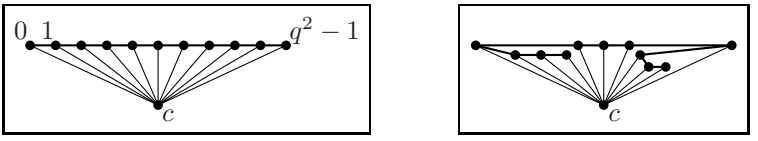
Proof. Let G' be an arbitrary triangulation of G. Note that the maximum degree Δ of G' is at least 3 since $n \geq 4$ and G' is triangulated. To relate Δ to the diameter d of G' we use a very crude counting argument—Moore's bound: starting from an arbitrary vertex of G we bound the number of vertices we can reach by a path of a certain length. Let j be the smallest integer such that $1 + (\Delta-1) + (\Delta-1)^2 + \cdots + (\Delta-1)^j \geq n$. Then $d \geq j$. By the definition of j we have $n \leq (\Delta-1)^{j+1}/(\Delta-2)$, which we can simplify to $n \leq 2(\Delta-1)^j$ since $\Delta \geq 3$. Hence we have $d \geq j \geq \frac{(\log n)-1}{\log(\Delta-1)}$.

Now, if $\Delta \geq \log n$, then Lemma 5 yields $\mathrm{fix}(G') \geq \sqrt{\log n}/3$. Otherwise $2d - 1 \geq \frac{2(\log n)-2}{\log\log n} - 1$, and we can apply Lemma 6. Observing that $\mathrm{fix}(G) \geq \mathrm{fix}(G')$ yields the desired bound. □

Remark 1. The proof of Theorem 3 (together with the auxiliary results stated earlier) yields an efficient algorithm for making a given straight-line drawing of a planar graph G with n vertices plane by moving some of its vertices to new positions. The running time is dominated by the time spent in the first step, i.e., computing the x-monotone path Π_1, which takes $O(n \log n)$ time [9]. The remaining steps of our method can be implemented to run in $O(n)$ time, including the computation of a Schnyder wood [10] needed in the proof of Lemma 6.

6 An Upper Bound for Outerplanar Graphs

In this section we want to show that the lower bound for outerplanar graphs in Corollary 1 is asymptotically tight. For a given positive integer q let H_q be the



(a) (b)

Fig. 2. The outerplanar graph H_q that we use in our upper-bound construction

outerplanar graph that consists of a path $0, 1, \ldots, q^2 - 1$ and an extra vertex $c = q^2$ that is connected to all other vertices, see Fig. 2(a). Note that H_q has many plane embeddings—e.g., Fig. 2(b)—but only two outerplane embeddings: Fig. 2(a) and its mirror image.

Let δ_q be the drawing of H_q where all vertices are placed on a horizontal line ℓ as follows. While vertex c can go to any (free) spot, vertices $0, \ldots, q^2 - 1$ are arranged in the order σ_q, namely

$$(q-1)q, (q-2)q, \ldots, 2q, q, \underline{0}, 1+(q-1)q, \ldots, 1+q, \underline{1}, \ldots, q^2-1, \ldots, (q-1)+q, \underline{q-1}.$$

The same sequence has been used by Goaoc et al. [4] to construct a planar (but not outerplanar) n-vertex graph G with $\mathrm{fix}(G) \leq \lceil \sqrt{n-2} \rceil + 1$.

We now make two observations about the structure of σ_q.

Observation 1 ([4]). *The longest increasing or decreasing subsequence of σ_q has length q.*

For the second observation let's define that two sequences Σ and Σ' of numbers *overlap* if $[\min(\Sigma), \max(\Sigma)] \cap [\min(\Sigma'), \max(\Sigma')] \neq \emptyset$.

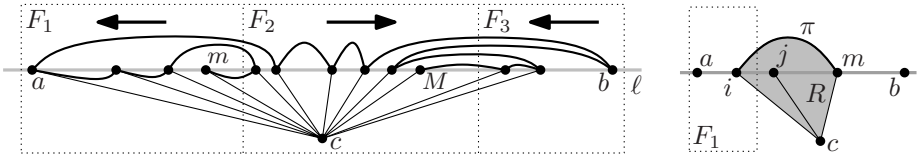
Observation 2. *Let Σ and Σ' be two non-overlapping decreasing or two non-overlapping increasing subsequences of σ_q. Then $|\Sigma \cup \Sigma'| \leq q + 1$.*

Proof. First consider the case that Σ and Σ' are both decreasing. Since they do not overlap we can assume without loss of generality that $\max(\Sigma) < \min(\Sigma')$. We define $V_i = \{iq + j : 0 \leq j \leq q - 1\}$ for $i = 0, \ldots, q - 1$. Then, since Σ and Σ' are both decreasing, they can each have at most one element in common with every V_i. Now suppose they have both one element in common with some V_{i_0}. Then, since $\max(\Sigma) < \min(\Sigma')$, Σ cannot have an element in common with any V_i, $i > i_0$, and Σ' cannot have an element in common with any V_i, $i < i_0$. Therefore, $|\Sigma \cup \Sigma'| \leq q + 1$.

Due to the symmetry of σ_q the case that Σ and Σ' are both increasing can be analyzed analogously. □

Given these observations we can now prove our upper bound on $\mathrm{fix}(H_q, \delta_q)$.

Theorem 4. *For any $q \geq 2$ it holds that $\mathrm{fix}(H_q, \delta_q) \leq 2q + 1 = 2\sqrt{n-1} + 1$, where $n = q^2 + 1$ is the number of vertices of H_q.*



(a) good sequence (b) bad sequence

Fig. 3. Analyzing the sequence of fixed vertices along the line ℓ

Proof. Let δ' be a plane drawing of H_q that maximizes the number of fixed vertices with respect to δ_q. Let F be the set of fixed vertices. Our proof exploits the fact that the simple structure of H_q forces the left-to-right sequence of the fixed vertices to also have a very simple structure.

Consider the drawing δ'. Clearly vertex c does not lie on ℓ. Thus we can assume that c lies below ℓ. Let a and b be the left- and rightmost vertices in F, respectively, and let m and M be the vertices with minimum and maximum index in F, respectively. Without loss of generality we can assume that m lies to the left of M, see Fig. 3(a).

We go through the vertices of F from left to right along ℓ. Let F_1 be the longest uninterrupted decreasing sequence of vertices in F starting from a. We claim that m is the last vertex in F_1. Assume to the contrary that $i \neq m$ is the last vertex of F_1, and let $j \in F$ be its successor on ℓ, see Fig. 3(b). If m is not the last vertex of F_1, then F_1 does not contain m. Thus m lies to the right of j. Consider the path $\pi = i, i-1, \ldots, m$. Since $j > i > m$, j is not a vertex of π. Clearly j lies below π, otherwise the edge jc would intersect π. Let R be the polygon bounded by π and by the edges ci and cm. Since δ' is plane, R is simple. Observe that j lies in the interior of R, which is shaded in Fig. 3(b). On the other hand, neither a nor b lies in the interior of R, otherwise the edge ac or the edge bc would intersect π.

We consider two cases. First suppose $j < a$. Then H_q contains the path $j, j+1, \ldots, a$ and we know that $a \neq i$ (since by definition of i and j we have $i < j$). Thus the path $j, j+1, \ldots, a$ does not contain any vertex incident to R. So it crosses some edge on the boundary of R. This contradicts δ' being plane. Now suppose $j > a$. Then H_q contains the path $j, j+1, \ldots, b$. In this case we can argue analogously since $m < b$ (otherwise m would lie to the right of M), reaching the same contradiction. Thus our assumption $i \neq m$ is wrong, and m is indeed the last vertex of F_1.

Now let F_2 be the longest uninterrupted *increasing* sequence of vertices in F starting from the successor of m. With similar arguments as above we can show that M is the last vertex in F_2. Finally let F_3 be the sequence of the remaining vertices from the successor of M to b. Again with similar arguments as above we can show that F_3 is decreasing.

The set F is partitioned by F_1, F_2, and F_3; F_2 is increasing, while F_1 and F_3 are decreasing. Thus Observations 1 and 2 yield $|F| \leq 2q + 1$ as desired. □

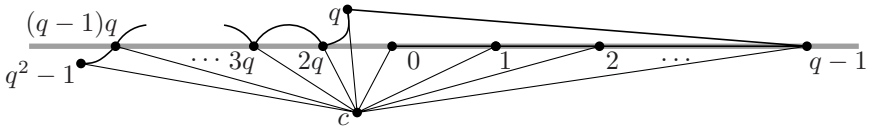


Fig. 4. This plane drawing of H_q shows that $\mathrm{fix}(H_q, \delta_q) \geq 2q-2$ since it keeps the vertices $0, 1, 2, \ldots, q-1, 2q, 3q, \ldots, (q-1)q$ of δ_q fixed

References

1. Avis, D.: Generating rooted triangulations without repetitions. Algorithmica 16, 618–632 (1996)
2. Bose, P., Dujmovic, V., Hurtado, F., Langerman, S., Morin, P., Wood, D.R.: A polynomial bound for untangling geometric planar graphs (October 2007), http://arxiv.org/abs/0710.1641
3. Erdős, P., Szekeres, G.: A combinatorial problem in geometry. Compos. Math. 2, 463–470 (1935)
4. Goaoc, X., Kratochvíl, J., Okamoto, Y., Shin, C.-S., Wolff, A.: Moving vertices to make drawings plane. In: Hong, S.-H., Nishizeki, T. (eds.) GD 2007. Proc. 15th Intern. Sympos. Graph Drawing. LNCS, vol. 4875, Springer, Heidelberg (to appear, 2008)
5. Hong, S.-H., Nagamochi, H.: Convex drawing of graphs with non-convex boundary. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 113–124. Springer, Heidelberg (2006)
6. Hopcroft, J., Tarjan, R.E.: Efficient planarity testing. J. ACM 21, 549–568 (1974)
7. Kang, M., Schacht, M., Verbitsky, O.: How much work does it take to straighten a plane graph out? (June 2007), http://arxiv.org/abs/0707.3373
8. Pach, J., Tardos, G.: Untangling a polygon. Discrete Comput. Geom. 28(4), 585–592 (2002)
9. Schensted, C.: Longest increasing and decreasing subsequences. Canadian Journal of Mathematics 13, 179–191 (1961)
10. Schnyder, W.: Embedding planar graphs on the grid. In: SODA 1990. Proc. 1st ACM-SIAM Symp. on Discrete Algorithms, pp. 138–148 (1990)
11. Spillner, A., Wolff, A.: Untangling a planar graph (September 2007), http://arxiv.org/abs/0709.0170
12. Tantalo, J.: Planarity (2007), http://planarity.net/
13. Tutte, W.T.: A theory of 3-connected graphs. Indagationes Mathematicae 23, 441–455 (1961)
14. Verbitsky, O.: On the obfuscation complexity of planar graphs (May & June 2007), http://arxiv.org/abs/0705.3748

Quantum Walks with Multiple or Moving Marked Locations

Andris Ambainis and Alexander Rivosh*

Department of Computer Science, University of Latvia, Raina bulv. 19, Riga, LV-1586, Latvia
andris.ambainis@lu.lv, alexander@biomed.lu.lv

Abstract. We study some properties of quantum walks on the plane. First, we discuss the behavior of quantum walks when moving marked locations are introduced. Second, we present an exceptional case, when quantum walk fails to find any of the marked locations.

1 Introduction

Quantum walks are quantum counterparts of random walks [2,9]. They have been useful for designing quantum algorithms for a variety of problems [6,3,13,4,10,5]. In many of those applications, quantum walks are used as a tool for search.

To solve a search problem using quantum walks, we introduce marked locations. Marked locations correspond to elements of the search space that we want to find. We then perform a quantum walk on search space with one transition rule at unmarked locations and another transition rule at marked locations (this process is also known as the *perturbed quantum walk*). If this process is set up properly, it leads to a quantum state in which marked locations have higher probability than the unmarked ones. This state can then be measured, finding a marked location with a sufficiently high probability. This method of search using quantum walks was first introduced in [12] and has been used many times since then.

Quantum walk search can be viewed as a generalization of Grover's quantum search algorithm [8]. Grover's search algorithm finds a solution in an unstructured search space of size N in $O(\sqrt{N})$ steps, by running a sequence of transformations on N-dimensional space that is different for the basis states corresponding to solutions and the basis states corresponding to non-solutions. It can be recast as a quantum walk on the complete graph with N vertices. By using quantum walks on other graphs, we can design search algorithms for specific problems (such as element distinctness [3]) which perform better than a simple application of Grover's search. As shown by Szegedy [13], one can also transform any reversible classical Markov chain into a quantum walk, with a quadratic speedup for the time in which the walk hits a marked location.

In this paper, we study the quantum walks on a finite two-dimensional grid of size $N \times N$. As shown in [4], after $O(N \log N)$ steps, a quantum walk with one or two marked locations reaches a state that is significantly different from

* Supported by University of Latvia Grant Y2-ZP01-100 and European Social Fund.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 485–496, 2008.

the state of a quantum walk with no marked location. [13] has generalized this to an arbitrary number of marked locations. Thus, a quantum walk on 2D grid can be used to detect the presence of a marked location, with a nearly-quadratic speedup over the best classical algorithm (which requires $\Theta(N^2)$ steps).

[4] also shows that, for one or two marked locations, measuring the state of the quantum walk after $O(N \log N)$ steps gives a marked location with probability at least $const/\log N$. Thus, for the case with one or two marked location, the quantum walk also finds a marked location. Similar results have been obtained for continuous time quantum walks by [7].

In this paper, we continue the study of the quantum walks on 2D grid in two directions. First, we look at the case when the marked locations are moving. Unlike the case with fixed marked locations, this case has no immediate applications to quantum algorithms. However, it shows some interesting phenomena. In particular, the ability of the quantum walk to detect the presence of moving marked locations strongly depends on the number of marked locations and their relative positions.

Second, we study the quantum walk on a 2-dimensional grid with multiple (non-moving) marked locations. As described above, it has been known that the quantum walk can detect the presence of a marked location [13]. It has been open question whether a single run of a quantum walk also finds a marked location. We show an example in which this is not the case.

2 Definitions

We consider quantum walks on a two-dimensional grid of size $N \times N$. The locations are labeled by their x and y coordinate as (x, y) for $x, y \in \{0, \ldots, N-1\}$. We assume that the grid has periodic boundary conditions. For example, going right from a location $(N-1, y)$ on the right edge of the grid leads to the location $(0, y)$ on the left edge of the grid.

Quantum walk is a quantum counterpart of a well-known process - classical random walk. In a classical random walk on the grid, we start in some location, for example $(0, 0)$. In each step we move from a location (x, y) to one of the neighbouring locations $(x-1, y)$, $(x+1, y)$, $(x, y-1)$, $(x, y+1)$ with probability $1/4$. (Since we have periodic boundary conditions, addition and multiplication are modulo N: $(N-1)+1 = 0$ and $0-1 = N-1$).

The most obvious way to quantize this random walk would be to define basis states $|i, j\rangle$, $i, j \in \{0, \ldots, N-1\}$, and let the state of the quantum walk $|\psi(t)\rangle$ be

$$|\psi(t)\rangle = \sum_{i,j} \alpha_i |i, j\rangle \tag{1}$$

To evolve the next step of quantum walk $|\psi(t+1)\rangle = U|\psi(t)\rangle$, we must have some unitary operator U. Similarly to the conventional random walk, it is natural to require that U maps each $|i, j\rangle$ to a superposition of $|i, j\rangle$ and the adjacent locations, in a way that is independent of i and j:

$$U|i, j\rangle = a|i-1, j\rangle + b|i, j\rangle + c|i+1, j\rangle + d|i, j-1\rangle + e|i, j+1\rangle \tag{2}$$

where a, b, c, d and e are independent of i, j. Unfortunately, there is no non-trivial unitary transformations U of this form. (This was shown for the quantum walk on the line in [11]. The proof carries over unchanged to the 2-dimensional grid).

To make it possible to apply non-trivial transformations, we add an additional "coin" register with four states, one for each direction: $|up\rangle$, $|down\rangle$, $|left\rangle$ and $|right\rangle$. At each step, we perform a unitary transform on the extra register, and then evolve the system according to the state of the coin register.

Basis states for the combined space are now $|i, j, d\rangle$ for $i, j \in \{0, \ldots, N-1\}$, $d \in \{up, down, left, right\}$ and the state of quantum walk is given by:

$$|\psi(t)\rangle = \sum_{i,j} (\alpha_{i,j,up}|i,j,up\rangle + \alpha_{i,j,down}|i,j,down\rangle$$

$$+\alpha_{i,j,left}|i,j,left\rangle + \alpha_{i,j,right}|i,j,right\rangle) \tag{3}$$

A step of the coined quantum walk is performed by first applying $1 \otimes C$, where C is a unitary transformation on the coin register. In this paper, the transformation on the coin register is the Grover's diffusion transformation D:

$$D = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \tag{4}$$

Then, we apply the shift transformation S which maps:

$$\begin{aligned} |i,j,up\rangle &\mapsto |i,j-1,down\rangle \\ |i,j,down\rangle &\mapsto |i,j+1,up\rangle \\ |i,j,left\rangle &\mapsto |i-1,j,right\rangle \\ |i,j,right\rangle &\mapsto |i+1,j,left\rangle \end{aligned}$$

Notice that, after moving to an adjacent location, we change the value of the direction register to the direction which is opposite to one from which we just came. This is necessary for the quantum walk algorithm of [4] to work. If we start our quantum walk in the state

$$|\psi(0)\rangle = \sum_{i,j} (\frac{1}{2N}|i,j,up\rangle + \frac{1}{2N}|i,j,down\rangle + \frac{1}{2N}|i,j,left\rangle + \frac{1}{2N}|i,j,right\rangle)$$

It can be easily verified that the state of the walk stays unchanged, regardless of the number of steps. For quantum walk search, we "mark" some locations. In unmarked locations, we apply the same transformations as above. In marked locations, we apply $-I$ instead of D as the coin flip transformation. The shift transformation remains the same in both marked and unmarked locations.

If there are marked locations, the state of this process starts to deviate from $|\psi(0)\rangle$. This means that we can detect the presence of a marked location by measuring whether the state of the quantum walk after t steps is $|\psi(0)\rangle$. This can be quantified, using the inner product (overlap) of quantum states.

Namely, a well known "folklore" lemma [1] says that

Lemma 1. *If $\langle\psi_1|\psi_2\rangle \geq 1-\epsilon$, then for any measurement M and any outcome i, the probability of finding i when measuring $|\psi_1\rangle$ and $|\psi_2\rangle$ differs by at most $\sqrt{2\epsilon}$.*

Thus, in order to be able to determine if there is a marked location, we need $\langle\psi(t)|\psi(0)\rangle$ to be small, where $|\psi(t)\rangle$ is the state of the quantum walk with marked locations after t steps and $|\psi(0)\rangle$ is the starting state (which is also the current state of the quantum walk with no marked locations, after any number of steps).

Detecting the existence of a marked location is different from determining which location is marked. It might be the case that the inner product $\langle\psi(t)|\psi(0)\rangle$ is small and it is difficult to determine which locations are marked. To measure the success of the algorithm in finding the marked location, we look at the probability of obtaining a marked location if the state of the quantum walk is measured after t steps.

3 Moving Marked Locations

We introduce moving marked locations. Their position on the plane depends on a number of performed steps. For example, if we have one marked location which is originally at (i, j) and is moving one position to the right each step, then, at step k, the location $(i + k, j)$ is marked and other locations are not marked.

The results in this section are obtained by computer simulation of quantum walks, except for Case 1 which is also proven analytically.

Case 1. We first examined the simplest case when only one moving marked location existed and there is no other marked locations - neither static, nor moving. The marked location was moving horizontally, by one position to the right per step (Figure 1).

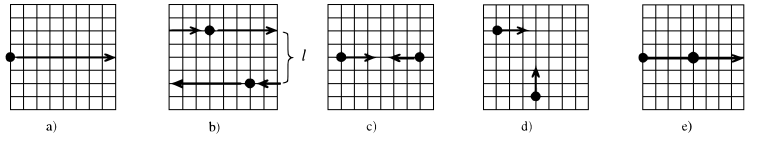


Fig. 1. Configurations of moving marked locations

The results of computer simulation in figure 2 show that the state of the quantum walk stays close to the starting state, for any number of steps. We now prove that this is indeed the case.

Theorem 1. *Let $|\Phi\rangle$ be the starting state of the quantum walk (the uniform superposition over all $|i, j, d\rangle$ and $|\Phi^t\rangle$ be the state of the quantum walk with one marked location, which is moving one location to the right each step, after t steps. Then,*

$$\langle\Phi^t|\Phi\rangle \geq 1 - \frac{2}{N} - \frac{2}{\sqrt{N}}.$$

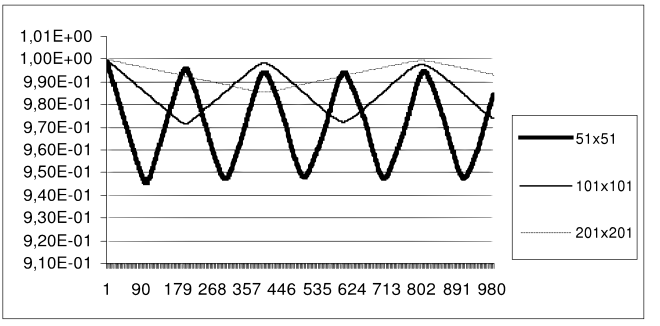


Fig. 2. Overlap between the starting state and the current state for case with one moving location, for various grid sizes. Number of steps: 1000.

Proof. We first notice that, instead of a moving marked location, we can move the grid and keep the marked location fixed. That is, we keep the marked location at the origin $(0, 0)$ and, after each step, move all amplitudes one position to the left: $|i, j, d\rangle \mapsto |i-1, j, d\rangle$. Then, the amplitude of $|i, j, d\rangle$ after t steps is the same as the amplitude of $|i, j + t, d\rangle$ if the marked location was moving to the right.

Consider the quantum state $|\Psi\rangle = \sum_{i,j,d} \alpha_{i,j,d}|i, j, d\rangle$, where

- $\alpha_{i,i,up} = 0$ and $\alpha_{i,j,up} = 1/2N$ for $i \neq j$.
- $\alpha_{i,-i,down} = 0$ and $\alpha_{i,j,down} = 1/2N$ for $i \neq -j$.
- $\alpha_{i,j,right} = 1/2N$ for all i, j.
- $\alpha_{i,j,left} = \alpha_{i,j,up} + \alpha_{i,j,down} - \alpha_{i,j,right}$.

We claim that the state $|\Phi\rangle$ is unchanged by the quantum walk with a marked location at $(0, 0)$ and the grid moving to the left. To see that, let $|\Phi'\rangle = \sum_{i,j,d} \alpha'_{i,j,d}|i, j, d\rangle$ be the state after applying the coin flip to $|\Phi\rangle$.

We claim that the effect of the coin flip is to switch $\alpha_{i,j,left}$ with $\alpha_{i,j,right}$ and $\alpha_{i,j,up}$ with $\alpha_{i,j,down}$. This follows separately for the marked location $(0, 0)$ and the unmarked locations. In the marked location, we start with $\alpha_{i,j,up} = \alpha_{i,j,down} = 0$, $\alpha_{i,j,left} = 1/N$ and

$$\alpha_{i,j,right} = \alpha_{i,j,up} + \alpha_{i,j,down} - \alpha_{i,j,left} = -1/2N$$

and apply $-I$ which results in $\alpha'_{i,j,up} = \alpha'_{i,j,down} = 0$, $\alpha'_{i,j,left} = -1/2N$ and $\alpha_{i,j,right} = 1/2N$. For the unmarked locations, we apply Grover's diffusion D which results in

$$\alpha'_{i,j,left} = \frac{\alpha_{i,j,left} + \alpha_{i,j,right} + \alpha_{i,j,up} + \alpha_{i,j,down}}{2} - \alpha_{i,j,left}$$

$$= \alpha_{i,j,up} + \alpha_{i,j,down} - \alpha_{i,j,left} = \alpha_{i,j,right}$$

and, similarly, $\alpha'_{i,j,right} = \alpha_{i,j,left}$, $\alpha'_{i,j,down} = \alpha_{i,j,up}$ and $\alpha'_{i,j,up} = \alpha_{i,j,down}$. We now consider the effect of the next two operations: the shift operator and moving

the grid one position left. Let $|\Psi''\rangle = \sum_{i,j,d} \alpha''_{i,j,d}|i,j,d\rangle$ be the state after those two operations. Then,

$$\alpha''_{i,j,up} = \alpha'_{i+1,j+1,down} = \alpha_{i+1,j+1,up}$$

Similarly,

$$\begin{aligned}
\alpha''_{i,j,down} &= \alpha'_{i+1,j-1,up} &&= \alpha_{i+1,j-1,down} \\
\alpha''_{i,j,right} &= \alpha'_{i+2,j,left} &&= \alpha_{i+2,j,right} \\
\alpha''_{i,j,left} &= \alpha'_{i+1,j+1,right} &&= \alpha_{i+1,j+1,left}.
\end{aligned}$$

By comparing this with the definitions of $\alpha_{i,j,d}$ we see that $\alpha''_{i,j,d} = \alpha_{i,j,d}$ for all i,j,d. Thus, the state $|\Psi\rangle$ is unchanged by the quantum walk.

Let $|\Phi\rangle$ be the uniform superposition over all $|i,j,d\rangle$. By a direct calculation,

$$\|\Psi\|^2 = \langle\Psi|\Phi\rangle = 1 - \frac{1}{N} + \frac{1}{2N^2}$$

for odd N and

$$\|\Psi\|^2 = \langle\Psi|\Phi\rangle = 1 - \frac{1}{N} + \frac{1}{N^2}$$

for even N. Let $|\psi\rangle$ be a normalized version of the state $|\Psi\rangle$: $|\psi\rangle = \frac{|\Psi\rangle}{\|\Psi\|}$. Then,

$$\langle\psi|\Phi\rangle = \frac{\langle\Psi|\Phi\rangle}{\|\Psi\|} \geq \sqrt{1 - \frac{1}{N} + \frac{1}{2N^2}} \geq 1 - \frac{1}{2N}.$$

Therefore, we can write $|\Phi\rangle = a|\psi\rangle + b|\phi\rangle$ for some state $|\phi\rangle$, with $a \geq 1 - 1/2N$ and

$$b = \sqrt{1-a^2} \leq \frac{1}{\sqrt{N}}.$$

Since the quantum walk leaves $|\Psi\rangle$ (and, therefore, also $|\psi\rangle$) unchanged, the state after t steps will be $|\Phi^t\rangle = a|\psi\rangle + b|\phi^t\rangle$ for some $|\phi^t\rangle$. We have

$$\langle\Phi^t|\Phi\rangle = a^2\langle\psi|\psi\rangle + ab\langle\phi|\psi\rangle + ab\langle\psi|\phi^t\rangle + b^2\langle\phi|\phi^t\rangle$$

$$\geq a^2 - ab - ab - b^2 \geq (1 - \frac{1}{N}) - \frac{2}{\sqrt{N}} - \frac{1}{N} = 1 - \frac{2}{\sqrt{N}} - \frac{2}{N}.$$

Thus, the overlap in this case always stays close to 1 (and the minimum overlap grows with size of grid). See figure 2, where the comparison for different grid sizes is shown. Theorem 1 and Lemma 1 together imply that probability to measure the marked location is also small (at most $\frac{2+o(1)}{\sqrt[4]{N}}$, by an application of Lemma 1, with $\epsilon \leq \frac{2}{\sqrt{N}} + \frac{2}{N}$ and even smaller according to our computer simulations, see figure 3).

Case 2. Then, we examined the case when two marked locations were moving horizontally on the parallel lines, one of them - by one position to the right per step, another one - in opposite direction (figure 1b). The results of our computer simulations were quite similar to results in case 1, i.e. the overlap

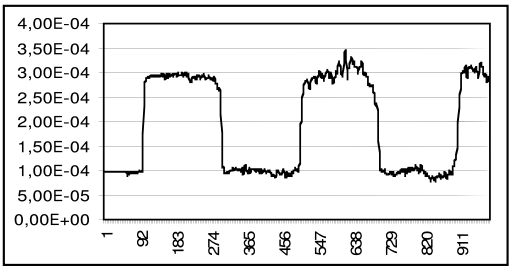


Fig. 3. Probability to measure a moving marked location. Grid size: 101×101. Number of steps:1000.

remains almost 1 and there were no states with large amplitude. With only one exception, described below, attempts to change the relative locations of the two moving locations did not give different results.

Case 3. Variation of initial distance between two marked locations lead us to the case when the distance between parallel lines is 0, i.e. these locations are moving in opposite directions on the same line (Figure 1c). We assume, that on the step when the positions of both marked locations are equal, there is only one marked location. Surprisingly, but in this case overlap meets 0 (Figure 4a). Moreover, the number of steps needed to reach that state on an $N \times N$ grid appears to be of the order $\Theta(N^{3/2})$ (Figure 5). But the probability to get one of the two marked locations as the result of measurement is still low (Figure 4b).

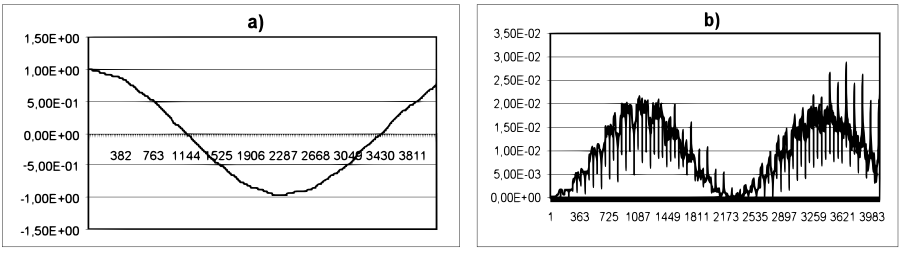


Fig. 4. a) Overlap for case with two moving locations, which are moving in the opposite directions on the same line. Grid size: 101x101. Number of steps: 4000. b) Sum of probabilities to get a moving location as the result of a measurement after the certain step.

Case 4. We also checked the case when marked locations are moving on perpendicular lines (Figure 1d) - with two sub-cases: when these locations never be in the same position and the case when there will be steps when positions of both locations are equal. Results were quite similar to the Case 2.

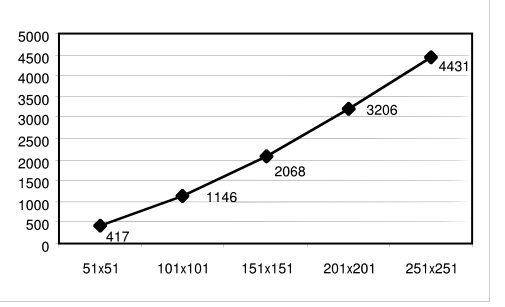


Fig. 5. Step on which overlap meets 0 depending on grid size

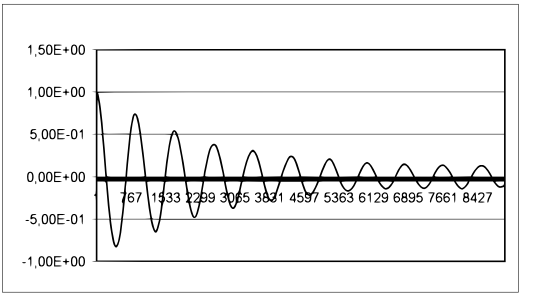


Fig. 6. Overlap for the case with one static and one moving marked locations. The static point lies on the line along which the other point moves. Number of steps: 9000. Grid size: 101×101.

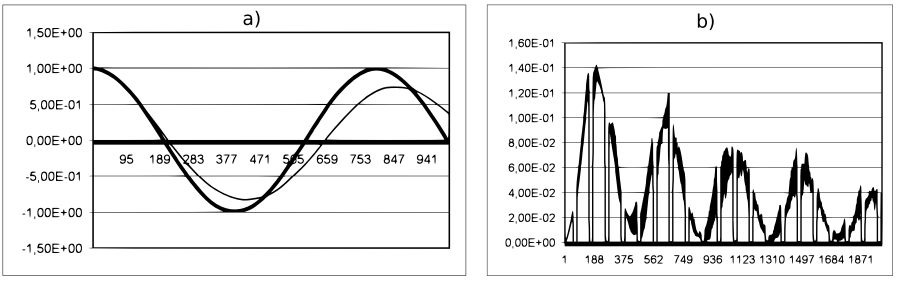


Fig. 7. a) Overlap for the case with one static and one moving marked locations compared to the case with only one static marked location. Number of steps: 1000. b) Sum of probabilities to get the static location as the result of a measurement after the certain step. Grid size: 101×101 for both cases.

Case 5. We introduced one static marked location and another one which is moving on horizontal line, by one position per step. In this case, the overlap reaches 0 quite fast. Interestingly, the state of the system depends on the distance

between the static and moving locations. If the static location is far from the line on which another location is moving, the behavior of the system looks like the case when only the static location is present. When the static location is close to the path of moving location, i.e. lies on the line another location is moving along (Figure 1e), the behavior changes. Namely, the oscillations of overlap are fading (Figure 6), unlike in the case with only the static location (where the overlap keeps oscillating between -1 and 1, see figure 7a for comparison).

4 Exceptional Case: Marked Locations on Diagonal

We now explore the quantum walks with multiple marked locations which are not moving. As shown by Szegedy [13], regardless of the number of marked locations, it is possible to detect if there is a marked location. More formally, if there is one or more marked locations, the overlap decreases to 0 or less in $O(N \log N)$ steps, for an appropriately defined quantum walk. This can then be used to detect the presence of a marked location. If we actually want to find the marked location, one solution is to run this algorithm multiple times, subdividing the grid into smaller and smaller pieces, until a marked location is uniquely determined. This works but is relatively complicated and requires multiple runs of quantum walk.

If there is just one or two marked locations, [4] have shown that a marked location can be found with probability at least $1/\log N$ by just running the quantum walk and measuring after t steps, for appropriately chosen $t = O(N \log N)$. Computer simulations show that the same approach works for almost any configuration of more than 2 marked locations. In this section, we show that this approach is not sufficient in general. We exhibit a case when the quantum walk algorithm of [4] does not find any of the locations with the probability larger than in their probability in the initial state. This happens when marked locations are placed on diagonal of the plane and fit the whole diagonal. Although the state of the quantum walk quickly becomes orthogonal to its starting state (Figure 8), the probabilities to get any state stay the same as in initial state. This is possible because the amplitudes of the states change their signs during the quantum walk.

Theorem 2. *If we run the quantum walk on the $N \times N$ grid with N marked locations on the diagonal for t steps, the probability of measuring a marked location at the end of the walk is $1/N$, regardless of the number of steps.*

Proof. We assume that locations (i, i) are marked and all other locations are unmarked. We define

$$|\Psi_{j,left}\rangle = \sum_i \frac{1}{\sqrt{2N}}|i, i+j, left\rangle + \frac{1}{\sqrt{2N}}|i, i+j, down\rangle, \tag{5}$$

$$|\Psi_{j,right}\rangle = \sum_i \frac{1}{\sqrt{2N}}|i, i+j, right\rangle + \frac{1}{\sqrt{2N}}|i, i+j, up\rangle. \tag{6}$$

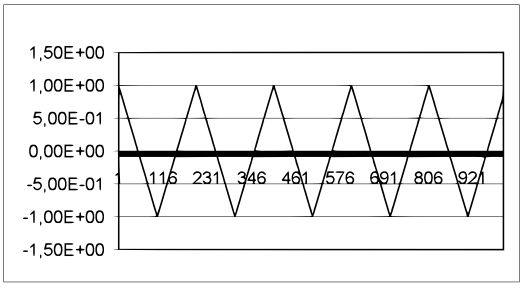


Fig. 8. Overlap for the case, when a diagonal of the grid is filled with marked locations (101 marked locations). Grid size: 101×101. Number of steps: 1000.

Then, the starting state is equal to

$$|\Psi_{start}\rangle = \sum_j \frac{1}{\sqrt{2N}}|\Psi_{j,left}\rangle + \frac{1}{\sqrt{2N}}|\Psi_{j,right}\rangle.$$

One step of a quantum walk consists of the coin flip C (which is equal to the diffusion operator in unmarked locations and $-I$ in marked locations) and the shift operator S.

Applying the diffusion transformation D to $|i,j,0\rangle = \frac{1}{\sqrt{2}}|i,j,left\rangle + \frac{1}{\sqrt{2}}|i,j,down\rangle$ gives the state $|i,j,1\rangle = \frac{1}{\sqrt{2}}|i,j,right\rangle + \frac{1}{\sqrt{2}}|i,j,up\rangle$. Similarly, applying D to $|i,j,1\rangle$ gives $|i,j,0\rangle$. Therefore, D transforms $|\Psi_{j,left}\rangle$ (which is the superposition of $|i,j,0\rangle$ with equal amplitudes) to $|\Psi_{j,right}\rangle$ (which is the superposition of $|i,j,1\rangle$ with equal amplitudes) and $|\Psi_{j,right}\rangle$ to $|\Psi_{j,left}\rangle$. In the marked locations (i,i), instead of D, we apply $-I$ which transforms $|i,i,0\rangle$ to $-|i,i,0\rangle$ and $|i,i,1\rangle$ to $-|i,i,1\rangle$. Thus, $|\Psi_{j,left}\rangle$ (which is the superposition of $|i,i,0\rangle$ with equal amplitudes) is mapped to $-|\Psi_{j,left}\rangle$ and $|\Psi_{j,right}\rangle$ is mapped to $-|\Psi_{j,left}\rangle$.

The shift operation S maps $|i,i+j,left\rangle$ and $|i,i+j,down\rangle$ to $|i-1,i+j,left\rangle$ and $|i,i+j+1,up\rangle$. Both $|i-1,i+j,left\rangle$ and $|i,i+j+1,up\rangle$ are components of $|\Psi_{j+1,right}\rangle$. Therefore, $|\Psi_{j,left}\rangle$ is mapped to $|\Psi_{j+1,right}\rangle$. Similarly, $|\Psi_{j,right}\rangle$ is mapped to $|\Psi_{j-1,left}\rangle$.

Together, these two transformations have the following effect:

$$\begin{aligned} |\Psi_{0,left}\rangle &\xrightarrow{C} -|\Psi_{0,left}\rangle \xrightarrow{S} -|\Psi_{1,right}\rangle, \\ |\Psi_{0,right}\rangle &\xrightarrow{C} -|\Psi_{0,right}\rangle \xrightarrow{S} -|\Psi_{-1,left}\rangle \end{aligned}$$

and

$$\begin{aligned} |\Psi_{j,left}\rangle &\xrightarrow{C} |\Psi_{j,right}\rangle \xrightarrow{S} -|\Psi_{j-1,left}\rangle, \\ |\Psi_{j,right}\rangle &\xrightarrow{C} |\Psi_{j,left}\rangle \xrightarrow{S} |\Psi_{j+1,right}\rangle \end{aligned}$$

for $j \neq 0$. If we repeat this t times, for $t < N$, the starting state

$$|\Psi_{start}\rangle = \sum_j \frac{1}{\sqrt{2N}}|\Psi_{j,left}\rangle + \frac{1}{\sqrt{2N}}|\Psi_{j,right}\rangle.$$

is mapped to

$$|\Phi^t\rangle = -\sum_{j=1}^{t} \frac{1}{\sqrt{2N}}|\Psi_{j,right}\rangle + \sum_{j=t+1}^{N} \frac{1}{\sqrt{2N}}|\Psi_{j,right}\rangle$$

$$-\sum_{j=N-t}^{N-1} \frac{1}{\sqrt{2N}}|\Psi_{j,left}\rangle + \sum_{j=0}^{N-t+1} \frac{1}{\sqrt{2N}}|\Psi_{j,left}\rangle.$$

After N steps, all components of $|\Psi_{start}\rangle$ acquire a - sign and the state becomes $-|\Psi_{start}\rangle$. Hence, if we run the walk for $M = kN + t$, $t < N$ steps, the resulting state is just $(-1)^k|\Phi^t\rangle$. Since the marked locations are (i, i), for all i, the states $|\Psi_{0,left}\rangle$ and $|\Psi_{0,right}\rangle$ consist of marked locations only and the states $|\Psi_{j,left}\rangle$ and $|\Psi_{j,right}\rangle$, $j > 0$, consist of unmarked locations only. For any t, the probability of the states $|\Psi_{0,left}\rangle$ and $|\Psi_{0,right}\rangle$ is $\frac{1}{2N}$ each. Thus, the probability of measuring a marked location is $\frac{1}{2N} + \frac{1}{2N} = \frac{1}{N}$.

5 Conclusion and Open Problems

In this paper, we explored quantum walks on the two-dimensional grid in which some locations are marked. First, we considered the case when the marked locations are moving. In contrast to the previous research on fixed marked locations, having a moving marked location makes very little difference compared with the quantum walk with no marked locations. An exception to this general patterns is when two marked locations are moving on the same line. Then, the quantum walk leads to a state which is significantly different from the case with no marked locations.

Three open problems on moving marked locations are:

1. Our results mostly rely on computer simulations. It would be good to develop more methods for rigorous analysis of the walk with moving marked locations
2. What are the other cases when the quantum walk behaves similarly to the case with two locations moving on the same line?
3. Does the walk with moving marked locations have algorithmic applications?

We also considered the case with both fixed and moving marked locations. Then, the behavior of the quantum walk is almost the same as in the case when there is only a fixed marked location. It leads to a state orthogonal to the starting state in approximately the same number of steps and finds the fixed marked location with a similar probability. The probability of finding the moving location is negligible.

Second, we considered the quantum walk with multiple fixed locations. As shown in [13], this walk always detects the presence of a marked location. We explored whether it finds the marked location. In most cases, the walk finds one of the marked locations with a good probability, but we found an example (N marked locations on a diagonal of $N \times N$ grid) in which it fails to find a marked location.

References

1. Aharonov, D., Kitaev, A., Nisan, N.: Quantum Circuits with Mixed States. In: Proceedings of STOC 1998, pp. 20–30 (1998)
2. Ambainis, A.: Quantum walks and their algorithmic applications. International Journal of Quantum Information 1, 507–518 (2003)
3. Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM J. Comput. 37(1), 210–239 (2007)
4. Ambainis, A., Kempe, J., Rivosh, A.: Coins make quantum walks faster. In: Proceedings of SODA 2005, pp. 1099–1108 (2005)
5. Buhrman, H., Špalek, R.: Quantum Verification of Matrix Products. In: SODA 2006. Proceedings of 17th Annual ACM-SIAM Symposium on Discrete Algorithms, Miami, Florida, pp. 880–889 (2006)
6. Childs, A.M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., Spielman, D.A.: Exponential algorithmic speedup by a quantum walk. In: Proceedings of the 35th ACM STOC, pp. 59–68 (2003)
7. Childs, A., Goldstone, J.: Spatial search and the Dirac equation. Physical Review A 70, 042312 (2004)
8. Grover, L.: A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th ACM STOC, Philadelphia, Pennsylvania, pp. 212–219. ACM Press, New York (1996)
9. Kempe, J.: Quantum random walks - an introductory overview. Contemporary Physics 44(4), 302–327 (2003)
10. Magniez, F., Santha, M., Szegedy, M.: An $O(n^{1.3})$ quantum algorithm for the triangle problem. In: Proceedings of SODA 2005, pp. 1109–1117 (2005), SIAM J. Comput. 37(2), 413–424 (2007)
11. Meyer, D.: From quantum cellular automata to quantum lattice gases. Journal of Statistical Physics 85, 551–574 (1996)
12. Shenvi, N., Kempe, J., Whaley, K.B.: A quantum random walk search algorithm. Physical Review A 67(5), 052307 (2003)
13. Szegedy, M.: Quantum speed-up of Markov Chain based algorithms. In: Proceedings of IEEE FOCS 2004, pp. 32–41 (2004)

Parallel Immune System for Graph Coloring

Jacek Dąbrowski

Gdańsk University of Technology
ul. Gabriela Narutowicza 11/12, 80-952 Gdańsk, Poland
Jacek.Dabrowski@eti.pg.gda.pl

Abstract. This paper presents a parallel artificial immune system designed for graph coloring. The algorithm is based on the clonal selection principle. Each processor operates on its own pool of antibodies and a migration mechanism is used to allow processors to exchange information. Experimental results show that migration improves the performance of the algorithm. The experiments were performed using a high performance cluster on a set of well-established graph instances available on the Web.

1 Introduction

A coloring of a graph $G = (V, E)$, where V is the set of $n = |V|$ vertices and E is the set of edges, is a mapping $c : V \mapsto 1..k$, such that for each edge $\{u, v\} \in E$ we have $c(u) \neq c(v)$. Optimization version of GCP is stated as follows: given a graph G, find a coloring with the minimum number k of colors used. This number is referred to as $\chi(G)$, the *chromatic number of graph G*. The GCP is a well-known NP-hard combinatorial optimization problem.

Artificial immune systems (AIS) can be defined as computational systems inspired by theoretical immunology, observed immune functions, principles and mechanisms designed to solve problems [5]. In recent years AIS have been applied to many different fields of computation, including pattern recognition [4], anomaly detection [6,8] and optimization.

This paper presents an algorithm based on clonal selection - a mechanism that allows lymphocytes to adapt to new, previously unencountered pathogens. This biological process is briefly described in the next section. Section 3 gives details of the implementation of the graph coloring algorithm. Section 4 presents experimental results on well-established benchmarking graph instances available from the Web - the 2nd DIMACS Implementation Challenge [1] and Michael Trick's Graph Coloring Resources [2]. The results are summarized in section 5.

2 Biological Inspirations

The function of the immune system is the defence of an organism from pathogens, i.e. self and non-self agents that could impair its functioning. Some of those

[1] http://dimacs.rutgers.edu/Challenges/
[2] http://mat.gsia.cmu.edu/COLOR/color.html

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 497–505, 2008.

agents are stopped by the physical barrier of skin and some by the physiological conditions inside the body. Pathogens that cross that first line of defence should be stopped by the innate and adaptive immune systems. Phagocyte cells of the innate IS are the cleaning crew of an organism - they can kill intruders and remove the remains from the system. Phagocytes identify the objects to be removed by specific three dimensonal features on their surface called *epitopes*. While a response to some types of pathogens is built-in in the innate IS, it is impossible to pack all the information needed to identify and destroy constantly mutating bacteria and viruses into the relatively short genome.

The adaptive immune system uses lymphocyte cells to mark pathogens for removal. There are two types of lymphocyte cells: T cells created in thymus and B cells created in bone marrow. Each lymphocyte recognizes a specific shape of an epitope, and a special developement path from stem cells arms the IS with millions of types of lymphocytes [9].

T cells perform vital functions in the immune response, but it is the learning process of clonal selection of B cells that inspired most of the AIS research. B cells defend the organism by launching antibodies on pathogens. Antibodies are protein strands that attach themselves to specific epitopes and mark the pathogens for removal. Stimulated B cells, with additional signals from accessory cells, start dividing (cloning) and maturing into terminal (non-dividing) antibody secreting cells. The rate of this process depends on the level of stimulation, B cells that recognize the antigen most acurately have the most clones and produce the most antibodies [3].

The dividing and maturing B cells undergo a process called somatic hypermutation that further improves the response of the immune system. The protein strands that recognize pathogens are slightly modified, and if a newly created lymphocyte is a better fit for the invading pathogen it divides (clones itself) at

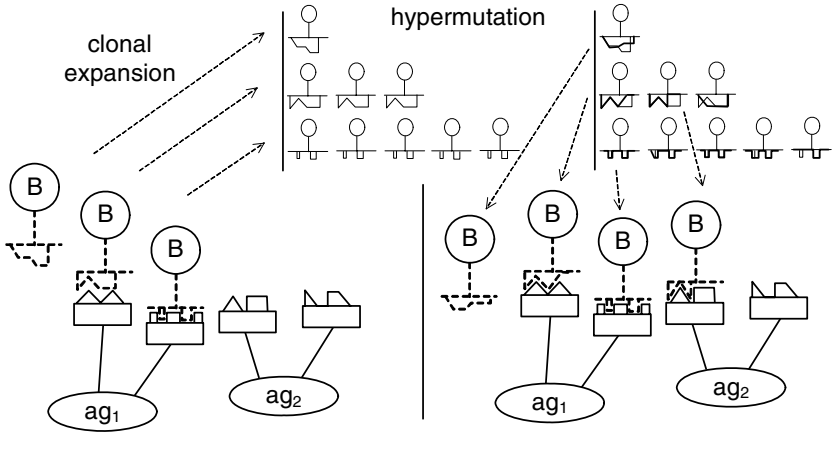


Fig. 1. Clonal selection mechanism

a higher rate. It has been observed that the extent of the mutation also depends on the level of stimulation - cells with low affinity may be mutated further.

This biological process is a basis for clonal selection algorithms, which use the clonal expansion and somatic hypermutation mechanisms on virtual antibodies to perform pattern recognition and optimization tasks. The analogy between pattern recognition and the function performed by the immune system is quite obvious. For optimization tasks a specific instance of the given problem becomes the single antigen, and the affinity of an antibody is measured by the quality of the solution it represents.

3 Clonal Selection Algorithm for GCP

The algorithm presented in this paper solves the optimization version of the graph coloring problem. It is done in steps - at each step the algorithm tries to lower the upper bound of the graphs chromatic number χ. The antibodies are initialized using a random assignment of colors to vertices or with DSATUR, a nondeterministic greedy heuristic [1]. The clonal selection process is designed to minimise the objective function, which is equal to the number of conflicts in a coloring. As soon as a valid coloring using k colors is found, the upper bound is lowered and all the antibodies in the pool are recolored so that they use at most $k-1$ colors.

Algorithm 1. Sibling selection

```
P_0 ← Initialize(G)                          ▷ generate initial antibody pool
t ← 0
while stop-condition do
    for all ab in P_t do
        n_c ← number of clones of ab
        clones ← φ
        for i = 0 to n_c do
            c ← Hypermutation(ab)
            clones ← clones ∪ c
        end for
        c ← Select(clones)
        P_{t+1} ← P_{t+1} ∪ c
    end for
    t ← t + 1
end while
```

Two variants of the selection process have been used. The difference is in the way the next-step antibody pool is selected. Sibling selection means that an antibody in the pool can only be replaced by its own clone. This approach is based on the CLONAFlex algorithm [10]. Such algorithm can be viewed as a method of computation effort management for independent searches, because

high-affinity clones of an antibody will never replace another antibody in the pool. The clone is chosen using tournament selection.

The second approach uses a single set for clones of all antibodies. The new pool is created by running N tournaments on this set. Here, clones of a poor quality antibody are less likely to be chosen than clones of high-affinity clone antibodies. This creates a higher selection pressure but can lead to loss of diversity.

Algorithm 2. Clones selection

```
P_0 ← Initialize(G)                          ▷ generate initial antibody pool
t ← 0
while stop-condition do
    clones ← φ
    for all ab in P_t do
        n_c ← number of clones of ab
        for i = 0 to n_c do
            c ← Hypermutation(ab)
            clones ← clones ∪ c
        end for
    end for
    P_{t+1} ← Select(clones)
    t ← t + 1
end while
```

All antibodies in the pool are ranked $1..N$ according to their affinity. The number of clones is given by formula:

$$n_c(i) = (1-r)\frac{N_c}{N} + r\frac{2N_c(N-i)}{N(N-1)} \ . \tag{1}$$

It distributes the total number of clones N_c among the N antibodies in the pool. The distribution is controlled through parameter $r \in< 0, 1 >$. It can be uniform, with all antibodies being cloned $\lfloor \frac{N_c}{N} \rfloor$ times ($r = 0$), it can be inversely proportional to the antibodies rank ($r = 1$), or it can be a combination of the two ($0 < r < 1$). The effect this parameter has on the performance of the algorithm is discussed in the next section.

The hypermutation mechanism changes the assignement of colors to vertices of the graph. The relative quality of an antibody determines the number of vertices that are affected during the mutation. The first vertex to be changed is chosen at random and vertices that belong to conflicting edges have a higher probability of being selected. Subsequent vertices are chosen from the neighbours of the last mutated vertex. Vertices are assigned with colors that result in the lowest number of conflicts.

To improve the performance a parallel version of the algorithm has been created. It uses an island model, where every processor works on its own pool of antibodies. A migration mechanism allows knowledge exchange between processes.

At predefined intervals each process chooses migrants using tournament selection. The receipient is either random or it is the next processor according to the MPI rank (cyclic migration). The receiving process chooses antibodies from its own pool to be replaced by the migrants using an inverted tournament selection (the worst antibody from each tournament 'wins' and becomes replaced). According to the experimental results there are no apparent differences between random and cyclic migration. During the benchmark runs the migration size was set to one (as the pool sizes were small), the tournaments size to three and the migration interval between five and thirty seconds (depending on the graph size and density).

4 Experimental Results

The results are based on a C++ implementation of the algorithm. The experiments were performed on a high performance cluster with 1.4 GHz Itanium 2 processors connected with an InfiniBand network.

4.1 Population Size

The size of the antibody pool is a very important parameter for evolutionary algorithms. A large population size will slow down the search process, too small will affect the diversity of solutions. Figure 2 presents an example of the performance of the clonal selection algorithm on `DSJC1000.5` graph. The number of clones created in each generation is 100, same for all population sizes.

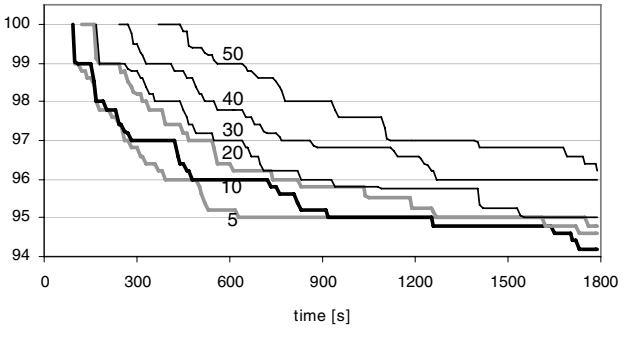


Fig. 2. Performance using varied pool sizes. Graph `DSJC1000.5`.

The results obtained show a good performance of the algorithm for relatively small population sizes when compared to other evolutionary techniques. With the number of clones within the tested range (40-300) a small population size of 20-40 antibodies seems to be a good choice for most graph instances.

4.2 The Number of Clones

The second factor that greatly influences the speed of the search is the number of clones generated each turn. The experiments have shown that regardless of the size of an instance the number of clones is best kept at a low level of 50 to 100 clones (for a pool size of 30). Figure 3 shows its impact on the performance of the algorithm.

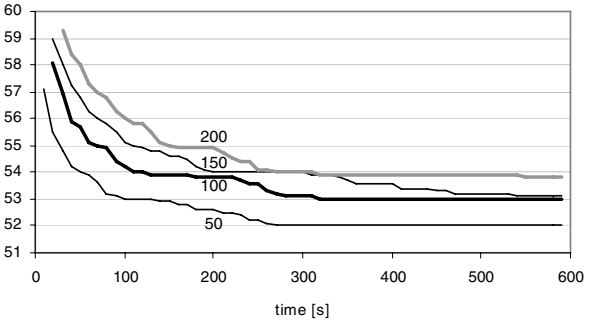


Fig. 3. Performance using varied numbers of clones. Graph `DSJC500.5`.

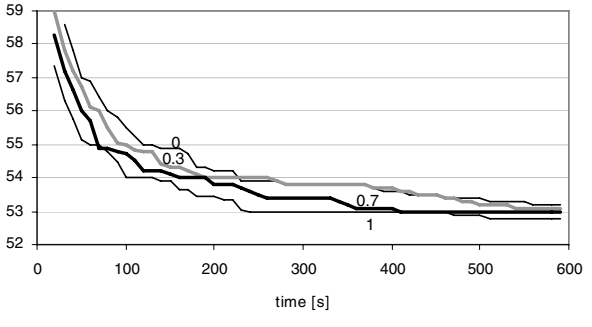


Fig. 4. Performance using varied clone distributions. Graph `DSJC500.5`.

Some authors [4] state that for the sake of simplicity one can create the same number of clones for all antibodies, regardless of their quality. Experimental results, e.g. figure 4, show that a simple formula based on the ranks of antibodies (and not the values of objective function), that changes the distribution of the number of clones, can improve the performance of the algorithm.

4.3 Parallel Speedup

When parallelizing heuristic methods one often can't use the classical speedup measurement because the behaviour of the serial and the parallel algorithms differ not only in terms of speed but also in the actual steps of the algorithm. This is the usual case with island-model evolutionary algorithms.

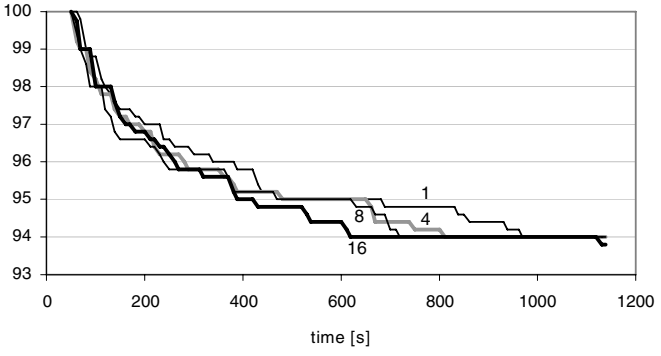


Fig. 5. Performance using varied numbers of processors

During the experiments it has been observed that the use of island-model parallelism increases the performance of the heuristic. There are two factors that can explain this increase. The first one is the simple fact, that if one runs more than one copy of the same heuristic process and always chooses the best solution produced by any of the copies then the quality of the selected solution will not be worse (and usually will be better) than the quality of a solution produced by any single run. The other factor is the migration mechanism that allows pools on different islands (processors) to exchange the best solutions. Figure 5 presents the average quality of the best solution found when coloring the `DSJC1000.5` graph using varying number of processors.

Experiments have shown that migration has a limited positive effect on the performance of the algorithm. The variance of the time needed to find a solution of a given quality between runs of the algorithm greatly outweights the difference made by varying frequency of migration.

4.4 Results for Benchmark Graphs

This section presents the performance of the parallel clonal selection algorithm (pAIS) on some of the well-established benchmark graphs compared with the performance of two other algorithms: greedy DSATUR [1] and parallel tabu search algorithm. The latter is a parallel version of *Tabucol*, a tabu search algorithm by Hertz and de Werra [7]. *Tabucol* uses a simple *1-exchange* neighborhood, where a move is a pair (v, i) denoting assignment of color i to vertex v. After a move is performed the pair (v, i) becomes a tabu move for $\lfloor 10 + 0.6f(x) \rfloor$ succeeding iterations, where f(x) is the number of color conflicts in the current solution x.

Results for some of the benchmark graphs are not included as those graphs are easily colorable. The book graphs and the graphs based on registry allocation can be quickly colored optimally using DSATUR.

Table 1 compares the results obtained using DSATUR, pAIS an pTS using 8 processors of the *holk* cluster. The result of nondeterministic DSATUR is the best out of ten runs, as each of them takes less than a second. The parameters of pAIS are specified below the table. There were ten runs for each graph instance.

Table 1. Performance of the algorithms on benchmark graphs

Graph	$\|V\|$	$\|E\|$	χ	DSATUR	pTS		pAIS	
					min	avg	min	avg
DSJC125.9	125	6961		51	44	44.8	44	44
DSJC125.5	125	3891		22	18	18	17	17
DSJC125.1	125	736		6	5	5	5	5
DSJC250.9	250	27897		92	73	73	72	72
DSJC250.5	250	15668		37	31	31	29	29
DSJC250.1	250	3218		10	9	9	9	9
DSJC500.9	500	112437		170	130	131	128	128.9
DSJC500.5	500	62624		65	53	53.2	51	51
DSJC500.1	500	12458		16	13	13	13	13
DSJC1000.9	1000	449449		299	243	243.8	241	242.4
DSJC1000.5	1000	249826		115	94	94.4	93	93
DSJC1000.1	1000	49629		27	22	22	21	21
DSJR500.1	500	3555		13	12	12	12	12
DSJR500.1c	500	121275		90	85	85	85	85
DSJR500.5	500	58662		130	124	125.2	123	123.6
QUEEN14_14	196	8372		19	16	16	15	15
QUEEN15_15	225	10360		21	17	17	16	16
QUEEN16_16	256	12640		23	17	17.9	17	17
latin_square_10	900	307350		132	105	105.8	104	104.7
flat300_20_0	300	21375	20	39	20	20	20	20
flat300_26_0	300	21633	26	41	34	34	32	32
flat300_28_0	300	21695	28	42	34	34	32	32
flat1000_50_0	1000	245000	50	113	92	92	90	90.8
flat1000_60_0	1000	245830	60	116	93	93.2	91	91.4
flat1000_76_0	1000	246708	76	114	93	93.5	92	92
le_450_15a	450	8168	15	17	16	16	16	16
le_450_15b	450	8169	15	16	15	15	15	15
le_450_15c	450	16680	15	23	21	21	19	20
le_450_15d	450	16750	15	24	21	21	20	20.1
le_450_25a	450	8260	25	25	25	25	25	25
le_450_25b	450	8263	25	25	25	25	25	25
le_450_25c	450	17343	25	29	26	26.8	26	26
le_450_25d	450	17425	25	28	27	27	26	26

DSATUR: best result out of ten runs;
pTS: Tabucol, *1-exchange* neighborhood, tabu length $\lfloor 10 + 0.6f(x) \rfloor$;
pAIS: $N = 20$, $N_c = 30$, $r = 0.5$

The last four columns of the table present the minimum and the average number of colors in the best found solution within 1800 seconds.

5 Conclusions

Clonal selection is an evolutionary approach that does not require a crossover operator. Designign a good crossover operator that preserves the good features of

the parents can prove to be difficult for some problems. The algorithm presented in this paper shows that clonal selection can be successfully applied to the Graph Coloring Problem.

The experiments have shown that for large graph coloring instances the depth of the search (stimulated by low pool size and low number of clones) is more important than its breadth when given a realistic time limit. This may not be the case when facing other combinatorial optimization problems, therefore further study on clonal selection algorithms is needed.

The results have also shown that the use of island-model parallelism increases the efficiency of the algorithm, however the effect of the migration mechanism should be investigated further.

Acknowledgments. The experiments were performed on a high performance cluster *holk* at the TASK Academic Computer Centre[3] in Gdańsk.

References

1. Brélaz, D.: New methods to color the vertices of a graph. Communications of the ACM 22, 251–256 (1979)
2. Cutello, V., Nicosia, G., Pavone, M.: A Hybrid Immune Algorithm with Information Gain for the Graph Coloring Problem. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 199–210. Springer, Heidelberg (2003)
3. de Castro, L.N., Von Zuben, F.J.: Artificial Immune Systems: Part I – Basic theory and applications, Technical Report – RT DCA 01/99, School of Computing and Electrical Engineering, State University of Campinas, Brasil (1999)
4. de Castro, L.N., Von Zuben, F.J.: Learning and Optimization Using the Clonal Selection Principle. IEEE Transaction on Evolutionary Computation 6(3) (2002)
5. de Castro, L.N., Timmis, J.I.: Artificial immune systems as a novel soft computing paradigm. Soft Computing 7, 526–544 (2003)
6. Dasgupta, D., Forrest, S.: Artificial Immune Systems in Industrial Applications. In: Proc. of the Second International Conference on Intelligent Processing and Manufacturing of Materials, vol. 1, pp. 257–267 (1999)
7. Hertz, A., de Werra, D.: Using Tabu Search Techniques for Graph Coloring. Computing 39, 345–351 (1987)
8. Kim, J., Bentley, P.: The Artificial Immune Model for Network Intrusion Detection. In: Proc. of the 7th European Conference on Intelligent Techniques and Soft Computing EUFIT (1999)
9. Nossal, G.J.V.: Life, Death and the Immune System. Scientific American Special Issue on the Immune System, 53–63 (September 1993)
10. Ong, Z.X., Tay, J.C., Kwoh, C.K.: Applying the Clonal Selection Principle to Find Flexible Job-Shop Schedules. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 442–455. Springer, Heidelberg (2005)

[3] http://www.task.gda.pl/

The Quantum Complexity of Group Testing*

Sebastian Dörn[1] and Thomas Thierauf[2]

[1] Inst. für Theoretische Informatik, Universität Ulm
Sebastian.Doern@uni-ulm.de
[2] Fak. Elektronik und Informatik, HTW Aalen
Thomas.Thierauf@HTW-Aalen.de

Abstract. We present quantum query and time complexity bounds for group testing problems. For a set S and a binary operation on S, we consider the decision problem whether a groupoid, semigroup or quasigroup is a group. Our quantum algorithms for these problems improve the best known classical complexity bounds. We also present upper and lower bounds for testing associativity, distributivity and commutativity.

1 Introduction

Quantum algorithms have the potential to demonstrate that for some problems quantum computation is more efficient than classical computation. A goal of quantum computing is to determine for which problems quantum computers are faster than classical computers. The most important known basic quantum algorithms are Shor's and Grover's algorithm. The first one is Shor's [Sho94] polynomial time quantum algorithm for the factorization of integers. The second one is Grover's search algorithm [Gro96]. Since then, we have seen some generalisations and applications of these two basic quantum techniques. The Shor algorithm has been generalized to a quantum algorithms for the hidden subgroup problem (see e.g. [CEMM98]). Grover's search algorithm can be used for quantum amplitude amplification [BHMT02] and quantum random walk search [Amb04, Sze04, MNRS07]. The application of these quantum search tools is a fast growing area in quantum computing. For example, quantum algorithms have been presented for several problems from computer science (see e.g. [BHT98, BDHHMSW01, Amb04]), graph theory (see e.g. [DHHM04, MSS05, AS06, Doe07a, Doe07b]) and (linear) algebra (see e.g. [MN05, BS06, DT07]).

In this paper we study the quantum complexity of group testing problems. For a set S and a binary operation on S, we consider the decision problem whether a groupoid, semigroup or quasigroup is a group. We also present upper and lower bounds for testing associativity, distributivity and commutativity. In particular, we improve the quantum query complexity for testing if a operation table is associative or a quasigroup from [DT07].

The motivation for studying the query complexity of algebraic problems is twofold. On the one hand side, these are fundamental and basic problems which

* Supported by DFG grants Scho 302/7-2.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 506–518, 2008.

have many applications in computer science. For example, testing if a black box is a group is very useful in cryptography. On the other hand, we can analyze how powerful are our tools for the construction of lower and upper bounds for the quantum query complexity of these problem. For many problems we can find optimal quantum algorithms by a combination of Grover search, amplitude amplification and quantum walk search. But for some problems this doesn't seem to work. Maybe this can be a motivation for the development of new quantum techniques.

In this paper our input is a operation table for a set S of size $n \times n$. In Section 3 we consider several group problems. Given a groupoid, semigroup or quasigroup S by its operation table, we have to decide whether S is a group. We present lower and upper bounds for the quantum query complexity of these group problems. In particular, we give nearly optimal quantum query algorithms for testing whether a groupoid or quasigroup is a group.

In Section 4 we present several bounds for testing associativity, distributivity and commutativity. For associativity testing we consider the binary operation $\circ : S \times S \to S'$, where $S' \subseteq S$. Dörn and Thierauf [DT07] constructed a quantum query algorithm which is faster than the trivial Grover search over all triples of S for $|S'| < n^{3/8}$. Here we improve the quantum query complexity of their algorithm, such that the algorithm is faster than the Grover search for $|S'| < n^{3/4}$. Moreover we determine the precise quantum query complexity for deciding whether a groupoid, semigroup and monoid is commutative.

2 Preliminaries

2.1 Quantum Query Model

In the query model, the input $x_1, \ldots, x_N$ is contained in a black box or oracle and can be accessed by queries to the black box. As a query we give i as input to the black box and the black box outputs x_i. The goal is to compute a Boolean function $f : \{0,1\}^N \to \{0,1\}$ on the input bits $x = (x_1, \ldots, x_N)$ minimizing the number of queries. The classical version of this model is known as decision tree.

The quantum query model was explicitly introduced by Beals et al. [BBCMW01]. In this model we pay for accessing the oracle, but unlike the classical case, we use the power of quantum parallelism to make queries in superposition. The state of the computation is represented by $|i, b, z\rangle$, where i is the query register, b is the answer register, and z is the working register. A quantum computation with T queries is a sequence of unitary transformations

$$U_0 \to O_x \to U_1 \to O_x \to \ldots \to U_{T-1} \to O_x \to U_T,$$

where each U_j is a unitary transformation that does not depend on the input x, and O_x are query (oracle) transformations. The oracle transformation O_x can be defined as $O_x : |i, b, z\rangle \to |i, b \oplus x_i, z\rangle$. The computations consists of the following three steps:

1. Go into the initial state $|0\rangle$.
2. Apply the transformation $U_T O_x \cdots O_x U_0$.
3. Measure the final state.

The result of the computation is the rightmost bit of the state obtained by the measurement.

The quantum computation determines f with bounded error, if for every x, the probability that the result of the computation equals $f(x_1, \ldots, x_N)$ is at least $1 - \epsilon$, for some fixed $\epsilon < 1/2$. In the query model of computation each query adds one to the query complexity of an algorithm, but all other computations are free. The time complexity of the algorithm is usually measured in terms of the total circuit size for the unitary operations U_i.

2.2 Tools for Quantum Algorithms

Here, we give three tools for the construction of our quantum algorithms.

Quantum Search. A search problem is a subset $S \subseteq [N]$ of the search space $[N]$. With S we associate its characteristic function $f_S : [N] \to \{0, 1\}$ with $f_S(x) = 1$ if $x \in S$, and 0 otherwise. Any $x \in S$ is called a solution to the search problem. Let $k = |S|$ be the number of solutions of S. It is a well known fact in quantum computing (see [Gro96, BBHT98]), that for $k > 0$, the expected quantum query complexity for finding one solution of S is $O(\sqrt{N/k})$, and for finding all solutions, it is $O(\sqrt{kN})$. Furthermore, whether $k > 0$ can be decided in $O(\sqrt{N})$ quantum queries to f_S. The running time complexity of Grover search is larger than its query complexity by a logarithmic factor.

Amplitude Amplification. The quantum amplitude amplification is a generalization of Grover's search algorithm. Let $\mathcal{A}$ be an algorithm for a problem with small success probability at least ϵ. Classically, we need $\Theta(1/\epsilon)$ repetitions of $\mathcal{A}$ to increase its success probability from ϵ to a constant, for example 2/3. There is a corresponding technique in the quantum case (see [BHMT02]). Let $\mathcal{A}$ be a quantum algorithm with one-sided error and success probability at least ϵ. Then there is a quantum algorithm $\mathcal{B}$ that solves $\mathcal{A}$ with success probability 2/3 by $O(\frac{1}{\sqrt{\epsilon}})$ invocations of $\mathcal{A}$.

Quantum Walk. Quantum walks are the quantum counterpart of Markov chains and random walks. Let $P = (p_{xy})$ be the transition matrix of an ergodic symmetric Markov chain on the state space X. Let $M \subseteq X$ be a set of marked states. Assume that the search algorithms use a data structure D that associates some data $D(x)$ with every state $x \in X$. From $D(x)$, we would like to determine if $x \in M$. When operating on D, we consider the following three types of costs:

- *Setup cost* s: The worst case cost to compute $D(x)$, for $x \in X$.
- *Update cost* u: The worst case cost for transition from x to y, and update $D(x)$ to $D(y)$.
- *Checking cost* c: The worst case cost for checking if $x \in M$ by using $D(x)$.

Theorem 1. [MNRS07] *Let $\delta > 0$ be the eigenvalue gap of an ergodic Markov chain P and let $\frac{|M|}{|X|} \geq \epsilon$. Then there is a quantum algorithm that determines if M is empty or finds an element of M with cost*

$$s + \frac{1}{\sqrt{\epsilon}} \left(\frac{1}{\sqrt{\delta}} u + c \right).$$

In the most practical applications (see [Amb04, MSS05]) the quantum walk takes place on the Johnson graph $J(n, r)$, which is defined as follows: the vertices are subsets of $\{1, \ldots, n\}$ of size r and two vertices are connected iff they differ in exactly one number. It is well known, that the spectral gap δ of $J(n, r)$ is $\Theta(1/r)$ for $1 \leq r \leq \frac{n}{2}$.

We apply the quantum walk on the graph categorical product of two Johnson graphs. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs, the *graph categorical product* $G = (V, E) = G_1 \times G_2$ of G_1, G_2 is defined as follows: $V = V_1 \times V_2$, and $((g_1, g_2), (g_1', g_2')) \in E$ iff $(g_1, g_1') \in E_1$ and $(g_2, g_2') \in E_2$.

2.3 Tool for Quantum Query Lower Bounds

In this paper, we use the following special case of a method by Ambainis [Amb02] to prove lower bounds for the quantum query complexity.

Theorem 2. [Amb02] *Let $\mathcal{F} = \{f : [n] \times [n] \to [n]\}$ be the set of all possible input function, and $\Phi : \mathcal{F} \to \{0, 1\}$. Let $A, B \subset \mathcal{F}$ such that $\Phi(f) = 1$ and $\Phi(g) = 0$ for all $f \in A$ and $g \in B$. Let $R \subset A \times B$, and m, m', l, l' be numbers such that*

1. *for every $f \in A$, there are at least m different $g \in B$ such that $(f, g) \in R$.*
2. *for every $g \in B$, there are at least m' different $f \in A$ such that $(f, g) \in R$.*
3. *for every $f \in A$ and $x, y \in [n]$, there are at most l different $g \in B$ such that $(f, g) \in R$ and $f(x, y) \neq g(x, y)$.*
4. *for every $g \in B$ and $x, y \in [n]$, there are at most l' different $f \in A$ such that $(f, g) \in R$ and $f(x, y) \neq g(x, y)$.*

Then every bounded-error quantum algorithm that computes Φ has quantum query complexity $\Omega\left(\sqrt{\frac{m \cdot m'}{l \cdot l'}}\right)$.

Let $f, g : [n] \times [n] \to [n]$, we define $d(f, g) = |\{\, x, y \in [n] \mid f(x, y) \neq g(x, y) \,\}|$. In some cases, we consider the special case $A, B \subset \{0, 1\}^{n \times n}$ and $(f, g) \in R$ if and only if f and g differ in exactly one position. Then it is $l = l' = 1$, and every bounded-error quantum algorithm that computes f has quantum query complexity of $\Omega\left(\sqrt{m \cdot m'}\right)$.

3 Group Problems

In this section we consider the decision problems whether a groupoid, semigroup or quasigroup S of size n with a binary operation $\circ$ is in fact a group. A *groupoid*

is a finite set S with a binary operation $\circ$ represented as operation table. The groupoid is called a *semigroup*, if it is associative. A *monoid* is a semigroup with an identity element. A *quasigroup* is a groupoid, where all equations $a \circ x = b$ and $x \circ a = b$ have unique solutions, and a *loop* is a quasigroup with an identity element.

3.1 Group Testing for Groupoids

We consider the problem whether a groupoid $(S, \circ)$ is in fact a group. There is a $O(n^2 \log n)$ deterministic algorithm for this problem by [RS00]. We develop a quantum algorithm that has time complexity $O(n^{\frac{13}{12}} \log^2 n)$. Furthermore, we present an $O(n \log n)$ query algorithm for this problem, that has time complexity $O(n^{3/2} \log n)$ however. The latter algorithm is nearly optimal with respect to the query complexity, as we prove a linear lower bound for this problem.

We need a generalization of a lemma from [RS00].

Definition 1. *Let $(S, \circ)$ be a groupoid represented by its operation table T. A row of T is called* cancellative*, if it is a permutation of S.*

Lemma 1. [RS00] *Let $\circ$ be cancellative in r rows. If $\circ$ is nonassociative then it has at least $r/4$ nonassociative triples.*

Theorem 3. *Whether a groupoid is a group can be decided by a quantum algorithm within $O(n^{\frac{13}{12}} \log^c n)$ expected steps, for some constant c.*

Proof. Let $(S, \circ)$ be a groupoid represented by its operation table T. Note that if S is a group, then every row of A is cancellative. Our first step is to determine whether the operation is associative. To do so, we choose an arbitrary subset A of S of size r. We determine r later. Then we check whether T is cancellative in the rows indexed by A. This is not the case, if we find a row with two equal elements. Hence we can solve this with a Grover search and the element distinctness quantum algorithm by Ambainis [Amb04]. The quantum query complexity of this procedure is $O(\sqrt{r} n^{\frac{2}{3}})$.

If any of the considered rows in not cancellative then we are done. Otherwise we randomly choose three elements $a, b, c \in S$ and check whether $(a \circ b) \circ c \neq a \circ (b \circ c)$. If the operation is not associative, then the probability of finding a nonassociative triple is at least $\frac{r}{4n^3}$ by Lemma 1. By using the quantum amplitude amplification we have an $O(n^{\frac{3}{2}}/\sqrt{r})$ quantum query algorithm for finding a nonassociative triple.

If there are no nonassociative triple, then $(S, \circ)$ is a semigroup. Whether this semigroup is a group can be decided with $O(n^{\frac{11}{14}} \log n)$ quantum queries by Theorem 6. The expected quantum query complexity of the whole algorithm we get

$$O\left(\sqrt{r} n^{\frac{2}{3}} + \frac{n^{\frac{3}{2}}}{\sqrt{r}} + n^{\frac{11}{14}} \log n\right),$$

which is minimized for $r = n^{\frac{5}{6}}$. Hence the expected time complexity of this algorithm is $O(n^{\frac{13}{12}} \log^c n)$ for a constant c, since the element distinctness procedure has running time of $O(n^{2/3} \log^c n)$. □

We can further improve the query complexity of the problem if we allow a larger running time.

Theorem 4. *Whether a groupoid is a group can be decided with $O(n \log n)$ expected quantum queries.*

Proof. Let $(S, \circ)$ be a groupoid represented by its operation table T. A well known fact from algebra is, that if $(S, \circ)$ is a quasigroup, then a random subset $R \subset S$ with $c \log n$ elements is a set of generators with probability at least $1 - \exp(c)$ (see [RS00]). We choose a random subset R of $O(\log n)$ elements of S. Then we check whether R is a generating set of $(S, \circ)$. To do so, let $S_0 = R$. We compute inductively $S_i = S_{i-1} \cup (R \circ S_{i-1})$. This adds at least one element in a step, until we reach some $k \leq n$ such that $S_k = S$. In this case, R is a set of generators. For each element a added to some set S_i, we query the $\log n$ elements $R \circ a$ to look for further elements. In total we query at most the $O(n \log n)$ elements of the $R \times S$ submatrix of T. The quantum time is bounded by $O(n^{3/2} \log n)$.

If R is a set of generators, we have to verify whether the multiplication table is associative. Light observed (see [CP61]) that if R is a set of generators of S, then it suffices to test all triples a, b, c in which b is an element of R. By using Grover search, the quantum query for finding a nonassociative triple (if there is one) is $O(n\sqrt{\log n})$. By Theorem 6 we can decide whether this semigroup is a group. The total quantum query complexity of is $O(n \log n)$. □

The upper bound of Theorem 4 almost matches the lower bound we have.

Theorem 5. *Whether a groupoid is a group requires $\Omega(n)$ quantum queries.*

Proof. We apply the Theorem 2. Let A be the operation table T of $\mathbb{Z}_n$ and let $\circ$ be the addition modular n. Then T is a group. The set B consists of all $n \times n$ matrices T', where one entry of T' is modified. Therefore the tables of B forming no groups. The relation R is defined by $R = \{ (T, T') \in (A, B) \mid d(T, T') = 1 \}$. Then R satisfies that $m = n^2(n-1)$, $m' = 1$, $l = n - 1$ and $l' = 1$. Therefore the quantum query complexity is $\Omega(n)$. □

3.2 Group Testing for Semigroups and Quasigroups

Dörn and Thierauf [DT07] considered the problem whether a finite monoid $(S, \circ)$ is in fact a group. They showed that the problem can be solved with $O(n^{\frac{3}{2}})$ queries by a (classical) randomized algorithm, and with $O(n^{\frac{11}{14}} \log n)$ expected queries by a quantum algorithm.

Now suppose that the input is only known to be a semigroup and we want to decide whether it is in fact a group. To do so, we first search for an identity element and then use the algorithm from [DT07]. To find the identity element,

we start by choosing an element a of S and search for an element $e \in S$ such that $a \circ e = a$. Then e is our candidate for the identity element. Recall that we finally want to decide whether S is a group. In this case, the identity element is unique. Hence if our candidate e doesn't work we can safely reject the input, even in the case that S actually has an identity element. To test our candidate e, it suffices to check whether $b \circ e = b$ for all $b \in S$. Obviously the two steps can be done in $O(n)$ queries classically and $O(\sqrt{n})$ quantum queries with Grover search. We summarize the observation:

Theorem 6. *Whether a given semigroup is a group can be decided with*

1. $O(n^{\frac{3}{2}})$ *queries by a randomized algorithm.*
2. $O(n^{\frac{11}{14}} \log n)$ *by a quantum query algorithm.*

The result should be contrasted with the following: if we want to decide whether a given semigroup is in fact a monoid, then the best known algorithms make $O(n^2)$ queries classically and $O(n)$ queries in the quantum setting.

Next we assume that the input $(S, \circ)$ is a quasigroup. That is, every row in the operation table is cancellative. To check associativity, we can apply Lemma 1 with $r = n$ and the test from the proof of Theorem 3. This yields an $O(n)$ quantum query algorithm to test whether the operation is associative, and hence a group. We show that this bound is tight up to constant factor.

Theorem 7. *Whether a given quasigroup or a loop is a group can be decided with quantum query complexity* $\Theta(n)$.

Proof. For the lower bound, we apply Theorem 2 in connection with an idea of [RS00] for proving an $\Omega(n^2)$ lower bound for this problem in classical computing. The set A consists of the operation table T of the group $(\mathbb{Z}_2^m, +)$, where $+$ is the vector addition modulo 2. Let $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{Z}_2^m$ with $\boldsymbol{a} \neq \boldsymbol{0}$. The set B consists of all operation tables of $(\mathbb{Z}_2^m, \circ)$, where $\circ$ is equal to $+$ except in the following four positions:

1. $\boldsymbol{b} \circ \boldsymbol{c} = \boldsymbol{b} + (\boldsymbol{a} + \boldsymbol{c})$,
2. $\boldsymbol{b} \circ (\boldsymbol{a} + \boldsymbol{c}) = \boldsymbol{b} + \boldsymbol{c}$,
3. $(\boldsymbol{a} + \boldsymbol{b}) \circ \boldsymbol{c} = \boldsymbol{b} + \boldsymbol{c}$,
4. $(\boldsymbol{a} + \boldsymbol{b}) \circ (\boldsymbol{a} + \boldsymbol{c}) = \boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c}$.

All tables of B are quasigroups because the above modifications simply exchange two elements in two rows of the table T, but they are not associative, since

$$\boldsymbol{a} + \boldsymbol{b} = (\boldsymbol{c} \circ (\boldsymbol{a} + \boldsymbol{b})) \circ \boldsymbol{c} \neq \boldsymbol{c} \circ ((\boldsymbol{a} + \boldsymbol{b}) \circ \boldsymbol{c}) = \boldsymbol{b}.$$

The relation R is defined by

$$R = \{\, (T, T') \in (A, B) \mid T' \text{ originates of the above four modifications of } T \,\}.$$

Then R satisfies $m = \Omega(n^3)$, $m' = 1$, $l = \Omega(n)$ and $l' = 1$. □

4 Testing Associativity, Distributivity and Commutativity

4.1 The Semigroup Problem

We consider the following semigroup problem. We have given two sets S and $S' \subseteq S$ and a binary operation $\circ : S \times S \to S'$ represented by a table. We denote with n the size of the set S. One has to decide whether S is a semigroup, that is, whether the operation on S is associative.

The complexity of this problem was first considered by Rajagopalan and Schulman [RS00], who gave a randomized algorithm with time complexity of $O(n^2 \log \frac{1}{\delta})$, where δ is the error probability. They also showed a lower bound of $\Omega(n^2)$. The previously best known algorithm was the naive $\Omega(n^3)$-algorithm that checks all triples.

In the quantum setting, one can do a Grover search over all triples $(a, b, c) \in S^3$ and check whether the triple is associative. The quantum query complexity of the search is $O(n^{3/2})$. Dörn and Thierauf [DT07] constructed a quantum query algorithm which is faster than the Grover search for $|S'| < n^{3/8}$. They also proved a quantum query lower bound of $\Omega(n)$. Here we improve the quantum query complexity of their algorithm by a more detailed analysis. Furthermore our algorithm is faster than the Grover search for $|S'| < n^{3/4}$.

Theorem 8. *Let $k = n^\alpha$ be the size of S' with $0 < \alpha \leq 1$. The quantum query complexity of the semigroup problem is*

$$\begin{cases} O(n^{\frac{5+\alpha}{4}}), & \text{for } 0 < \alpha \leq \frac{1}{3}, \\ O(n^{\frac{6+2\alpha}{5}}), & \text{for } \frac{1}{3} < \alpha \leq \frac{3}{4}, \\ O(n^{\frac{3}{2}}), & \text{for } \frac{3}{4} < \alpha \leq 1. \end{cases}$$

Proof. We use the quantum walk search scheme of Theorem 1. The quantum walk is done on the categorical graph product G_J of two Johnson graphs $J(n, r)$. Let A and B two subsets of S of size r. We will determine r later. We search for a pair $(a, b) \in S^2$, such that a, b are two elements of a nonassociative triple. Then the marked vertices of G_J correspond to pairs (A, B) with $(A \circ B) \circ S \neq A \circ (B \circ S)$. In every step of the walk, we exchange one row and one column of A and B. The database of our quantum walk is the set

$$D(A, B) = \{ (a, b, a \circ b) \mid a \in A \cup S' \text{ and } b \in B \cup S' \}.$$

Now we compute the quantum query costs for the setup, update and checking. The setup cost for the database $D(A, B)$ is $O((r + k)^2)$ and the update cost is $O(r+k)$. To check whether a pair (A, B) is marked, we have to test if $(A \circ B) \circ S \neq A \circ (B \circ S)$.

Now we claim, that the quantum query cost to check this inequality is $O(\sqrt{nrk})$. Therefore we search for a pair $(b, c) \in B \times S$ with $(A \circ b) \circ c \neq A \circ (b \circ c)$. The computation of $A \circ (b \circ c)$ requires only one query, by using our database, since $(b \circ c) \in S'$. The result is a vector of size r, which we denote by $(y_1, \ldots, y_r)$.

The evaluation of $(A \circ b)$ needs no queries by using our database, let $(c_1, \ldots, c_r)$ be the result. This vector consists of at most k different entries $(x_1, \ldots, x_k)$. Now we use Grover's algorithm for searching an $i \in [k]$, such that $x_i \circ c \neq y_j$ for an $j \in [r]$ with $x_i = c_j$. This search can be done in $O(\sqrt{k})$ quantum queries. Therefore, by applying two Grover search subroutines, the checking cost is $O(\sqrt{nrk})$.

The spectral gap of the walk on G_J is $\delta = O(1/r)$ for $1 \leq r \leq \frac{n}{2}$, see [BS06]. If there is a triple (a, b, c) with $(a \circ b) \circ c \neq a \circ (b \circ c)$, then there are at least $\binom{n-1}{r-1}^2$ marked sets (A, B). Therefore we have $\epsilon \geq r^2/n^2$.

Let $r = n^\beta$ for $0 < \beta < 1$. Assuming $r > k$, then the quantum query complexity of the semigroup problem is

$$O\left(r^2 + \frac{n}{r}\left(\sqrt{r} \cdot r + \sqrt{nrk}\right)\right) = O\left(n^{2\beta} + n^{1+\frac{\beta}{2}} + n^{\frac{3+\alpha-\beta}{2}}\right).$$

Now we choose β depending on α such that this expression is minimal. Suppose that $2\beta \leq 1 + \frac{\beta}{2}$, i.e. $\beta \leq \frac{2}{3}$. From the equation $1 + \frac{\beta}{2} = \frac{3+\alpha-\beta}{2}$, we get $\beta = \frac{1+\alpha}{2}$. Then the quantum query complexity of the semigroup problem is $O(n^{\frac{5+\alpha}{4}})$ for $r = n^{\frac{1+\alpha}{2}}$ and $\alpha \leq \frac{1}{3}$. Otherwise if $2\beta > 1 + \frac{\beta}{2}$, i.e. $\beta > \frac{2}{3}$, we get $\beta = \frac{3+\alpha}{5}$ from the equation $2\beta = \frac{3+\alpha-\beta}{2}$. Then the quantum query complexity is $O(n^{\frac{6+2\alpha}{5}})$ for $r = n^{\frac{3+\alpha}{5}}$ and $\alpha > \frac{3}{4}$. If $\alpha > \frac{3}{4}$, the query complexity is bigger than $O(n^{\frac{3}{2}})$, therefore we use Grover search instead of quantum walk search. □

Note that the time complexity of our algorithm is $O(n^{1.5} \log n)$.

4.2 The Distributivity Problem

In the distributivity problem we are given a set S and two binary operations $\oplus : S \times S \to S$ and $\otimes : S \times S \to S$ represented by tables. One has to decide whether $(S, \oplus, \otimes)$ is distributive, i.e. we have to test whether the two equation $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ and $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ are satisfied. A triple $(a, b, c) \in S^3$ that fulfills both equations is called a *distributive triple*. In classical computing, it is not known whether this problem can be solved in less than cubic time. In the quantum setting, one can do a Grover search over all triples $(a, b, c) \in S^3$ and check whether each triple is distributive. The quantum query complexity of the search is $O(n^{3/2})$. We show a linear lower bound on the query complexity.

Theorem 9. *The distributivity problem requires $\Omega(n)$ quantum queries.*

Proof. Let $S = \{0, 1, \ldots, n-1\}$. We apply the Theorem 2. The set A consists of all pairs of $n \times n$ matrices $T_\oplus$ and $T_\otimes$, where $T_\otimes$ is the zero-matrix, and the entry at position $(1, 0)$ in $T_\oplus$ is 1, and 0 otherwise. It is easy to see, that the tables of A are distributive, since $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z) = 0$ for all $x, y, z \in S$. The set B consists of all pairs of $n \times n$ matrices $T'_\oplus$ and $T'_\otimes$, where the entry of position $(1, 0)$ in $T'_\oplus$, and (a, b) in $T'_\otimes$ is 1, for $a, b \in S - \{0, 1\}$, and 0 otherwise. Then $a \otimes (b \oplus c) = 0$ and $(a \otimes b) \oplus (a \otimes c) = 1$ with $b \neq c$. Therefore the tables of B are not distributive.

From each $(T_\oplus, T_\otimes) \in A$, we can obtain $(T'_\oplus, T'_\otimes) \in B$ by replacing the entry 0 of $T_\otimes$ at (a, b) by 1, for any $a, b \notin \{0, 1\}$. Hence we have $m = \Omega(n^2)$. From each $(T'_\oplus, T'_\otimes) \in B$, we can obtain $(T_\oplus, T_\otimes) \in A$, by replacing the entry 1 of $T_\otimes$ at position (a, b) by 0, for $a, b \notin \{0, 1\}$. Thus we have $m' = 1$. By Theorem 2, the quantum query complexity is $\Omega(\sqrt{m \cdot m'}) = \Omega(n)$. □

If $(S, \oplus)$ is a commutative quasigroup, then we can get a faster algorithm to check distributivity. The key is that one nondistributive triple implies the existence of more such triples. Similar to Lemma 1, we have the following lemma.

Lemma 2. *Let S be a set and $\oplus, \otimes$ be two binary operations on S, such that $(S, \oplus)$ is a commutative quasigroup. If $(S, \oplus, \otimes)$ is nondistributive, then it has at least $\Omega(n)$ nondistributive triples.*

Proof. Let (a, b, c) be a nondistributive triple. Let $a = a' \oplus a''$ and consider the following cycle.

$$\begin{aligned}
(a' \oplus a'') \otimes (b \oplus c) &= ((a' \oplus a'') \otimes b) \oplus ((a' \oplus a'') \otimes c) \\
&= ((a' \otimes b) \oplus (a'' \otimes b)) \oplus ((a' \oplus a'') \otimes c) \\
&= (a' \otimes b) \oplus (a'' \otimes b) \oplus (a' \otimes c) \oplus (a'' \otimes c) \\
&= (a' \otimes b) \oplus (a' \otimes c) \oplus (a'' \otimes (b \oplus c)) \\
&= (a' \otimes (b \oplus c)) \oplus (a'' \otimes (b \oplus c)) \\
&= (a' \oplus a'') \otimes (b \oplus c).
\end{aligned}$$

Suppose that $a \otimes (b \oplus c) \neq (a \otimes b) \oplus (a \otimes c)$. Then at least one of the above equations does not hold. Therefore at least one of the following triples must be nondistributive:

$$(a', a'', b), \quad (a', a'', c), \quad (a'', b, c), \quad (a', b, c), \quad (a', a'', b \oplus c).$$

Since $(S, \oplus)$ is a quasigroup, a can be written as $a' \oplus a''$ in n different ways. For each of these, distributivity fails in at least one of the five categories from above. Therefore there exists a category for which there are $\geq n/5$ failures.

The case that $(a \oplus b) \otimes c \neq (a \otimes c) \oplus (b \otimes c)$ can be handled similarly □

By using Lemma 2 in combination with the amplitude amplification (similar to Theorem 3) we have

Theorem 10. *Let $(S, \oplus)$ be a commutative quasigroup and $(S, \otimes)$ a groupoid. Whether $(S, \oplus, \otimes)$ is distributive can be decided with quantum query complexity of $O(n)$.*

4.3 The Commutativity Problem

In the commutativity problem we have given a finite set S of size n with a binary operation $\circ : S \times S \to S$ represented by a table. One has to decide whether S is a commutative. In the quantum setting, one can solve the problem in linear time by a Grover search over all tuple $(a, b) \in S^2$ that checks whether the tuple is commutative. We show that the commutativity problem requires $\Omega(n)$ quantum queries, even when S is a monoid.

Theorem 11. *The quantum query complexity of the commutativity problem for groupoids, semigroups, and monoids is* $\Theta(n)$.

Proof. We start by showing the lower bound for semigroups via Theorem 2. Let $S = \{0, 1, \ldots, n-1\}$. The set A consists of the zero matrix of order n. The set B consists of all $n \times n$ matrices, where the entry of position (a, b) is 1, for $a \neq b \in S - \{0, 1\}$, and 0 otherwise. All operation tables of the sets A and B are semigroups. Then we have $m = \Omega(n^2)$, $m' = 1$, and the quantum query lower bound for testing if a given semigroup is commutative is $\Omega(n)$.

We reduce the commutativity problem for semigroups to the commutativity problem for monoids. Let S be a semigroup represented as a operation table. We define a monoid $M = S \cup \{e\}$ with the identity element $e \notin S$, that is, with $a \circ e = e \circ a = a$, for all $a \in S$. Then the semigroup S is commutative iff the monoid M is commutative. □

Magniez and Nayak [MN05] quantize a classical Markov chain for testing the commutativity of a black box group given by the generators. The constructed an $O(k^{2/3} \log k)$ quantum query algorithm, where k is the number of generators of the group. In the case when $(S, \circ)$ is a quasigroup, a random set of $c \log n$ elements will be a set of generators with probability at least $1 - \exp(c)$ [RS00]. Therefore we obtain the following result:

Theorem 12. *Whether a quasigroup, loop or group is commutative can be decided with quantum query complexity* $O((\log n)^{\frac{2}{3}} \log\log n)$.

Conclusions

The table below summarizes the quantum query complexity (**QQC**) and the quantum time complexity (**QTC**) of the algebraic problems considered in the paper. Some of these results are proved in [DT07]. It remains open to close the gaps between the upper and the lower bounds where they don't match.

Problem	**Description**	**QQC**	**QTC**
Semigroup	Decide if $S \times S \to S'$ is a semigroup for constant size of S'.	$\Omega(n)$ $O(n^{\frac{5}{4}})$	$O(n^{\frac{3}{2}} \log n)$
Identity	Decide if a groupoid has an identity element.	$\Theta(n)$	$O(n \log n)$
Quasigroup	Decide if a groupoid is a quasigroup.	$\Omega(n)$ $O(n^{\frac{7}{6}})$	$O(n^{\frac{7}{6}} \log n)$
Group I	Decide if a groupoid is a group.	$\Omega(n)$ $O(n \log n)$	$O(n^{\frac{13}{12}} \log^c n)$
Group II	Decide if a semigroup is a group.	$O(n^{\frac{11}{14}} \log n)$	$O(n^{\frac{11}{14}} \log^c n)$
Group III	Decide if a quasigroup is a group.	$\Theta(n)$	$O(n \log n)$
Group Commut. I	Decide if a groupoid/ semigroup/monoid is commutative.	$\Theta(n)$	$O(n \log n)$
Group Commut. II	Decide if a quasigroup/group is commutative.	$\tilde{O}((\log n)^{\frac{2}{3}})$	$\tilde{O}((\log n)^{\frac{2}{3}})$

References

[Amb02] Ambainis, A.: Quantum Lower Bounds by Quantum Arguments. Journal of Computer and System Sciences 64, 750–767 (2002)

[Amb04] Ambainis, A.: Quantum walk algorithm for element distinctness. In: Proceedings of FOCS 2004, pp. 22–31 (2004)

[AS06] Ambainis, A., Špalek, R.: Quantum Algorithms for Matching and Network Flows. In: Proceedings of STACS 2006, pp. 172–183 (2006)

[BBCMW01] Beals, R., Buhrman, H., Cleve, R., Mosca, M., de Wolf, R.: Quantum lower bounds by polynomials. Journal of ACM 48, 778–797 (2001)

[BBHT98] Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik 46(4-5), 493–505 (1998)

[BDHHMSW01] Buhrman, H., Dürr, C., Heiligman, M., Høyer, P., Magniez, F., Santha, M., de Wolf, R.: Quantum Algorithms for Element Distinctness. In: Proceedings of CCC 2001, pp. 131–137 (2001)

[BHMT02] Brassard, G., Hóyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. AMS Contemporary Mathematics 305, 53–74 (2002)

[BHT98] Brassard, G., Hóyer, P., Tapp, A.: Quantum Cryptanalysis of Hash and Claw-Free Functions. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 163–169. Springer, Heidelberg (1998)

[BS06] Buhrman, H., Špalek, R.: Quantum Verification of Matrix Products. In: Proceedings of SODA 2006, pp. 880–889 (2006)

[CEMM98] Cleve, R., Ekert, A., Macchiavello, C., Mosca, M.: Quantum algorithms revisited. In: Proceedings of the Royal Society of London, Series A, pp. 339–354 (1998)

[CP61] Clifford, A.H., Preston, G.B.: The Algebraic Theory of Semigroups. American Mathematical Society (1961)

[Doe07a] Dörn, S.: Quantum Complexity Bounds of Independent Set Problems. In: Proceedings of SOFSEM 2007 (SRF), pp. 25–36 (2007)

[Doe07b] Dörn, S.: Quantum Algorithms for Graph Traversals and Related Problems. In: Proceedings of CIE 2007, pp. 123–131 (2007)

[DT07] Dörn, S., Thierauf, T.: The Quantum Query Complexity of Algebraic Properties. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) FCT 2007. LNCS, vol. 4639, pp. 250–260. Springer, Heidelberg (2007)

[DHHM04] Dürr, C., Heiligman, M., Høyer, P., Mhalla, M.: Quantum query complexity of some graph problems. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 481–493. Springer, Heidelberg (2004)

[Gro96] Grover, L.: A fast mechanical algorithm for database search. In: Proceedings of STOC 1996, pp. 212–219 (1996)

[MN05] Magniez, F., Nayak, A.: Quantum complexity of testing group commutativity. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1312–1324. Springer, Heidelberg (2005)

[MNRS07] Magniez, F., Nayak, A., Roland, J., Santha, M.: Search via Quantum Walk. In: Proceedings of STOC 2007, pp. 575–584 (2007)

[MSS05] Magniez, F., Santha, M., Szegedy, M.: Quantum Algorithms for the Triangle Problem. In: Proceedings of SODA 2005, pp. 1109–1117 (2005)

[RS00] Rajagopalan, S., Schulman, L.J.: Verification of identities. SIAM J. Computing 29(4), 1155–1163 (2000)

[Sho94] Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of FOCS 1994, pp. 124–134 (1994)

[Sze04] Szegedy, M.: Quantum speed-up of Markov chain based algorithms. In: Proceedings of FOCS 2004, pp. 32–41 (2004)

Quantum Walks: A Markovian Perspective

Diego de Falco[1,2] and Dario Tamascelli[1,2]

[1] Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39/41, 20135 Milano, Italy
[2] CIMAINA, Centro Interdipartimentale Materiali e Interfacce Nanostrutturati, Università degli Studi di Milano

Abstract. For a *continuous-time* quantum walk on a line the variance of the position observable grows quadratically in time, whereas, for its classical counterpart on the same graph, it exhibits a linear, diffusive, behaviour. A *quantum* walk, thus, propagates at a rate which is linear in time, as compared to the square root rate for a classical *random* walk. Indeed, it has been suggested that there are graphs that can be traversed by a *quantum* walker exponentially faster than by the classical *random* analogue. In this note we adopt the approach of exploring the conditions to impose on a Markov process in order to emulate its quantum counterpart: the central issue that emerges is the problem of taking into account, in the numerical generation of each sample path, the *causative* effect of the ensemble of trajectories to which it belongs. How to deal numerically with this problem is shown in a paradigmatic example.

Keywords: continuous-time quantum walks, birth-and-death processes, sample paths.

1 Paradigmatic Examples

The identity

$$\sum_{x=-\infty}^{+\infty} J_x(t)^2 = 1, \tag{1}$$

satisfied by the Bessel functions of first kind and integer order $J_x(t)$, shows that the function

$$\rho(t,x) = J_x(t)^2 \tag{2}$$

is, for each time t, a probability mass function on the relative integers.

We raise here the question of finding examples of *phenomena* of probabilistic time evolution described by this probability mass function.

We will give two *distinct*, apparently very different, answers to the above question: finding the relationship between the two distinct examples we are going to exhibit below and discussing the extent and generality of this relationship will be the main focus of this paper.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 519–530, 2008.

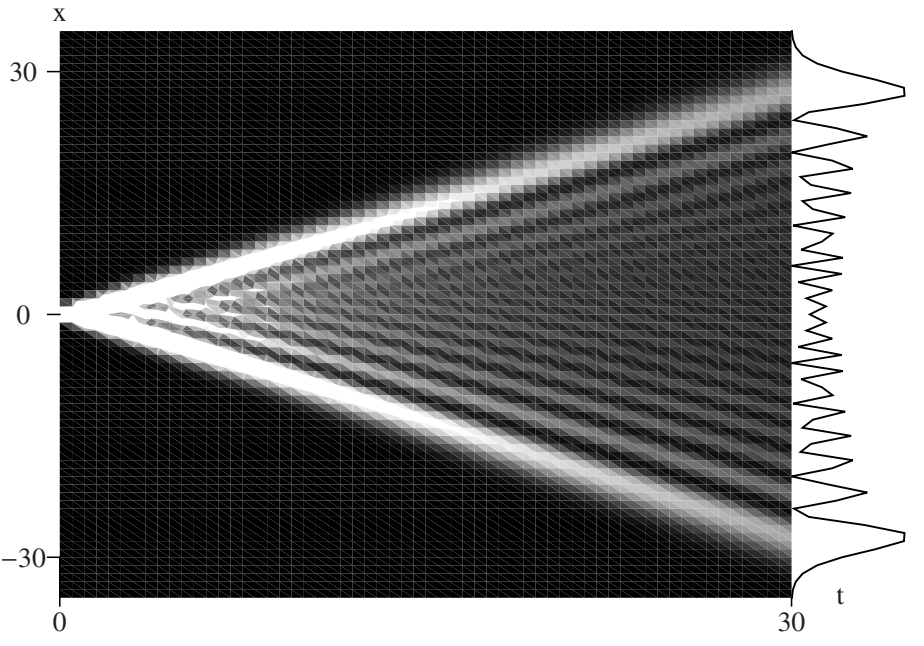


Fig. 1. A density plot of $\rho(t,x) = |\psi(t,x)|^2$. The profile of $\rho(30,x)$ as a function of x is shown on the right.

Example 1. The function

$$\psi(t,x) = i^x J_x(t) \tag{3}$$

is the solution of the Schrödinger equation on the relative integers

$$i\frac{d}{dt}\psi(t,x) = -\frac{1}{2}(\psi(t,x-1) + \psi(t,x+1)) \tag{4}$$

under the initial condition

$$\psi(0,x) = \delta_{0,x}. \tag{5}$$

Otherwise stated, the function $\rho(t,x) = J_x(t)^2$ is, at every time t, the probability distribution of a *continuous-time* quantum walk on the graph having the relative integers as vertices, with edges between nearest neighbour sites [1]. This quantum walk starts at time 0 from the origin.

Figure 1, a density plot of $\rho(t,x) = J_x(t)^2$, clearly shows the linear propagation expected in such a quantum walk. The reader more intersted in the *phenomenon* than in the equation (in this case eq. (4)) will appreciate recognizing in figure 1 the intensity pattern of propagation of light in a waveguide lattice [2].

Example 2. Consider a birth-and-death random process $q(t)$ on the relative integers, evolving according to the following rules:

i. *geometric mean rule*: for every edge $\{x, x+1\}$ and every time t the fraction of transitions per unit time taking place along this edge (number of transitions $x \to x+1$ *plus* number of transitions $x+1 \to x$ per unit time)/(sample size) is equal to the geometric mean of the probability of the process being in x and the probability of being in $x+1$;
ii. *local unidirectionality rule*: for every edge $\{x, x+1\}$, and depending on time t, only transitions $x \to x+1$ or only transitions $x+1 \to x$ are allowed;
iii. *"horror vacui" rule*: for every site x, if at a time t_x the probability of being in x passes through the value 0, then there is an interval of time following t_x in which along the edges $\{x-1, x\}$ and $\{x, x+1\}$ only transitions toward x are allowed; this time interval terminates as soon as the probability of being in one of the two neighbours of x crosses the value 0 (at which instant the *"horror vacui" rule* takes hold for such a neighbour).

As to the initial conditions, we suppose that there exists $\tau_0 > 0$ such that, for every integer x,

$$\rho(t, x) \equiv P(q(t) = x) > 0, \text{ for } 0 < t < \tau_0 \tag{6}$$

and

$$\lim_{t \to 0^+} \rho(t, x) = \delta_{0,x}. \tag{7}$$

Together with the above initial condition on the position of the process, we impose, as a condition on its initial "velocity", the requirement that in the time interval $[0, \tau_0)$ only transitions taking the process *away* from the origin are allowed.

We, finally, impose a left-right symmetry on the position of the process, in the form

$$\rho(t, x) = \rho(t, -x) \tag{8}$$

and a left-right symmetry on its "velocity" expressed in terms of its birth rate $\lambda(t, x)$ and its death rate $\mu(t, x)$ as

$$\lambda(t, x) = \mu(t, -x). \tag{9}$$

The transition probabilities per unit time $\lambda(t, x)$ and $\mu(t, x)$ are defined, respectively, by

$$p(t + \tau, x + 1; t, x) = \tau \cdot \lambda(t, x) + o(\tau) \tag{10}$$

$$p(t + \tau, x - 1; t, x) = \tau \cdot \mu(t, x) + o(\tau) \tag{11}$$

for $\tau \to 0^+$.

Here and elsewhere we indicate by $p(t, x; t_0, x_0)$ the conditional probability

$$P(q(t) = x | q(t_0) = x_0)$$

of finding the process at time t in x, given that at time t_0 it is in x_0.

Condition (i) can, now, be written as the equation

$$\lambda(t,x)\rho(t,x) + \mu(t,x+1)\rho(t,x+1) = \sqrt{\rho(t,x)\rho(t,x+1)}, \tag{12}$$

relating the three unknown fields λ, μ and ρ. The left hand side is, indeed, the probability per unit time of a transition along the link $\{x, x+1\}$. Notice that, because of (ii), equation (12) allows, locally, to express λ or μ as a function of the values of ρ at two neighbouring points.

A further equation involving the unknown fields is the continuity equation

$$\begin{aligned}\frac{d}{dt}\rho(t,x) = (\mu(t,x+1)\rho(t,x+1) - \lambda(t,x)\rho(t,x)) + \\ + (\lambda(t,x-1)\rho(t,x-1) - \mu(t,x)\rho(t,x)),\end{aligned} \tag{13}$$

expressing the fact that the probability mass at x increases because of transitions $x \pm 1 \to x$ and decreases because of transitions $x \to x \pm 1$.

In the time interval $[0, \tau_0)$, we can therefore write, using also the left-right symmetry and the initial condition of allowing only transitions taking the process away from the origin (namely, for $0 \le t < \tau_0$, $\lambda(t,x) > 0$ for $x \ge 0$ and $\mu(t,x) > 0$ for $x \le 0$),

$$\frac{d}{dt}\rho(t,0) = -2\sqrt{\rho(t,0)\rho(t,1)} \tag{14a}$$

$$\frac{d}{dt}\rho(t,x) = +\sqrt{\rho(t,x-1)\rho(t,x)} - \sqrt{\rho(t,x)\rho(t,x+1)}, \text{ for } x > 0. \tag{14b}$$

Equations (14) are satisfied by $\rho(t,x) = J_x(t)^2$, for values of t such that $J_x(t)$ is positive for every non negative integer x. This determines the numerical value of τ_0 to be the smallest positive solution of the equation $J_0(t) = 0$, namely

$$\tau_0 = 2.4048. \tag{15}$$

For a suitable value of $\tau_1 > \tau_0$ condition (iii) will allow, in the time interval $[\tau_0, \tau_1)$, for transitions $\pm 1 \to 0$, so that equations (14) are to be substituted, in this interval, by

$$\frac{d}{dt}\rho(t,0) = +2\sqrt{\rho(t,0)\rho(t,1)}, \tag{16a}$$

$$\frac{d}{dt}\rho(t,1) = -\sqrt{\rho(t,0)\rho(t,1)} - \sqrt{\rho(t,2)\rho(t,1)} \tag{16b}$$

$$\frac{d}{dt}\rho(t,x) = +\sqrt{\rho(t,x-1)\rho(t,x)} - \sqrt{\rho(t,x)\rho(t,x+1)}, \text{ for } x > 1. \tag{16c}$$

Equations (16) are again satisfied by $\rho(t,x) = J_x(t)^2$, but, this time, for values of t such that $J_0(t) < 0$ and $J_x(t)$ is positive for every positive integer x. This determines the numerical value of τ_1 to be the smallest positive root of $J_1(t) = 0$, namely

$$\tau_1 = 3.8317. \tag{17}$$

The above considerations can be iterated: using the fact that between two consecutive zeroes of $J_x(t)$ there is one and only one zero of $J_{x+1}(t)$, one can control the changes of sign determined by (iii) in the continuity equation, to the effect of proving that the process $q(t)$ described by the conditions posed above satisfies, for every t, the condition

$$\rho(t,x) \equiv P(q(t)=x) = J_x(t)^2. \tag{18}$$

Figure 2, to be compared with figure 1, shows a few sample paths of the process $q(t)$.

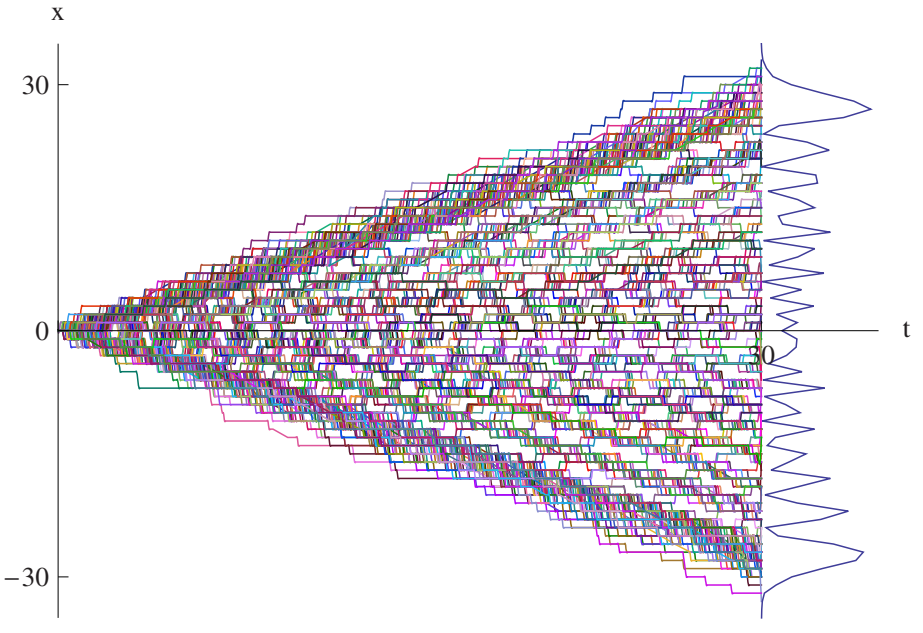


Fig. 2. A sample of 500 paths of the stochastic process of *Example 2*. For the purpose of comparison with figure 1, the *empirical distribution* at time $t = 30$, of a sample of $5 \cdot 10^4$ trajectories, is shown on the right.

The reader more interested in the *phenomenon* than in the *equations* (in this case the continuity equation (13) for the evolution of $\rho(t,x)$ and the forward Kolmogorov equation for the evolution of $p(t,x;t_0,x_0)$) will see, in section 3, how the numerical procedure leading to figure 2 actually makes use *only* of a step by step implementation of the dynamical rules (i), (ii), (iii).

2 Quantum Walks vs. Random Walks

In this section we look at quantum mechanics as a metaphor suggesting, at the heuristic level, an interesting dynamical behaviour for a random (Markov)

process exploring a graph or decision tree. We base our work on the classical results of Guerra and Morato [3] on the formulation of quantum-mechanical behaviour in terms of controlled stochastic processes; the picture of a quantum walk that emerges through its stochastic analogue is that of a swarm of walkers moving according to transition rules involving the distribution of the *entire* swarm.

Consider a quantum system having as state space a Hilbert space the dimension of which we will indicate by s, and as generator of the time evolution a Hamiltonian operator that we will indicate by H.

Having fixed an orthonormal basis, $|\,\phi_1\,\rangle, |\,\phi_2\,\rangle, \ldots, |\,\phi_s\,\rangle$, a graph G is defined, *starting from the selected basis and from the selected Hamiltonian*, by stating that G has $\Lambda_s = \{1, ..., s\}$ as its set of vertices and edges $\{k, j\}$ such that $j \neq k$ and $|\langle\, \phi_k\, |H|\, \phi_j\, \rangle| > 0$.

In the context of this section, the graph G will play the role played, in the more elementary context of *Example 1* of section 1, by the linear graph having the relative integers as vertices, with edges between nearest neighbour sites. Similarly, the role played in section 1 by equation (4) will be played in this section by the Schrödinger equation in the representation determined by the selected basis:

$$i\frac{d}{dt}\psi(t,k) = \sum_{j=1}^{s} H_{k,j} \cdot \psi(t,j) \tag{19}$$

with

$$H_{k,j} = \langle\, \phi_k\, |H|\, \phi_j\, \rangle. \tag{20}$$

We pose in the following terms the question of finding in the general context of this section, an analogue of *Example 2* of section 1:

Easy problem: find a constructive procedure associating with each solution ψ of (19) a Markov process $q(t)$ on the graph G having at each time t probability distribution

$$\rho(t,k) = P(q(t) = k) = |\psi(t,k)|^2. \tag{21}$$

If this process exists and satisfies (for a suitable field ν of transition probabilities per unit time) the condition, that we impose as an analogue of conditions (10) and (11),

$$p(t+\tau, j; t, k) \equiv P(q(t+\tau) = j | q(t) = k) = \tau \cdot \nu_j(t,k) + o(\tau) \tag{22}$$

for $\tau \to 0^+$ and for each j being a neighbour of k in the graph G, then it will satisfy the continuity equation

$$\frac{d}{dt}\rho(t,k) = \sum_{j \in N(k)} \rho(t,j)\nu_k(t,j) - \rho(t,k)\nu_j(t,k) = \tag{23}$$

$$= \sum_{j \in N(k)} (\rho(t,j)\nu_k(t,j) + \rho(t,k)\nu_j(t,k)) \left(\frac{\rho(t,j)\nu_k(t,j) - \rho(t,k)\nu_j(t,k)}{\rho(t,j)\nu_k(t,j) + \rho(t,k)\nu_j(t,k)} \right).$$

In the above equation, we have indicated by $N(k)$ the set of neighbours in G of the vertex k, namely the collection of vertices j such that $j \neq k$ and $\{j,k\}$ is an edge.

In the second line of equation (23) we have separated the term

$$\rho(t,j)\nu_k(t,j) + \rho(t,k)\nu_j(t,k),$$

symmetric in j and k (on the analogue of which we have imposed in Section 1 the *geometric mean rule*), from the antisymmetric term

$$\frac{\rho(t,j)\nu_k(t,j) - \rho(t,k)\nu_j(t,k)}{\rho(t,j)\nu_k(t,j) + \rho(t,k)\nu_j(t,k)},$$

of absolute value ≤ 1, representing the net relative flux of probability mass *from* j *into* k.

If, now, the same ρ appearing in (23) satisfies also $\rho(t,k) = |\psi(t,k)|^2$ for a ψ satisfying (19), it must be

$$\psi(t,k) = \sqrt{\rho(t,x)}\exp(i \cdot S(t,k)) \tag{24}$$

for some phase function S to be determined by inserting the Ansatz (24) into equation (19). Doing so, and separating the real and imaginary parts of the resulting equation, one gets two equations:

$$\frac{d}{dt}S(t,k) = -H_{k,k} - \sum_{j\in N(k)} h_{k,j}\sqrt{\frac{\rho(t,j)}{\rho(t,k)}}\cos(\beta_{k,j}(t)) \tag{25}$$

and

$$\frac{d}{dt}\rho(t,k) = \sum_{j\in N(k)} 2h_{k,j}\sqrt{\rho(t,k)\rho(t,j)}\sin(\beta_{k,j}(t)), \tag{26}$$

where we have set

$$h_{k,j} = |H_{k,j}| \tag{27}$$

and

$$\beta_{k,j}(t) = Arg(H_{k,j}) + S(t,j) - S(t,k). \tag{28}$$

In order to check that our *Easy problem* admits at least one solution, it is sufficient to compare the purely kinematic relations

$$\frac{d}{dt}\rho(t,k) = \sum_{j\in N(k)} 2h_{k,j}\sqrt{\rho(t,k)\rho(t,j)}\sin(\beta_{k,j}(t)),$$

and

$$\frac{d}{dt}\rho(t,k) = \sum_{j\in N(k)} (\rho(t,j)\nu_k(t,j)+\rho(t,k)\nu_j(t,k))\left(\frac{\rho(t,j)\nu_k(t,j) - \rho(t,k)\nu_j(t,k)}{\rho(t,j)\nu_k(t,j) + \rho(t,k)\nu_j(t,k)}\right),$$

viewed, for assigned ψ and therefore for assigned ρ and β, as constraints on the unknown transition probabilities per unit time of the process $q(t)$ to be constructed. The simplest way to satisfy this constraint is by requiring term by term equality in the sums that appear in the right hand sides, and by equating in each term the symmetric and antisymmetric factors.

We thus get the equations

$$\rho(t,j)\nu_k(t,j) + \rho(t,k)\nu_j(t,k) = 2h_{k,j}\sqrt{\rho(t,k)\rho(t,j)} \tag{29}$$

$$\rho(t,j)\nu_k(t,j) - \rho(t,k)\nu_j(t,k) = 2h_{k,j}\sqrt{\rho(t,k)\rho(t,j)}\sin(\beta_{k,j}(t)) \tag{30}$$

that are solved by

$$\begin{aligned}\nu_k(t,j) &= h_{k,j}\sqrt{\frac{\rho(t,k)}{\rho(t,j)}}\,(1+\sin(\beta_{k,j}(t))) = \\ &= h_{k,j}\sqrt{\frac{\rho(t,k)}{\rho(t,j)}}\,(1+\sin(Arg(H_{k,j}) + S(t,j) - S(t,k)))\end{aligned} \tag{31}$$

for $k \in N(j)$.

For more details, and for the physical motivation (related to questions of time reversal invariance) of the merits of this particular choice, we refer to [4].

It is immediate to check that (29) is precisely the *geometric mean rule* (i) of section 1.

It is also an easy exercise to check that (31) specializes, due to the phase factor i^x in equation (3), to conditions (ii) and (iii) in the simple context of section 1.

3 Autonomous Generation

The *Hard problems*, as opposed to the kinematical *Easy problem* reviewed in section 2, are

I. understand (25) as a dynamical condition on the processes $q(t)$ that solve our *Easy problem*;
II. *autonomously* simulate these processes by actual implementation of this dynamical condition.

Problem I is discussed in full detail in [4] following the general approach of [3] in which stochastic control theory is successfully proposed as a very simple model simulating quantum-mechanical behaviour.

We are not able to tackle problem (II) in its generality. We can only go back to section 1 and show that the three dynamical rules and the initial conditions stated there in assigning *Example 2* are enough to generate the sample paths of figure 2.

This is far from obvious because of the *geometric mean rule*: it requires, in the numerical generation of each sample path, to take into account the *causative effect* (through the *estimated* probability distribution) on each trajectory of the *ensemble* of trajectories to which it belongs [5] .

Even to show, as in figure 2, a small sample of trajectories, the need of carefully estimating at each time step the density ρ imposes the simultaneous generation of a large number $N_{tr.}$ of trajectories.

The numerical procedure leading to figure 2, makes, by purpose, no reference to the solution of the continuity equation we have given in section 1, nor to the solution of the Kolmogorov equations for the conditional probabilities $p(t, x; t_0, x_0)$ that can be easily found by similar techniques. We present here this procedure in some detail because the challenges one meets in simulating the process $q(t)$ by implementing rules (i), (ii), (iii) and the initial conditions listed in section 1 give an operational meaning to the notion of *autonomous* simulation.

The state of the system at each time $t = \tau \cdot k$, where the integer k runs from 1 to n_{steps} and τ is the time step, is described by the pair

- *configuration array* of length N_{tr}: its j-th element $q_j(t)$ indicates the current position of the j-th trajectory; a space cut-off is introduced through an integer parameter L such that each trajectory is followed as long as $-L \leq q(t) \leq L$; the empirical density ρ_{emp} of the process at each time is estimated from the *configuration array*;
- *transition array* indexed from $-L$ to L: its x-th element is an ordered pair of bits (m_x, l_x): if $m_x = 1$ (resp. $l_x = 1$) then transitions $x \to x - 1$ (resp. $x \to x + 1$) are allowed, whereas if $m_x = 0$ (resp. $l_x = 0$) they are forbidden.

In our implementation $n_{tr.} = 5 \cdot 10^4$, $\tau = 0.05$, and the process has been followed up to time $t_{max} = \tau \cdot n_{steps} = 100$, well beyond the time window shown in figure 2; the space cut-off has been set at $L = 150$.

The algorithm consists of the iteration n_{steps} times of the following steps:

1. estimate ρ_{emp} from the *configuration array*;
2. increment each $q_j(t)$ by $Move(t, q_j(t))$, where the random variable $Move(t, x)$ takes the values $-1, 0, +1$ with probabilities $\tau \cdot \mu_{emp}(t, x), 1 - \tau \cdot (\mu_{emp}(t, x) + \lambda_{emp}(t, x)), \tau \cdot \lambda_{emp}(t, x)$, respectively.

   The empirical transition rates λ_{emp} and μ_{emp} are here given by

$$\lambda_{emp}(t, x) = \sqrt{\frac{\rho_{emp}(t, x+1)}{\rho_{emp}(t, x)}}\, l_x; \tag{32a}$$

$$\mu_{emp}(t, x) = \sqrt{\frac{\rho_{emp}(t, x-1)}{\rho_{emp}(t, x)}}\, m_x, \tag{32b}$$

3. estimate the new empirical distribution $\rho_{emp}(t + \tau, x)$;
4. if $\rho_{emp}(t, x) > 0$ and $\rho_{emp}(t + \tau, x) = 0$ then update the *transition array* following the *"horror vacui" rule* (iii), namely setting $l_{x-1} = 1$ and $m_{x+1} = 1$, and restore the *local unidirectionality rule* by setting $(m_x, l_x) = (0, 0)$.

In the initialization step the *transition array* has been given the initial assignment

$$(m_x, l_x) = \begin{cases} (1, 0) \text{ if } x < 0 \\ (1, 1) \text{ if } x = 0 \\ (0, 1) \text{ if } x > 0 \end{cases}$$

Before discussing the initialization of the *configuration array*, we observe that the rough first order updating rule of step 2. runs into trouble if a proposed transition involves a site at which ρ_{emp} vanishes, the problem being with zeroes of the numerators under the square root of (32a) and (32b). The most evident form of this fact is that, given that the process at time 0 is at position 0, the probability that it moves at all in a time step is

$$1 - J_0^2(\tau) = \frac{\tau^2}{2} + O(\tau^4). \tag{33}$$

We have found an inexpensive way out of this difficulty by initializing the *configuration array* by the assignment:

$$q_j(0) = 0, \text{ for } j = (2L+1)+1 \ldots, N_{tr}.$$
$$q_j(0) = j - L - 1, \text{ for } j = 1, \ldots, 2L+1$$

The first line says that most of the trajectories start from the origin; the second that trajectory 1 starts from $-L$, ..., trajectory $2L+1$ starts from L. The second line makes sure that initially there is at least one trajectory per site; this situation is restored, after step 4. by:

5. set $q_j(t+\tau) = j - L - 1$, for $j = 1, \ldots, 2L+1$.

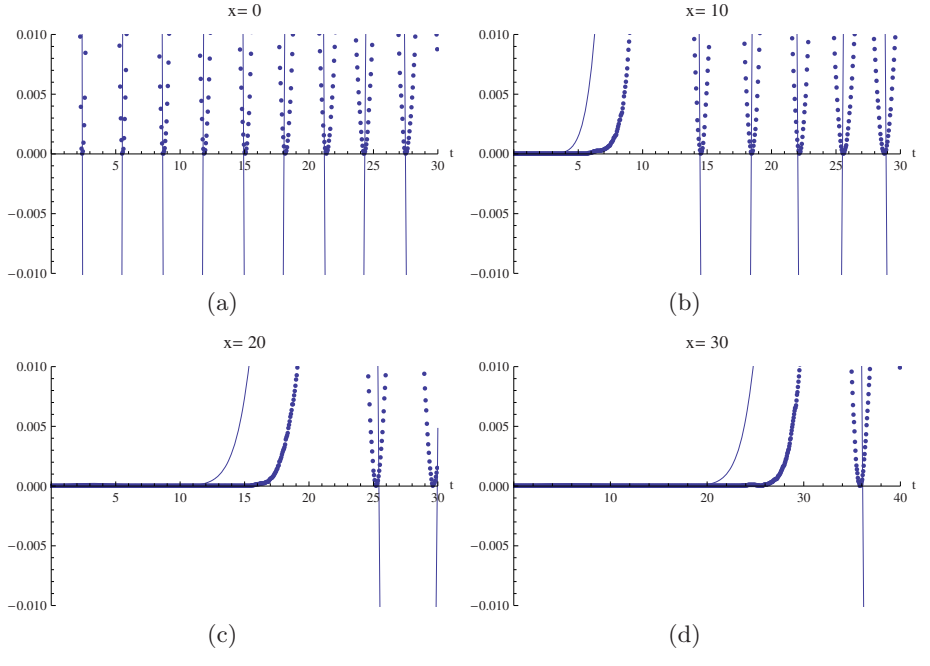


Fig. 3. Thin solid lines: graphs of $J_x(t)$ as a function of t for several values of x. Thick dashed lines: graphs, as a function of t, of the fraction of trajectories that visit x at time t.

The dummy trajectories labelled by $j = 1, \ldots, 2L+1$ provide some probability mass when needed to prevent the first order procedure from getting stuck.

Comparison between figures 1 and 2 gives an idea of how well our simple procedure fills the *configuration array* .

An analogous comparison is conducted in figure 3 for the *transition array*: the issue there is how well our procedure catches the instants of time at which the *control mechanism* expressed by the "*horror vacui" rule* takes hold, namely the zeroes of $J_x(t)$.

4 Conclusions and Outlook

If you give me a quantum walk efficiently exploring a graph or decision tree [6], I take your computational basis, your initial condition and your Hamiltonian and cook for you a stochastic process by computing its transition probabilities per unit time according to the recipe of section 2 and its transition probabilities $p(t, x; t_0, x_0)$ by integration of the Kolmogorov equations (this can be done in quite explicit terms for the *Example 2* of section 1) or by a clever exploitation of a few rules controlling the dynamics, as done in section 3. The discussion of section 2 makes it clear that my *random* walk will, by construction, visit your graph or decision tree as efficiently as your *quantum* walk.

Can the above statement be reconciled with the statement that the quantum *glued trees* algorithm of [7] outperforms any classical algorithm? How are the *classical alternatives* defined in the original literature on exponential speedup by quantum walk? Does the *causative effect of the ensemble* disqualify a Markov process from being classical ?

On these points, all we can do is to advance a conjecture: the cost of my random simulation of your quantum walk is hidden in the size $N_{tr.}$ of the sample I am required to generate. We have indeed called attention, since section 1, on the *geometric mean rule*: for every edge of your graph the probability per unit time of a transition of my process along that edge is equal to the geometric mean of the probabilities of the process at the two vertices joined by that edge.

We pose as a problem of future research the quantitative assessment of the cost (as measured by $N_{tr.}$) of the density estimation step required before each updating in the simulation.

References

1. Childs, A., Farhi, E., Gutmann, S.: An example of the difference between quantum and classical random walks. Quantum Information Processing 1, 35–43 (2002)
2. Perets, H., Lahini, Y., Pozzi, F., Sorel, M., Morandotti, R., Silberberg, Y.: Realization of quantum walks with negligible decoherence in waveguide lattices, arXiv:quant-ph/0707.0741v2 (2007)
3. Guerra, F., Morato, L.: Quantization of dynamical systems and stochastic control theory. Phys. Rev. D 27, 1774–1786 (1983)
4. Guerra, F., Marra, R.: Discrete stochastic variational principles and quantum mechanics. Phys. Rev. D 29(8), 1647–1655 (1984)

5. Smolin, L.: Could quantum mechanics be an approximation to another theory?, arXiv-quant-ph/0609109 (2006)
6. Farhi, E., Gutmann, S.: Quantum computation and decision trees. Phys. Rev. A 58, 915–928 (1998)
7. Childs, A., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., Spielman, D.: Exponential algorithmic speed up by quantum walk. In: STOC 2003. Proc. 35th ACM symp., pp. 59–68 (2003)

A Memetic Algorithm for Global Induction of Decision Trees

Marek Krętowski

Faculty of Computer Science, Białystok Technical University
Wiejska 45a, 15-351 Białystok, Poland
mkret@wi.pb.edu.pl

Abstract. In the paper, a new memetic algorithm for decision tree learning is presented. The proposed approach consists in extending an existing evolutionary approach for global induction of classification trees. In contrast to the standard top-down methods, it searches for the optimal univariate tree by evolving a population of trees. Specialized genetic operators are selectively applied to modify both tree structures and tests in non-terminal nodes. Additionally, a local greedy search operator is embedded into the algorithm, which focusses and speeds up the evolutionary induction. The problem of over-fitting is mitigated by suitably defined fitness function. The proposed method is experimentally validated and preliminary results show that the proposed approach is able to effectively induce accurate and concise decision trees.

1 Introduction

Evolutionary Computations is the name commonly used for describing a group of optimization and search techniques inspired by the process of natural evolution. Their main advantages over greedy search methods is their ability to avoid local optima. On the other hand it is known that pure evolutionary methods are not the fastest methods and a lot of effort is put into speeding them up. One of the possible solutions is a combination of evolutionary approach with local search techniques, which is known as *Memetic Algorithms* [10]. However, designing a competent memetic algorithm for a given problem is not an easy task and a number of important issues have to be addressed (e.g. where and when local search should be applied during the evolutionary search).

In this paper, an evolutionary learning of decision trees based on the training dataset is investigated. There are two main approaches to induction of decision trees: top-down and global. In the first approach, the optimal test searches and data splitting are recursively repeated to consecutive subsets of the training data until the stoping condition is not met. Usually, the growing phase is followed by the post-pruning. Apart of the classical top-down system like *CART* [3] or *C4.5* [18], several EC-based systems which learn (mainly oblique) decision trees in the top-down manner (e.g. *BTGA* [5], *OC1-ES* [4], *DDT-EA* [11]) have been proposed so far.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 531–540, 2008.

In this paper, the second approach to decision tree induction is advocated. In contrast to the step-wise construction, the whole tree is being searched at the time. It means the simultaneous search for an optimal structure of the tree and for all tests in non-terminal nodes. This process is obviously much more computationally complex but it can reveal hidden regularities, which are almost undetectable by greedy methods

The global approach was initially proposed by Koza in [9], where genetic programming was used for evolving LISP S-expressions that correspond to simple decision trees. A similar idea was investigated in the *GATree* system [17] which directly evolves classification trees with nominal tests. Fu *et al.* proposed a genetic algorithm called *GAIT* [8], which evolves binary trees initially obtained by applying *C4.5* on small sub-samples of the original data. Two simple genetic operator are utilized and individual performance is judged by measuring classification accuracy on the validation set. It should be noted that only tests from initial trees can be used in the internal nodes. Another interesting global system is called *GALE* [15]. It is a fine-grained parallel evolutionary algorithm for evolving both orthogonal and oblique decision trees. *GALE* uses squared reclassification accuracy as a fitness and simple operators (one point cross-over from genetic programming and random perturbation of the test).

In the paper, for the first time a memetic algorithm is proposed for global induction of decision trees. It combines typical evolution of trees with the local search for optimal tests in non-terminal nodes. The local optimality criteria come from *CART* and *C4.5* systems. This kind of hybridization should profit from both global and greedy methods and should improve the efficiency of the search.

The rest of the paper is organized as follows. In the next section the proposed memetic algorithm for global induction of univariate decision trees is described. Experimental validation of the method on artificial and real-life data is presented in section 4. In the last section, the paper is concluded and possible future works are sketched.

2 Memetic Algorithm for Global Induction

As the presented system evolved from our previous classical evolutionary algorithm [12,13,14], the general structure of the memetic algorithm follows the standard evolutionary framework [16]. The local search component responsible of the optimal test search in internal nodes is introduced in the initialization and embedded into the mutation operator.

2.1 Representation, Initialization and Termination Condition

Representation. There are two ways of representing candidate solutions in the evolutionary search. In the first one, individuals are encoded in the fixed-size (usually binary) chromosomes and standard genetic operators can be used. The second possibility consists in applying more sophisticated representations

(e.g. variable-length) and developing specialized genetic operators. As a structure of the optimal decision tree for a given learning set is not known *a priori* it is obvious that the second approach is chosen for the global induction.

In the presented system, decision trees are represented in their actual form as classical univariate trees where each test in a non-terminal node concerns only one attribute (nominal or continuous valued). Additionally, in every node information about learning vectors associated with the node is stored. This enables the algorithm to perform more efficiently local structure and tests modifications during applications of genetic operators.

In case of a nominal attribute at least one value is associated with each branch. It means that an inner disjunction is built-in into the induction algorithm. For a continuous-valued feature typical inequality tests are considered. Specialized genetic operators consider only boundary thresholds [7] as potential splits. A boundary threshold for the given attribute is defined as a midpoint between such a successive pair of examples in the sequence sorted by the increasing value of the attribute, in which the examples belong to two different classes. All boundary thresholds for each continuous-valued attribute are calculated before starting the evolutionary induction [12]. It significantly limits the number of possible splits and focuses the search process. It should be however noted that locally applied the optimal test search can find a split, which is not based on a precalculated threshold. In all internal nodes except from the root only limited sub-sample of learning vectors can be used for the optimal test search.

Initialization. An initial population is usually randomly created with emphasis on diversity of candidate solutions, which is especially useful when large search space has to be penetrated. It is also known that proper initialization can focus and significantly speed up the search process.

In the presented system, initial individuals are created by applying the classical top-down algorithm to randomly chosen sub-samples of the original training data (10% of data, but not more then 500 examples). Additionally, for any initial tree one of five test search strategies in non-terminal nodes is applied. Three strategies come from the very well-known decision tree systems i.e. *CART* [3] and *C4.5* [18] and they are based on the corresponding optimality criteria: *GiniIndex*, *InfoGain* and *GainRatio*. The fourth strategy is dipolar [11], where a test splitting randomly selected mixed dipole (a pair of feature vectors from different classes) is found. The last strategy is a random combination of all the aforementioned strategies. The recursive partitioning is finished when all training objects in a node belong to the same class or the number of objects in a node is lower than the predefined value (default value: 5). Finally, the resulting trees are post-pruned according to the fitness function.

Termination condition. The evolution terminates when the fitness of the best individual in the population does not improve during the fixed number of generations (default value is equal 1000). This can be treated as a sign of algorithm convergence. Additionally, the maximum number of generations is specified, which allows limiting the computation time in case of a very slow convergence (default value: 10000).

2.2 Genetic Operators

There are two specialized genetic operators corresponding to the classical mutation and cross-over. Application of both operators can result in changes of the tree structure and tests in non-terminal nodes. Additionally the local search component is built into the mutation-like operator.

Mutation operator. A mutation-like operator [14] is applied with a given probability to a tree (default value is 0.8) and it guarantees that at least one node of the selected individual is mutated. Firstly, the type of the node (leaf or internal node) is randomly chosen with equal probability and if a mutation of a node of this type is not possible, the other node type is chosen. A ranked list of nodes of the selected type is created and a mechanism analogous to ranking linear selection [16] is applied to decide which node will be affected.

While concerning internal nodes, the location (the level) of the node in the tree and the quality of the subtree starting in the considered node are taken into account. It is evident that a modification of the test in the root node affects the whole tree and has a great impact, whereas a mutation of an internal node in lower parts of the tree has only a local impact. In the proposed method, nodes on higher levels of the tree are mutated with lower probability and among nodes on the same level the number of misclassified objects by the subtree is used to sort them. Additionally, perfectly classifying nodes with only leaves as descendants and with a test composed of one feature are excluded from a ranking, because their mutation cannot improve the fitness.

As for leaves, the number of objects from other classes than the decision assigned to the leaf is used to put them in order, but homogenous leaves are not included. As a result, leaves which are worse in terms of classification accuracy are mutated with higher probability.

Modifications performed by a mutation operator depend on the node type (i.e. if the considered node is a leaf node or an internal node). For a non-terminal node a few possibilities exist:

- A completely new test can be found. With the user defined probability (default value: 0.05) a new test can be locally optimized or can be chosen to split a randomly drawn mixed dipole from the learning subset associated with the node. The local search for the optimal test can be based on the following criteria: *GiniIndex*, *InfoGain* and *GainRatio*. It should be noted that for nominal features only tests with the maximal number of outcomes (no inner disjunction) are analyzed due to the computational complexity constraints.
- The existing test can be altered by shifting the splitting threshold (continuous-valued feature) or re-grouping feature values (nominal features). These modifications can be purely random or can be guided by dipolar principles of splitting mixed dipoles and avoiding to split pure ones.
- A test can be replaced by another test or tests can be interchanged,
- One sub-tree can be replaced by another sub-tree from the same node.
- A node can be transformed (pruned) into a leaf.

Modifying a leaf makes sense only if it contains objects from different classes. The leaf is transformed into an internal node and a new test is chosen in the aforementioned way.

Cross-over operator. There are also several variants of cross-over operators. Three of them start with selecting of cross-over positions in two affected individuals. One node is randomly chosen in each of two trees. In the most straightforward variant, the subtrees starting in the selected nodes are exchanged. This corresponds to the classical cross-over from genetic programming. In the second variant, which can be applied only when non-internal nodes are randomly chosen and the numbers of outcomes are equal, only tests associated with the nodes are exchanged. The third variant is also applicable only when non-internal nodes are drawn and the numbers of descendants are equal. Branches which start from the selected nodes are exchanged in random order. There is also a variant of crossover inspired by the dipolar principles. In the internal node in the first tree a cut mixed dipole is randomly chosen and for the cross-over the node with the test splitting this dipole is selected in the second tree.

Additional operations. The application of any genetic operator can result in a necessity for relocation of the input vectors between parts of the tree rooted in the modified node. Additionally the local maximization of the fitness is performed by pruning lower parts of the sub-tree on the condition that it improves the value of the fitness.

It was observed by Bennett *et al.* [2] that in oblique trees enlarging the margin, it is profitable in terms of classification accuracy. In the presented system, a simple mechanism called *centering* based on this observation is introduced and it is applied to the best decision tree found. In case of an inequality test, the threshold can also be shifted to half-distance between corresponding feature values. It should be noted that such a post-processing does not change the fitness corresponding to the final tree. The centering cannot be applied to tests based on nominal features. For them, another kind of test improvement is used. If there is an internal node with nominal test, and there are descendant leaves which have the same decision, then such leaves are merged and inner disjunction is used in the splitting node.

2.3 Selection

As a selection mechanism the ranking linear selection [16] is applied. Additionally, the chromosome with the highest value of the fitness function in the iteration is copied to the next population *(elitist strategy)*.

2.4 Fitness Function

A fitness function drives the evolutionary search process and is the most important and sensitive component of the algorithm. The goal of any classification system is the correct prediction of class labels of new objects, however such a

target function cannot be defined directly. Instead, the accuracy on the training data is often used. However, it is well-known that their direct optimization leads to an over-fitting problem. In a typical top-down induction of decision trees, the over-specialization problem is mitigated by defining a stopping condition and by applying a post-pruning [6].

In the presented approach a complexity term is introduced into the fitness function preventing the over-specialization. The fitness function, which is maximized, has the following form:

$$Fitness(T) = Q_{Reclass}(T) - \alpha \cdot (S(T) - 1), \tag{1}$$

where $Q_{Reclass}(T)$ is the re-classification quality, $S(T)$ is the size of the tree T expressed as the number of nodes and α is a relative importance of the complexity term (default value is 0.001) and a user supplied parameter. Subtracting 1.0 eliminates the penalty when the tree is composed of only one leaf (in majority voting). It is worth to mention that the equation (1) is a form of regularization with $S(T) - 1$ playing the role of a stabilizer and α the role of a regularization parameter.

It is rather obvious that there is no optimal value of α for all possible datasets. When the concrete problem is analyzed, tuning this parameter may lead to the improvement of the results (in terms of classification accuracy or classifier complexity).

3 Experimental Results

The proposed memetic approach (denoted as *GDT-MA*) to learning decision trees is assessed on both artificial and real life datasets and is compared to the well-known $C4.5$ system. It is also compared to the pure evolutionary versions of the global inducer - *GDT-AP*. All prepared artificial datasets comprise training and testing parts. Examples of artificial datasets are presented in Fig. 1. In case of data from a UCI repository [1] for which testing data are not provided, a 10-fold stratified cross-validation was employed. Each experiment on evolutionary algorithms was performed 10 times and the average result of such an evaluation was presented. All systems were tested with a default set of parameters.

3.1 Artificial Datasets

Results of experiments with artificial datasets are gathered in the Table 1. For all domains *GDT-MA* and *GDT-AP* performed very well, both in terms of classification accuracy and tree complexity. Compared to the *C4.5* system both global inducers were able to find a proper decision trees when top-down system failed and returned a default class.

3.2 Real-Life Datasets

Results obtained for the real-life datasets are gathered in Table 2. It can be observed that in terms of the classification accuracy *GDT-MA* performs comparable to *C4.5* (for certain datasets it is slightly better for other is slightly

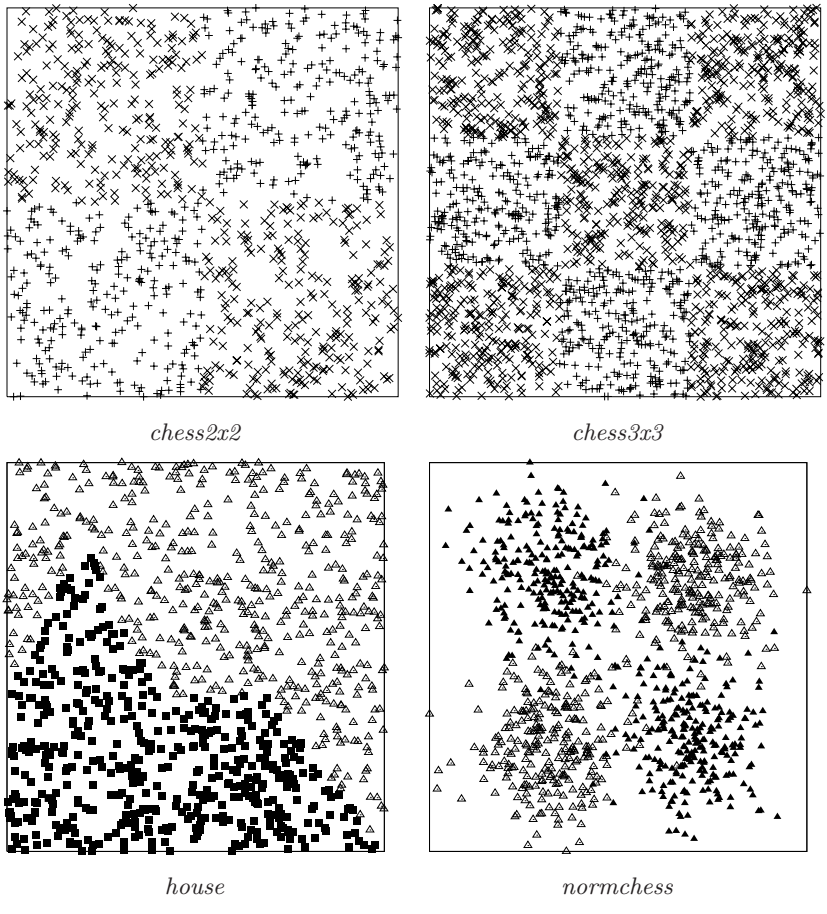


Fig. 1. Examples of artificial datasets

Table 1. Results on artificial data

	C4.5		GDT-MA		GDT-AP	
Dataset	size	quality	size	quality	size	quality
chess2x2	1	50	4	99.9	4	99.8
chess2x2x2	1	50	8	99.8	8	99.7
chess3x3	9	99.7	9	99.8	9	99.7
chess3x3x3	54	99.3	27.2	99.0	27.1	98.9
house	21	97.4	12.1	96.4	13.3	96.6
normchess	1	50	4.1	95.5	4.2	95.5
normwave	15	94	8.8	92.6	9.1	93.5

worse than its competitor). However, it is easily noticeable that in terms of the simplicity of the solution, the proposed memetic algorithm is significantly better

Table 2. Results on UCI datasets

Dataset	C4.5		GDT-MA		GDT-AP	
	size	quality	size	quality	size	quality
balance-scale	57	77.5	20.8	79.8	32.8	78.2
bcw	22.8	94.7	5.7	95.6	6.6	95.8
bupa	44.6	64.7	33.6	63.7	69.3	62.8
cars	31	97.7	3	97.9	4	98.7
cmc	136.8	52.2	19.2	55.7	13.1	53.8
german	77	73.3	18.4	74.2	16.5	73.4
glass	39	62.5	35.3	66.2	40.4	63.6
heart	22	77.1	29	76.5	44.9	74.2
page-blocks	82.8	97	7.4	96.5	7.5	96.4
pima	40.6	74.6	14.8	74.2	14.3	73.8
sat	435	85.5	18.9	83.8	19.2	83
vehicle	138.6	72.7	43.2	71.1	45.1	70.3
vote	5	97	10.9	96.2	13.5	95.6
waveform	107	73.5	30.7	71.9	36.2	72.3
wine	9	85	5.1	88.8	5.2	86.3

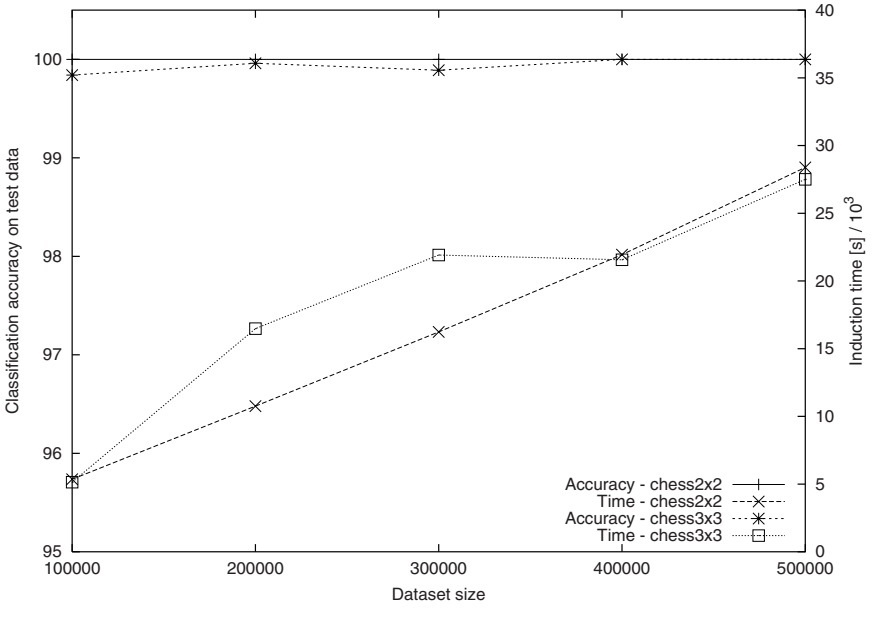


Fig. 2. Performance of the memetic algorithm on large datasets

that *C4.5*. It is also worth to mention that *GDT-MA* was more accurate than its pure evolutionary rival for 12 out of 15 analyzed real-life datasets.

3.3 Evaluation of Algorithm Performance on Large Datasets

In order to verify that the proposed method can be applied to large datasets, a performance test is conducted. The experiment was performed on two variants of the *chess* dataset: *ches2x2* and *chess3x3*, with increasing number of generated observations (starting from 100000 learning vectors up to 500000). In Fig. 2 obtained results in terms of the classification accuracy and the induction time are presented.

The promising outcome of this experiment is that it shows that the $GDT-MA$ system can deal with relatively large datasets (500000 observations) in acceptable time - 7 hours as measured on a typical machine (Xeon 3.2GHz, 2GB RAM). It should be noticed that for all datasets, optimal trees were found, both in terms of the classification accuracy and the tree size. It can be also observed that induction times scale almost linearly with the dataset size.

4 Conclusions

In the paper, for the first time a specialized memetic algorithm is developed for global induction of decision trees. The local search for optimal tests in non-terminal nodes based on the classical optimality criteria is embedded into the evolutionary search process. The necessary modification encompasses the initialization and the mutation operator. Even preliminary experimental validation shows that such a hybridization is profitable and improves the efficiency of the evolutionary induction.

The presented approach is still under development. First of all, the influence of the local search operator on the performance of the global inducer must be studied in more details. Furthermore, additional optimality criteria (e.g. *TwoingRule* from the *CART* system) are planned to be implemented.

Acknowledgments

This work was supported by the grant W/WI/5/05 from Białystok Technical University.

References

1. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Bennett, K., Cristianini, N., Shave-Taylor, J., Wu, D.: Enlarging the margins in perceptron decision trees. Machine Learning 41, 295–313 (2000)
3. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group (1984)
4. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. IEEE Transactions on Evolutionary Computation 7(1), 54–68 (2003)

5. Chai, B., Huang, T., Zhuang, X., Zhao, Y., Sklansky, J.: Piecewise-linear classifiers using binary tree structure and genetic algorithm. Pattern Recognition 29(11), 1905–1917 (1996)
6. Esposito, F., Malerba, D., Semeraro, G.: A comparative analysis of methods for pruning decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(5), 476–491 (1997)
7. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. of IJCAI 1993, pp. 1022–1027. Morgan Kaufmann, San Francisco (1993)
8. Fu, Z., Golden, B., Lele, S., Raghavan, S., Wasil, E.: A genetic algorithm-based approach for building accurate decision trees. INFORMS Journal on Computing 15(1), 3–22 (2003)
9. Koza, J.: Concept formation and decision tree induction using genetic programming paradigm. In: Schwefel, H.-P., Männer, R. (eds.) PPSN I. LNCS, vol. 496, pp. 124–128. Springer, Heidelberg (1991)
10. Krasnogor, N., Smith, J.E.: A tutorial for competent memetic algorithms: model, taxonomy and design issues. IEEE Transactions on Evolutionary Computation 9(5), 474–488 (2005)
11. Krętowski, M.: An evolutionary algorithm for oblique decision tree induction. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 432–437. Springer, Heidelberg (2004)
12. Krętowski, M., Grześ, M.: Global learning of decision trees by an evolutionary algorithm. In: Information Processing and Security Systems, pp. 401–410. Springer, Heidelberg (2005)
13. Krętowski, M., Grześ, M.: Evolutionary learning of linear trees with embedded feature selection. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 400–409. Springer, Heidelberg (2006)
14. Krętowski, M., Grześ, M.: Evolutionary induction of mixed decision trees. International Journal of Data Warehousing and Mining 3(4), 68–82 (2007)
15. Llora, X., Garrell, J.: Evolution of decision trees. In: Proc. of CCAI 2001, pp. 115–122. ACIA Press (2001)
16. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
17. Papagelis, A., Kalles, D.: Breeding decision trees using evolutionary techniques. In: Proc. of ICML 2001, pp. 393–400. Morgan Kaufmann, San Francisco (2001)
18. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)

Geometric Rates of Approximation by Neural Networks

Věra Kůrková[1] and Marcello Sanguineti[2]

[1] Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, Prague 8, Czech Republic
vera@cs.cas.cz
[2] Department of Communications, Computer, and System Sciences (DIST)
University of Genova, Via Opera Pia 13, 16145 Genova, Italy
marcello@dist.unige.it

Abstract. Model complexity of feedforward neural networks is studied in terms of rates of variable-basis approximation. Sets of functions, for which the errors in approximation by neural networks with n hidden units converge to zero geometrically fast with increasing number n, are described. However, the geometric speed of convergence depends on parameters, which are specific for each function to be approximated. The results are illustrated by examples of estimates of such parameters for functions in infinite-dimensional Hilbert spaces.

1 Introduction

Feedforward neural networks can be formally described as devices producing input-output functions depending on flexible parameters. Often input-output functions have the form of linear combinations of functions computable by units specific for the given type of a network. Both coefficients of the linear combinations and parameters of the computational units are adjustable in the process of learning.

Networks with a moderate number of computational units have been successfully used in many pattern recognition and classification tasks, some of them high-dimensional. So in some cases, networks with a relatively small model complexity can provide a good approximation of functions with a large number of variables. Thus neural networks seem to be more suitable for high-dimensional tasks than linear models, whose model complexity grows exponentially with improvements of accuracy in approximation of functions defined by certain smoothness conditions [1]. In linear models, merely coefficients of linear combinations of the first n elements from a basis with a *fixed* ordering are adjustable. Neural networks belong to the class of nonlinear models, which are sometimes called *variable-basis* ones. During learning in addition to coefficients of linear combinations, also parameters of computational units are searched for. Thus a suitable n-tuple of functions corresponding to the type of network units (such as perceptrons or radial-basis functions) is chosen.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 541–550, 2008.

Some insight why in solving high-dimensional tasks variable-basis models can perform better than linear models can be obtained from an estimate of rates of variable-basis approximation by Maurey, Jones, and Barron [2,3,4]. For functions from the convex hull of a bounded subset G of a Hilbert space, they derived an upper bound on the square of the error in approximation by convex combination of n elements of G. The bound has the form $\frac{1}{n}\left(s_G^2 - \|f\|^2\right)$, where s_G is the supremum of norms of elements of G and $\|f\|$ is the norm of the function to be approximated.

Several authors derived tight improvements of Maurey-Jones-Barron's estimate for various sets G (e.g., G orthogonal [5,6], G formed by function computable by sigmoidal perceptrons [7] or G with certain property of covering numbers [8]). However, all these tightness results are worst-case estimates (they give upper bounds holding for all functions from convex hulls). Thus one can expect that for some subsets of such hulls, much better rates might hold.

The first step to description of such subsets was done by Lavretsky [9]. He noticed that when in the iterative construction derived by Jones [3] and improved by Barron [4], in each step only functions satisfying a certain angular condition are chosen, then the term $\frac{1}{n}$ in the Maurey-Jones-Barron upper bound can be replaced with $(1-\delta^2)^{n-1}$, where $\delta \in (0,1]$ is the arccosine corresponding to the angular constraint. However, Lavretsky left open the problem of characterization of functions satisfying the angular condition. He illustrated his result by one example, which, however, sounds as a sort of tautology: he showed that a function that can be expressed as a series formed by orthonormal functions with geometrically decaying coefficients can be approximated by the partial sums with geometrically fast rate.

In this paper we extend Lavretsky's result by showing that for every function f in the convex hull of G there exists $\delta_f \in (0,1]$ such that the rate of approximation of f by convex combinations of n functions from G is bounded from above by $(1-\delta_f^2)^{n-1}\left(s_G^2 - \|f\|^2\right)$. However, we do not prove that every function in the convex hull satisfies the angular condition implying Lavretsky's estimate, instead we derive the geometric rate by further modifying the incremental construction originally used by Jones [3] and later improved by Barron [4] and refined by Lavretsky [9].

The paper is organized as follows. In section 2, Maurey-Jones-Barron's theorem and its improvements are stated together with necessary terminology. In Section 3 our main theorem on geometric rates of approximation for functions in convex hulls together with its corollary are proven. In Section 4, examples of functions in convex hulls of orthonormal sets with estimates of the parameters determining geometric rates are given. Section 5 is a brief discussion.

2 Maurey-Jones-Barron's Theorem and Its Improvements

Many computational models used in soft-computing can be mathematically described as *variable-basis* schemes. Such models compute functions from sets of the form

$$\operatorname{span}_n G = \left\{ \sum_{i=1}^{n} w_i g_i \,\middle|\, w_i \in \mathbb{R},\, g_i \in G \right\},$$

where G is the set of functions that can be computed by computational units of a given type and $\mathbb{R}$ denotes the set of real numbers. Note that for G linearly independent, sets $\operatorname{span}_n G$ are not convex.

A useful tool for estimation of rates of decrease of errors in approximation by $\operatorname{span}_n G$ with n increasing is Maurey-Jones-Barron's theorem [2,3,4]. This theorem is formulated for approximation by

$$\operatorname{conv}_n G = \left\{ \sum_{i=1}^{n} a_i g_i \,\Big|\, a_i \in [0,1], \sum_{i=1}^{n} a_i = 1\,,\ g_i \in G \right\}.$$

The following estimate is a version of Jones' result [3] as improved by Barron [4] and also of an earlier estimate derived by Maurey [2]. For a subset M of a normed linear space $(X, \|.\|)$ and $f \in X$, we denote by $\|f - M\| = \inf_{g \in M} \|f - g\|$ the *distance* of f from M. By cl we denote the *closure* with respect to the topology induced by $\|.\|$.

Theorem 1 (Maurey-Jones-Barron). *Let $(X, \|.\|)$ be a Hilbert space, G its bounded subset, $s_G = \sup_{g \in G} \|g\|$ and $f \in \operatorname{cl}\operatorname{conv} G$. Then for every positive integer n,*

$$\|f - \operatorname{conv}_n G\|^2 \le \frac{s_G^2 - \|f\|^2}{n}.$$

In [10] (see also [11]), Theorem 1 was extended using the concept of G-variation defined for all functions $f \in X$ as

$$\|f\|_G = \min \left\{ c > 0 \,|\, \frac{f}{c} \in \operatorname{cl}\operatorname{conv}(G \cup -G) \right\}.$$

Note that $\|.\|_G$ is the Minkowski functional of the set $\operatorname{cl}\operatorname{conv}(G \cup -G)$ and so it is a norm on the linear subspace of X containing those $f \in X$, for which $\|f\|_G < \infty$. It is easy to check that Theorem 1 implies that for a Hilbert space $(X, \|.\|)$, G its bounded subset with $s_G = \sup_{g \in G} \|g\|$, and $f \in X$,

$$\|f - \operatorname{span}_n G\|^2 \le \frac{(s_G \|f\|_G)^2 - \|f\|^2}{n}. \tag{1}$$

Lavretsky [9] noticed that the argument derived by Jones [3] and Barron [4] can yield better rates, when applied to functions satisfying a certain angular relationship with respect to G. For $\delta > 0$, he defined

$$F_\delta(G) = \big\{ f \in \operatorname{cl}\operatorname{conv} G \,|\, (\forall h \in \operatorname{conv} G,\ f \ne h)(\exists g \in G) \\ \big((f-g)\cdot(f-h) \le -\delta \|f-g\|\,\|f-h\|\big) \big\}. \tag{2}$$

Note that for all $\delta > 0$, $G \subseteq F_\delta$ (indeed, for every $f \in G$ setting $g = f$, we get $(f-f)\cdot(f-h) \le -\delta\|f-h\|\,\|f-f\|$).

Lavretsky realized that the incremental construction developed by Jones and Barron uses in each step a certain property of functions from the convex hulls, which restated in his terminology says that $\operatorname{conv} G = F_0(G)$. Strengthening the

condition on the function f to be approximated by assuming that $f \in F_\delta(G)$ for some $\delta > 0$, he derived the following geometric estimate of rates of approximation by sets $\mathrm{conv}_n G$ [9, Theorem 1].

Theorem 2 (Lavretsky). *Let $(X, \|\cdot\|)$ be a Hilbert space, G its bounded symmetric subset containing 0, $s_G = \sup_{g\in G} \|g\|$ and $\delta > 0$. Then for every $f \in F_\delta(G)$, and every positive integer n,*

$$\|f - \mathrm{conv}_n G\|^2 \leq (1-\delta^2)^{n-1}(s_G^2 - \|f\|^2)\,.$$

Unfortunately, the definition (2) of F_δ does not enable an easy verification whether a function is in F_δ. Even, it is not clear whether sets F_δ contain other functions than the elements of G. Only for finite-dimensional Hilbert spaces and G satisfying certain conditions, Lavretsky [9, Theorem 2] described subsets of $\mathrm{conv}\, G$, called "affine interiors", with the property that for each their element f there exists $\delta_f \in (0,1]$, for which $f \in F_\delta$. For a convex subset M of a normed linear space $(X, \|.\|)$ such that $0 \in M$, Lavretsky defined its *affine interior* as

$$I_{Aff}(M) = \{f \in M \,|\, (\exists \varepsilon_f > 0)(\forall h \in \mathrm{span} M)(\|h\| < \varepsilon_f \Rightarrow f + h \in M\}.$$

Theorem 3 (Lavretsky). *Let $(X, \|\cdot\|)$ be a finite-dimensional Hilbert space, G its bounded symmetric subset such that $0 \in G$ and $\mathrm{card}\, G \geq \dim X$, and $s_G = \sup_{g\in G} \|g\|$. Then $I_{Aff}(G) \neq \emptyset$ and for every $f \in I_{Aff}(G)$, there exists $\delta_f \in (0,1]$ such that $f \in F_{\delta_f}(G)$.*

Thus under the assumptions of Theorem 3, for every $f \in I_{Aff}(G)$ there exists $\delta_f \in (0,1]$ such that

$$\|f - \mathrm{conv}_n G\|^2 \leq (1-\delta_f^2)^{n-1}(s_G^2 - \|f\|^2)\,. \tag{3}$$

3 Geometric Rates of Variable-Basis Approximation

In this section, we show that for *every function f in the convex hull of any bounded subset G of any Hilbert space* there exists $\delta_f \in (0,1]$ such that (3) holds. Thus we considerably extend Lavretsky's result (Theorem 3) on functions admitting geometric rates of approximation.

Theorem 4. *Let $(X, \|.\|)$ be a Hilbert space, G its bounded subset, $s_G = \sup_{g\in G} \|g\|$. Then for every $f \in \mathrm{conv}\, G$ there exists $\delta_f \in (0,1]$ such that for every positive integer n*

$$\|f - \mathrm{conv}_n G\|^2 \leq (1-\delta_f^2)^{n-1}\,\left(s_G^2 - \|f\|^2\right).$$

Proof. Let $f = \sum_{j=1}^m a_j\, g_j$ be a representation of f as a convex combination of elements of G with all $a_j > 0$ and let $G' = \{g_1, \ldots, g_m\}$. We shall construct a sequence of functions $\{f_n \,|\, n = 1, \ldots, m\}$ and a sequence of positive real numbers

$\{\delta_n \,|\, n = 1, \ldots, m\}$ such that for each $n = 1, \ldots, m$, $f_n \in \mathrm{conv}_n G$ and $\|f - f_n\|^2 \leq (1 - \delta_n)^{n-1} \left(s_G^2 - \|f\|^2\right)$.

We start with choosing some $g_{j_1} \in G'$ satisfying $\|f - g_{j_1}\| = \min_{g \in G'} \|f - g\|$ and we set $f_1 = g_{j_1}$. As $\sum_{j=1}^m a_j \|f - g_j\|^2 = \|f\|^2 - 2f \cdot \sum_{i=1}^m a_j g_j + \sum_{j=1}^m a_j \|g_j\|^2 \leq s_G^2 - \|f\|^2$, we get $\|f - f_1\|^2 \leq s_G^2 - \|f\|^2$ and so the statement holds for $n = 1$.

Assuming that we already have f_{n-1}, we define f_n. When $f_{n-1} = f$, we set $f_n = f_{n-1}$ and the estimate holds trivially.

When $f_{n-1} \neq f$, we define f_n as the convex combination

$$f_n = \alpha_n f_{n-1} + (1 - \alpha_n) g_{j_n}, \tag{4}$$

with $g_{j_n} \in G'$ and $\alpha_n \in [0, 1]$ chosen in such a way that for some $\delta_n > 0$

$$\|f - f_n\|^2 \leq (1 - \delta_n^2)^{n-1} \|f - f_{n-1}\|^2.$$

First, we choose a suitable g_{j_n} and then we find α_n depending on our choice of g_{j_n}. Denoting $e_n = \|f - f_n\|$, we get by (4)

$$e_n^2 = \alpha_n^2 e_{n-1}^2 + 2\alpha_n(1 - \alpha_n)(f - f_{n-1}) \cdot (f - g_{j_n}) + (1 - \alpha_n)^2 \|f - g_{j_n}\|^2. \tag{5}$$

For all $j \in \{1, \ldots, m\}$, set

$$\eta_j = -\frac{(f - f_{n-1}) \cdot (f - g_j)}{\|f - f_{n-1}\| \, \|f - g_j\|}$$

(the definition is correct as now we are considering the case when $f \neq f_{n-1}$ and we have assumed that all $a_j > 0$ and thus for all j, $f \neq g_j$). Note that for all j, $\eta_j \in [0, 1]$ as it is the arccosine of the angle between the vectors $f - f_{n-1}$ and $f - g_j$.

As $f = \sum_{j=1}^m a_j \, g_j$, we have

$$\sum_{j=1}^m a_j (f - f_{n-1}) \cdot (f - g_j) = (f - f_{n-1}) \cdot (f - \sum_{j=1}^m a_j g_j) = 0.$$

Thus either (i) there exist some $g \in G'$, for which $(f - f_{n-1}) \cdot (f - g) < 0$ or (ii) for all $g \in G'$, $(f - f_{n-1}) \cdot (f - g) = 0$.

We show that the second possibility (ii) implies that $f = f_{n-1}$. Indeed, $f_{n-1} \in \mathrm{conv}_{n-1} G'$ and thus it can be expressed as $f_{n-1} = \sum_{k=1}^{n-1} b_k g_k$ with all $b_k \in [0, 1]$ and $\sum_{k=1}^{n-1} b_k = 1$. If for all $g \in G'$, $(f - f_{n-1}) \cdot (f - g) = 0$, then $\|f - f_{n-1}\|^2 = (f - f_{n-1}) \cdot (f - \sum_{k=1}^{n-1} b_k g_k) = \sum_{k=1}^{n-1} b_k (f - f_{n-1}) \cdot (f - g_k) = 0$.

So in the case now considered, i.e., $f \neq f_{n-1}$, (i) holds and thus the subset $G'' = \{g \in G' \,|\, (f - f_{n-1}) \cdot (f - g) < 0\}$ is nonempty. Let $g_{j_n} \in G''$ be chosen so that $\eta_{j_n} = \max_{j=1,\ldots,m} \eta_j$ and set $\delta_n = \eta_{j_n}$. As $G'' \neq \emptyset$, we have $\delta_n > 0$.

Set $r_n = \|f - g_{j_n}\|$. By (5) we get

$$e_n^2 = \alpha_n^2 e_{n-1}^2 - 2\alpha_n(1 - \alpha_n)\delta_n e_{n-1} r_n + (1 - \alpha_n)^2 r_n^2. \tag{6}$$

To define f_n as a convex combination of f_{n-1} and g_{j_n}, it remains to find $\alpha_n \in [0,1]$, for which e_n^2 is minimal as a function of α_n. By (6) we have

$$e_n^2 = \alpha_n^2 \left(e_{n-1}^2 + 2\delta_n e_{n-1} r_n + r_n^2\right) - 2\alpha_n \left(\delta_n e_{n-1} r_n + r_n^2\right) + r_n^2. \tag{7}$$

Thus

$$\frac{\partial e_n^2}{\partial \alpha_n} = 2\alpha_n \left(e_{n-1}^2 + 2\delta_n e_{n-1} r_n + r_n^2\right) - 2\left(\delta_n e_{n-1} r_n + r_n^2\right)$$

and

$$\frac{\partial^2 e_n^2}{\partial^2 \alpha_n} = 2\left(e_{n-1}^2 + 2\delta_n e_{n-1} r_n + r_n^2\right).$$

As now we are considering the case when $f \neq f_{n+1}$, we have $e_{n-1} > 0$ and hence $\frac{\partial e_n^2}{\partial^2 \alpha_n} > 0$. So the minimum is achieved at

$$\alpha_n = \frac{\delta_n e_{n-1} r_n + r_n^2}{e_{n-1}^2 + 2\delta_n e_{n-1} r_n + r_n^2}. \tag{8}$$

Plugging (8) into (5) we get

$$e_n^2 = \frac{(1-\delta_n^2) e_{n-1}^2 r_n^2}{e_{n-1}^2 + 2\delta_n e_{n-1} r_n + r_n^2} < \frac{(1-\delta_n^2) e_{n-1}^2 r_n^2}{r_n^2} = (1-\delta_n^2) e_{n-1}^2.$$

Let $k = \max\{n \in \{1,\dots,m\} \mid f_n \neq f_{n-1}\}$. Setting $\delta_f = \min\{\delta_n \mid n = 1,\dots,k\}$, we get by induction the upper bound

$$\|f - \mathrm{conv}_n G\|^2 \leq (1-\delta_f^2)^{n-1} \left(s_G^2 - \|f\|^2\right)$$

holding for all n (for $n > m$ it holds trivially with $f_n = f$). □

Note that we do not claim that every $f \in \mathrm{conv}\, G$ is an element of $F_{\delta_f}(\mathrm{conv}\, G)$ for some $\delta_f > 0$. In each step of our proof, we merely need a suitable angular relationship for one element of $\mathrm{conv}\, G$ (the one which was constructed as the approximant in the previous step) instead of all elements of $\mathrm{conv}\, G$ as in the case of membership in $F_{\delta_f}(\mathrm{conv}\, G)$.

Inspection of the proof of Theorem 4 suggests the incremental procedure (Fig. 1) constructing a sequence of approximants $f_n \in \mathrm{conv}_n G$ for a function $f = \sum_{j=1}^m a_j g_j \in \mathrm{conv}\, G$.

Theorem 4 implies an estimate holding for all functions in $\mathrm{span}\, G$.

Corollary 1. *Let $(X, \|.\|)$ be a Hilbert space, G its bounded subset, $s_G = \sup_{g\in G} \|g\|$, and $f \in \mathrm{span}\, G$. Then for every $b > 0$ with $f \in \mathrm{conv}(b(G \cup -G))$ there exists $\delta_{f,b} \in (0,1]$ such that for every positive integer n*

$$\|f - \mathrm{span}_n G\|^2 \leq (1-\delta_{f,b}^2)^{n-1} \left((s_G\, b)^2 - \|f\|^2\right).$$

Proof. By Theorem 4, there exists $\delta_{f,b} \in (0,1]$ such that for every positive integer n, $\|f - \mathrm{conv}_n(b(G \cup -G))\|^2 \leq (1-\delta_{f,b}^2)^{n-1} \left((s_G\, b)^2 - \|f\|^2\right)$. As $\|f - \mathrm{span}_n G\| \leq \|f - \mathrm{conv}_n\big(b(G \cup -G)\big)\|$ the statement follows. □

Note that we cannot extend Corollary 1 to $b_f = \inf\{b > 0 \mid f \in \mathrm{conv}(b(G \cup -G))\}$ as $\inf\{\delta_{f,b} \mid b > 0 \,\&\, f \in \mathrm{conv}(b(G \cup -G))\}$ might be equal to zero.

1. INITIALIZATION:
   - CHOOSE $g_{j_1} \in \{g_j \,|\, j = 1, \ldots, m\}$ SUCH THAT $\|f - g_{j_1}\| = \min_{j=1,\ldots,m} \|f - g_j\|$;
   - $f_1 = g_{j_1}$
2. FOR $n = 2, \ldots, m-1$:
   (A) FOR $j = 1, \ldots, m$, COMPUTE $\eta_j := -\frac{(f-f_{n-1})\cdot(f-g_j)}{\|f-f_{n-1}\|\,\|f-g_j\|}$
   (B) IF FOR $j = 1, \ldots, m$ ONE HAS $\eta_j = 0$, THEN
      - $f^* := f_{n-1}$;
      - $n^* := n-1$;
      - END.
   (C) ELSE
      - $\delta_n := \max\{\eta_j > 0 \,|\, j = 1, \ldots, m\}$;
      - CHOOSE SOME g_{j_n} WITH $\delta_n = \eta_{j_n}$;
      - COMPUTE $e_{n-1} := \|f - f_{n-1}\|$;
      - COMPUTE $r_n := \|f - g_{j_n}\|$;
      - COMPUTE $\alpha_n := \frac{\delta_n e_{n-1} r_n + r_n^2}{e_{n-1}^2 + 2\delta_n e_{n-1} r_n + r_n^2}$;
      - $f_n := \alpha_n f_{n-1} + (1 - \alpha_n)\, g_{j_n}$;
      - $n := n + 1$.

Fig. 1. Incremental procedure

4 Sets of Functions with Geometric Rates of Approximation

Inspection of the proof of Theorem 4 shows that δ_f is not defined uniquely: it depends on the choice of a representation of $f = \sum_{j=1}^{m} a_j g_j$ as a convex combination of elements of G and on the choice of functions g_{j_n} for those n with more than one g_j with the same arccosine δ_n. For each $\delta \in (0, 1]$ define

$$A_\delta(G) = \left\{ f \in \operatorname{conv} G \,|\, \|f - \operatorname{span}_n G\|^2 \leq (1-\delta^2)^{n-1} \left(s_G^2 - \|f\|^2\right) \right\}.$$

Theorem 4 implies that

$$\operatorname{conv} G = \bigcup_{\delta \in (0,1]} A_\delta(G).$$

The following proposition shows that when G is an orthonormal basis, then there exists a sequence of functions $\{h_k\} \in \operatorname{conv} G$ such that each sequence $\{\delta_k\}$, for which $h_k \in A_{\delta_k}$ for all k, converges to zero exponentially fast. Thus the estimate $(1-\delta_k^2)^{n-1}\left(1 - \|h_k\|^2\right)$ guarantees sufficiently small error only for rather large n.

Proposition 1. *Let $(X, \|.\|)$ be an infinite-dimensional separable Hilbert space and G its orthornormal basis. Then for every positive integer k there exists $h_k \in \operatorname{conv} G$ such that $\|h_k\| = \frac{1}{\sqrt{2k}}$, $\|h_k - \operatorname{conv}_k G\| \geq \frac{1}{2\sqrt{k}}$ and for every $\delta_k \in (0, 1]$ for which $h_k \in A_{\delta_k}(G)$,*

$$\delta_k^2 \leq 1 - 5^{\frac{1}{k-1}} e^{-\frac{\ln(k-1)}{k-1}}.$$

Proof. Let $G = \{g_i\}$. For each positive integer k define $h_k = 1/(2k)\sum_{i=1}^{2k} g_i$. Then $h_k \in \operatorname{conv} G$ and $\|h_k\| = 1/\sqrt{2k}$. It is easy to see that $\|h_k - \operatorname{span}_k G\| = 1/(2\sqrt{k})$. Thus by Theorem 4, $1/(4k) = \|h_k - \operatorname{span}_k G\|^2 \leq \|h_k - \operatorname{conv}_k G\|^2 \leq (1-\delta_k^2)^{k-1}\,(1-1/(2k))$. So

$$\delta_k^2 \leq 1 - \left(\frac{1}{2(2k-1)}\right)^{\frac{1}{k-1}} \leq 1 - \left(\frac{1}{5(k-1)}\right)^{\frac{1}{k-1}} = 1 - 5^{-\frac{1}{k-1}} e^{-\frac{\ln(k-1)}{k-1}}.$$

□

For $r > 0$ we denote by $B_r(\|.\|)$ the ball of radius r centered at 0, i.e., $B_r(\|.\|) = \{f \in X \mid \|f\| \leq r\}$.

Corollary 2. *Let $(X, \|.\|)$ be an infinite-dimensional separable Hilbert space and G its orthornormal basis. Then for every $\delta \in (0,1]$, $A_\delta(G)$ is not convex and for every $r > 0$, $B_r(\|.\|) \not\subseteq A_\delta(G)$.*

Proof. As $G \subseteq A_\delta(G)$, if $A_\delta(G)$ were convex, we would get $A_\delta(G) = \operatorname{conv} G$, which contradicts Proposition 1. Inspection of its proof shows that for each $r > 0$, there exists some $h_k \in \operatorname{conv} G$ with $\|h_k\| = \frac{1}{\sqrt{2k}} < r$ and for each $\delta_k > 0$ with $h_k \in A_{\delta_k}$, $\delta_k < \delta$. □

For a subset M of a normed linear space $(X, \|.\|)$ such that $X = \operatorname{span} M$ and a positive real number ε, define the *ε-interior* of M in $(X, \|.\|)$ as

$$I_\varepsilon(M) = \{f \in M \,|\, (\forall h \in X)\,(\|h\| \leq \varepsilon \Rightarrow f + h \in M)\}.$$

It is easy to see that $\cup_{\varepsilon>0} I_\varepsilon(M) = I_{Aff}(M)$.

The next proposition is an extension of Lavretsky's result [9, Theorem 2] (here stated as Theorem 3) on a relationship of affine interiors of certain subsets of finite-dimensional Hilbert spaces and sets $F_\delta(G)$.

Proposition 2. *Let $(X, \|\cdot\|)$ be a Hilbert space, G its bounded subset with $\operatorname{span} G = X$ and ε, $\delta > 0$ be such that $\varepsilon = 2 s_G \delta$. Then $I_\varepsilon(\operatorname{conv} G) \subseteq F_\delta(G)$.*

Proof. To prove the statement by contradiction assume that there exist $f \in I_\varepsilon(\operatorname{conv} G)$ such that $f \notin F_\delta(G)$. Then there exists $h \in \operatorname{conv} G$ such that for all $g \in G$, $(f-g)\cdot(f-h) > -\delta\|f-g\|\ \|f-h\|$. Hence $h \neq f$ and we have

$$(g-f)\cdot\frac{f-h}{\|f-h\|} < \delta\|f-g\| \leq 2 s_G \delta = \varepsilon$$

for all $g \in G$. So for all $g = \sum_{i=1}^m a_i g_i \in \operatorname{conv} G$, we get

$$\left(\sum_{i=1}^m a_i g_i - f\right)\cdot\frac{f-h}{\|f-h\|} = \sum_{i=1}^m a_i (g_i - f)\cdot\frac{f-h}{\|f-h\|} < \varepsilon \sum_{i=1}^m a_i = \varepsilon.$$

Thus for any $g \in \operatorname{conv} G$

$$(g-f)\cdot\frac{f-h}{\|f-h\|} < \varepsilon. \tag{9}$$

Since $f \in I_\varepsilon(\operatorname{conv} G)$, $f + \varepsilon \frac{f-h}{\|f-h\|} \in \operatorname{conv} G$. Setting $g = f + \varepsilon \frac{f-h}{\|f-h\|}$, we get

$$(g-f) \cdot \frac{f-h}{\|f-h\|} = \left(f + \varepsilon \frac{f-h}{\|f-h\|} - f \right) \cdot \frac{f-h}{\|f-h\|} = \varepsilon,$$

which contradicts (9). □

Corollary 3. *Let $(X, \|\cdot\|)$ be a finite-dimensional Hilbert space and G its bounded subset with $\operatorname{span} G = X$. Then there exist $r > 0$ and $\delta \in (0,1]$ such that for all $f \in B_r(\|.\|)$, $\|f - \operatorname{conv}_n G\|^2 \leq (1-\delta^2)^{n-1} \left(s_G^2 - \|f\|^2\right)$.*

Proof. Any symmetric convex set is the unit ball of the norm defined by its Minkowski functional, so in particular $\operatorname{conv}(G \cup -G)$ is the unit ball of a norm on X. As in a finite-dimensional Hilbert space all norms are equivalent, there exists some $c > 0$ such that $B_c(\|.\|) \subseteq \operatorname{conv}(G \cup -G)$. So for every $r > 0$ and $\varepsilon > 0$ such that $r + \varepsilon \leq c$, $B_r(\|.\|) \subseteq I_\varepsilon(\operatorname{conv}(G \cup -G))$. Thus the statement follows from Proposition 2 and Theorem 2. □

5 Discussion

We have described sets of functions, for which errors in approximation by sets $\operatorname{conv}_n G$ converge to zero geometrically fast. Our results show that for all functions in the convex hull of G, the Maurey-Jones-Barron's upper bound $\frac{1}{n}\left(s_G^2 - \|f\|^2\right)$ can be improved by replacing the term $\frac{1}{n}$ with $(1-\delta_f^2)^{n-1}$, where $\delta_f \in (0,1]$ is specific for each function to be approximated. Geometric upper bounds on rates of approximation by computational models of the variable-basis type provide a theoretical insight into capabilities of neural networks. Such estimates can be combined with characterizations of functions belonging the corresponding convex hulls (see, e.g. [11]). The incremental procedure in Fig. 1, which was obtained from an inspection of the proof of Theorem 4, may represent a first step towards a design of learning algorithms for networks with a low model complexity.

Acknowledgements

V. K. was partially supported by Project 1ET100300517 of the program "Information Society" of the National Research Program of the Czech Republic and the Institutional Research Plan AV0Z10300504. M. S. was partially supported by PRIN grants from the Italian Ministry for University and Research, projects "Models and Algorithms for Robust Network Optimization" and "New Algorithms and Methodologies for the Approximate Solution of Nonlinear Functional Optimization Problems in a Stochastic Environment". Collaboration between V. K. and M. S. was partially supported by the 2007-2009 Scientific Agreement among University of Genova, National Research Council of Italy, and Academy of Sciences of the Czech Republic, Project "Learning from data by neural networks and kernel methods as an approximate optimization".

References

1. Pinkus, A.: N-widths in Approximation Theory. Springer, New York (1986)
2. Pisier, G.: Remarques sur un résultat non publié de B. Maurey. In: Séminaire d'Analyse Fonctionnelle, vol. I(12), École Polytechnique, Centre de Mathématiques, Palaiseau (1980–1981)
3. Jones, L.K.: A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. Annals of Statistics 20, 608–613 (1992)
4. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans. on Information Theory 39, 930–945 (1993)
5. Kůrková, V., Savický, P., Hlaváčková, K.: Representations and rates of approximation of real–valued Boolean functions by neural networks. Neural Networks 11, 651–659 (1998)
6. Kůrková, V., Sanguineti, M.: Bounds on rates of variable-basis and neural-network approximation. IEEE Trans. on Information Theory 47, 2659–2665 (2001)
7. Makovoz, Y.: Random approximants and neural networks. J. of Approximation Theory 85, 98–109 (1996)
8. Kůrková, V., Sanguineti, M.: Estimates of covering numbers of convex sets with slowly decaying orthogonal subsets. Discrete Applied Mathematics 155, 1930–1942 (2007)
9. Lavretsky, E.: On the geometric convergence of neural approximations. IEEE Trans. on Neural Networks 13, 274–282 (2002)
10. Kůrková, V.: Dimension-independent rates of approximation by neural networks. In Warwick, K., Kárný, M. (eds.) Computer-Intensive Methods in Control and Signal Processing. The Curse of Dimensionality. Birkhäuser, Boston, pp. 261–270 (1997)
11. Kůrková, V.: High-dimensional approximation and optimization by neural networks. In: Suykens, J., Horváth, G., Basu, S., Micchelli, C., Vandewalle, J. (eds.) Advances in Learning Theory: Methods, Models and Applications, Ch. 4., pp. 69–88. IOS Press, Amsterdam (2003)

A Sensitive Metaheuristic for Solving a Large Optimization Problem

Camelia-M. Pintea[1], Camelia Chira[1], D. Dumitrescu[1], and Petrica C. Pop[2]

[1] Babeş-Bolyai University, Cluj-Napoca 400084, Romania
[2] North University, Baia-Mare 430122, Romania
{cmpintea,cchira,ddumitr}@cs.ubbcluj.ro, pop_petrica@yahoo.com

Abstract. A metaheuristic for solving complex problems is proposed. The introduced *Sensitive Robot Metaheuristic (SRM)* is based on the *Ant Colony System* optimization technique. The new model relies on the reaction of virtual sensitive robots to different stigmergic variables. Each robot is endowed with a particular stigmergic sensitivity level ensuring a good balance between search diversification and intensification. Comparative tests are performed on large-scale NP-hard robotic travel problems. These tests illustrate the effectiveness and robustness of the proposed metaheuristic.

1 Introduction

As real world problems demand increasing autonomies and more complex artificial systems, engineers often look to nature for a possible model. Social insects with their limited structures and communication capabilities coordinate to construct large and complex nests [2]. They provide potential powerful models for collective robotic systems.

Stigmergic nest-building techniques used by many types of social insects are an example of adaptive behavior that can be very useful in coping with complexity and solving large scale computational problems. The construction of metaheuristics with several classes of specialized robots has the potential to produce solutions to actual NP-hard problems.

The aim of this paper is to provide an effective metaheuristic to address complex problems. The introduced technique is called *Sensitive Robot Metaheuristic (SRM)* and combines elements of *Ant Colony System (ACS)* [3] and autonomous mobile robots. The model relies on a collection of robots each of them being endowed with a stigmergic sensitivity level that allows it to detect and react to different stigmergic variables. *SRM* is applied to solve a robotic travel problem that refers to the minimization of the drilling operations time on printed circuit boards.

Extensive computational experiments and comparison of the proposed *SRM* with *Nearest Neighbor (NN)*, a composite heuristic [11], a *Random Key Genetic Algorithm (RKGA)* [12] and *Ant Colony System (ACS)* for *GTSP* [10] indicate the potential of the introduced metaheuristic.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 551–559, 2008.

2 Sensitive Stigmergic Robots

The proposed metaheuristic is called *Sensitive Robot Metaheuristic (SRM)* and aims to address complex NP-hard problems. This section introduces the concept of stigmergic autonomous robot and describes the proposed metaheuristic.

2.1 Stigmergy and Autonomous Robots

The proposed metaheuristic combines the concepts of stigmergic communication and autonomous robot search. Stigmergy occurs when an action of an insect is determined or influenced by the consequences of the previous action of another insect [2].

Stigmergy [6] provides a general mechanism that relates individual and colony-level behaviors: individual behavior modifies the environment, which in turn modifies the behavior of other individuals. The behavior-based approach to design intelligent systems has produced promising results in a wide variety of areas including military applications, mining, space exploration, agriculture, factory automation, service industries, waste management, health care and disaster intervention. Autonomous robots can accomplish real-world tasks without being told exactly how.

Researchers try to make the coupling between perception and action as direct as possible. This aim remains the distinguishing characteristic of behavior-based robotics. The proposed *SRM* technique attempts to address this goal in an intelligent stigmergic manner.

2.2 Sensitive Robots

A stigmergic robot action is determined by the environmental modifications caused by prior actions of other robots. Quantitative stigmergy regards stimulus as a continuous variable. The value of such a variable modulates the intensity or probability of future actions. Qualitative stigmergy [2,13,14] involves discrete stimulus. In this case the action is not modulated but switched to a different action, [2,13,14].

Qualitative stigmergic mechanism better suits our aims since robot stigmergic communication does not rely on chemical deposition. The robot communication relies on local environmental modifications that can trigger specific actions. "Micro-rules" define action-stimuli pairs for a robot. The set of all micro-rules used by a homogeneous group of stigmergic robots defines their behavioral repertoire and determines the type of structure the robots will create [2,13,14].

Within the proposed model, *Sensitive robots* refers to artificial entities with a *Stigmergic Sensitivity Level (SSL)* which is expressed by a real number in the unit interval [0, 1].

Robots with small *SSL* values are highly independent and can be considered environment explorers. They have the potential to autonomously discover new

promising regions of the search space. Therefore, search diversification can be sustained.

Robots with high *SSL* values are able to intensively exploit the promising search regions already identified. In this case the robot behavior emphasizes search intensification. The *SSL* value can increase or decrease according to the search space topology encoded in the robot experience.

3 Sensitive Robot Metaheuristic for Drilling Problem

The proposed *Sensitive Robot Metaheuristic (SRM)* can be implemented using two teams of sensitive robots. Robots of the first team have small *SSL* values. These sensitive-explorer robots are called *small SSL-robots (sSSL)* and can sustain search diversification. Robots of the second team have high *SSL* values. These sensitive-exploiter robots called *high SSL-robots (hSSL)* intensively exploit promising search regions already identified by the first team.

SRM is applied for solving a robotic travel problem called drilling problem. This problem can be viewed as an instance of the *Generalized Traveling Salesman Problem.*

3.1 Generalized Traveling Salesman Problem

Let $G = (V, E)$ be an n-node undirected graph whose edges are associated with non-negative costs. We assume that G is a complete graph. Let $V_1, ..., V_p$ be a partition of V into p subsets called *clusters*. The cost of an edge $(i, j) \in E$ is c_{ij}.

The *generalized traveling salesman problem* refers to finding a minimum-cost tour H spanning a subset of nodes such that H contains exactly one node from each cluster V_i, $i \in \{1, ..., p\}$. The problem involves two related decisions: choosing a node subset $S \subseteq V$, such that $|S \cap V_k| = 1$, for all $k = 1, ..., p$ and finding a minimum cost Hamiltonian tour in S.

3.2 Drilling Problem

The considered robotic travel problem refers to minimizing the drilling operations time on a large printed circuit boards (Figure 1).

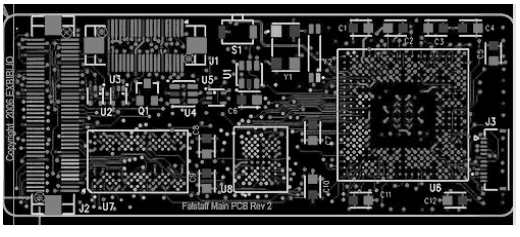


Fig. 1. A Printed Circuit Board

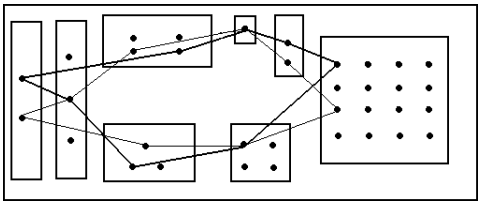


Fig. 2. A schematic representation of the drilling problem on a Printed Circuit Board

The process of manufacturing the printed circuit board (PCB) is difficult and complex. Each layer of the PCB requires the ability of one layer to connect to another layer achieved through drilling small holes. These holes require precision and are done with the use of an automated drilling machine driven by computer programs.

The large drilling problem is a particular class of the generalized traveling salesman problem involving a large graph and finding the minimal tour for drilling on a large-scale printed circuit board (Figure 2.).

3.3 SRM for Solving a Large Drilling Problem

Initially the robots are placed randomly in the search space. In each iteration a robot moves to a new node and the parameters controlling the algorithm are updated.

A robot chooses the next move with a probability based on the distance to the candidate node and the stigmergic intensity on the connecting edge. Each time unit evaporation takes place. This is to stop the stigmergic intensity increasing unboundedly.

In order to prevent robots visiting a cluster twice in the same tour a *tabu list* [3] is maintained. The stigmergic value of an edge is denoted by τ and the visibility value is η.

Let us consider ${J^k}_i$ to be the unvisited successors of node i by robot k and $u \in {J^k}_i$. The *sSSL* robots probabilistically choose the next node. Let i be the current robot position (the current node). Similarly to the *ACS* technique [3], the probability of choosing u as the next node is given by:

$$ {p^k}_{iu}(t) = \frac{[\tau_{iu}(t)][\eta_{iu}(t)]^\beta}{\Sigma_{o \in {J^k}_i}[\tau_{io}(t)][\eta_{io}(t)]^\beta}, \tag{1} $$

where β is a positive parameter, $\tau_{iu}(t)$ is the stigmergic intensity and $\eta_{iu}(t)$ is the inverse of the distance on edge (i, u) at moment t.

The membership of robots to one of the two teams is modulated by a random variable uniformly distributed over $[0, 1]$. Let q be a realization of this random variable and q_0 a constant, $0 \leq q_0 \leq 1$. The *sSSL* robots are characterized by the inequality $q > q_0$ while for the *hSSL* robots $q \leq q_0$ holds.

A *hSSL-robot* uses the information supplied by the *sSSL* robots. *hSSL* robots choose the new node j in a deterministic manner according to the following rule:

$$j = argmax_{u \in J_i^k}\{\tau_{iu}(t)[\eta_{iu}(t)]^{\beta}\}, \tag{2}$$

where the value of β determines the relative importance of stigmergy versus heuristic information.

The trail stigmergic intensity is updated using the local stigmergic correction rule:

$$\tau_{ij}(t+1) = q_0^2\tau_{ij}(t) + (1-q_0)^2 \cdot \tau_0. \tag{3}$$

Only the elitist robot that generates the best intermediate solution is allowed to *globally* update the stigmergic value. The elitist robot can take advantage of global knowledge of the best tour found to date and reinforce this tour in order to focus future searches more effectively. The global updating rule is:

$$\tau_{ij}(t+1) = q_0^2\tau_{ij}(t) + (1-q_0)^2 \cdot \Delta\tau_{ij}(t), \tag{4}$$

where $\Delta\tau_{ij}(t)$ is the inverse value of the best tour length. In the updating rules, q_0 is reinterpreted as the evaporation rate.

The robots work one by one in each step. A run of the algorithm returns the shortest tour found. Termination criteria is given by a given number of iterations (N_{iter}).

The description of the *Sensitive Robot Metaheuristic* for solving the drilling problem is depicted in Algorithm 1.

```
Algorithm 1. Sensitive Robot Algorithm
Begin
  Set parameters, initialize stigmergic values of the trails;
  For k=1 to m do
    Place robot k on a randomly chosen node
    from a randomly chosen cluster;
  For i=1  to Niter do
      Each robot incrementally builds a solution
      based on the autonomous search sensitivity;
      sSSL robots are characterized by the inequality q>q0
      while the others are considered hSSL robots;
      The sSSL robots probabilistically choose
      the next node using (1)
      A hSSL-robot uses the information supplied by
      the sSSL robots to find the new node j using (2);
      A local stigmergic updating rule (3);
    A global updating rule is applied by the elitist robot (4);
   Endfor
End
```

Let us consider n to be the number of nodes, e the number of edges and p the number of clusters in the input graph, m the number of robots and NC the

number of cycles. The complexity of this algorithm leads to $O(p \cdot n \cdot m \cdot NC)$, [3]. For an exact algorithm obtained by trying all the $(p-1)!$ possible cluster sequences [10], the complexity is $O((p-1)!(ne + nlogn))$.

4 Numerical Experiments

The validation of *SRM* concerns the minimization of the drilling operations time on printed circuit boards.

The numerical experiments are based on the *TSP* library [1] that provides optimum objective values for each problem. The drill problems with Euclidean distances have been considered.

In the *SRM* algorithm the values of the parameters were chosen as follows: $\beta = 5$, τ_0=0.01, $q_0 = 0.9$. The parameters were chosen based on [3,10]. The total number of robots considered is 25. The sensitivity level q for *hSSL* robots is considered to be distributed in the interval $(q_0, 1)$, while *sSSL* robots have a sensitivity level in the interval $(0, q_0)$.

The solutions of all algorithms represent the average of five consecutive runs for each problem. Termination criteria are given by the maximum of 200 trials and 100 tours.

Table 1 illustrates *SRM* results after five consecutive runs of the algorithm. To divide the set of nodes into subsets we used the procedure proposed in [4]. This procedure sets the number of clusters $nc = [n/5]$, identifies the nc farthest nodes from each other (called centers) and assigns each remaining node to its nearest center.

The program is implemented in java and run on a AMD Athlon 2600+, 333Mhz with 2GB memory.

Table 1. Sensitive Robotic Metaheuristic results for five runs. The table shows the values of the reported optimum values [1], the minimum, maximum and the mean value of *SRM* after five runs of the algorithm. The number of the optimum values within the specified number of runs are also shown.

Drilling Problem	Reported optimum	No. optimum	Mean value	Minimum value	Maximum value
32U159	22664	5	**22664**	**22664**	**22664**
40D198	10557	5	**10557**	**10557**	**10557**
84FL417	9651	1	9654.4	**9651**	9657
89PCB442	21657	2	21659.6	**21657**	21662

To evaluate the performance of the proposed algorithm, the *SRM* has been compared to *Nearest Neighbor (NN)* [11], a composite heuristic (GI^3) [11], a Random Key Genetic Algorithm ($RKGA$) [12] and Ant Colony System (ACS) for *GTSP* [10]. The results of these algorithms for the average of five consecutive runs for each problem have been considered.

In the *Nearest Neighbor* algorithm the rule is always to go next to the nearest as-yet-unvisited location. The corresponding tour traverses the nodes in the constructed order. The composite heuristic GI^3 has three phases: the construction of an initial partial solution, the insertion of a node from each non-visited node-subset, and a solution improvement phase, [11].

The *Random Key Genetic Algorithm* combines a genetic algorithm with a local tour improvement heuristic. Solutions are encoded using random keys, which circumvent the feasibility problems encountered when using traditional GA encodings, [12]. The *Ant Colony System* for *GTSP* has been introduced to solve the *Generalized Traveling Salesman Problem (GTSP)* [10]. The basic idea of *ACS* for *GTSP* is that of simulating the behavior of a set of agents that cooperate to solve a problem by means of simple communications.

The comparative results are shown in Table 2.

Table 2. *Sensitive Robotic Metaheuristic (SRM)* versus other algorithms: *Nearest Neighbor (NN)*, a composite heuristic GI^3, [11], a Random Key Genetic Algorithm (RKGA), [12] and Ant Colony System (ACS) for *GTSP*. (Mean values).

Drilling Problem	Reported Optimum	*NN*	GI^3	*RKGA*	*ACS*	*SRM*
32U159	22664	26869	23254	**22664**	22729.2	**22664**
40D198	10557	12038	10620	**10557**	10575.2	**10557**
84FL417	9651	10553	9697	9656	9766.2	**9654.4**
89PCB442	21657	26756	22936	22026	22137.8	**21659.6**

A statistical analysis is performed in the following. The Expected Utility Approach [5] technique has been employed to determine the most accurate heuristic. Let x be the percentage deviation of the heuristic solution and the best known solution of a particular heuristic on a given problem:

$$x = \frac{heuristicsolution - bestknownsolution}{bestknownsolution} \times 100.$$

The expected utility function can be: $\gamma - \beta(1 - \overline{b}t)^{-\overline{c}}$, where $\gamma = 500$, $\beta = 100$ and $t = 0.05$. $\overline{b}$ and $\overline{c}$ are the estimated parameters of the Gamma function. Because four problems have been used for testing, the following notations are used for Table 3: $\overline{x} = \frac{1}{4}\sum_{j=1}^{4} x_j$, $s^2 = \frac{1}{4}\sum_{j=1}^{4}(x_j - \overline{x})^2$, $\overline{b} = \frac{s^2}{\overline{x}}$, $\overline{c} = (\frac{\overline{x}}{s})^2$.

The last column provides the rank 1 to 5 of the entries. As indicated in Table 3, *SRM* has Rank 1 being the most accurate algorithm within the compared algorithms.

The compared results from Table 2 indicated that the newly introduced *SRM* algorithm outperforms the other heuristics considered.

The new model has to be improved in terms of execution time. Potential improvements regard the parameter values or an efficient combination with other

Table 3. Statistical analysis. Calculations for the expected utility function for the compared heuristics.

Heuristic	$\overline{x}$	s^2	$\overline{b}$	$\overline{c}$	$\gamma - \beta(1-\overline{b}t)^{-\overline{c}}$	Rank
NN	16.5	31.25	1.8939	8.7122	262.0747	5
GI^3	2.3956	4.8206	2.0123	1.1905	386.5441	4
$RKGA$	0.4385	0.5558	1.2675	0.3459	397.7087	2
ACS	0.97	0.6783	0.6993	1.3871	394.9359	3
SRM	0.01	0.0001	0.01	1.0000	399.9499	1

algorithms. Another way to improve the algorithm is making the robots working full parallel in inner loop of the algorithm.

In the future we will perform also other numerical experiments to assess the performance and speed of the new algorithm including Lin-Kernighan algorithm and its variants(e.g. iterated Lin-Kernighan [8,9] and Helsgauns variant [7]).

5 Conclusions

A bio-inspired robot-based model for complex travel robotic problems is proposed and tested, each robot with a stigmergic sensitivity level that facilitates the exploration (by low-sensitive robots) as well as exploitation (by high-sensitive robots) of the search space.

The computational results of the proposed *Sensitive Robot Metaheuristic (SRM)* for the drilling problem, are good and competitive in both solution quality and computational time with the existing heuristics from the literature. This result is furthermore certified by the statistical analysis performed based on calculations for the expected utility function for the compared heuristics.

References

1. Bixby, B., Reinelt, G.: (1995), http://nhse.cs.rice.edu/softlib/catalog/tsplib.html
2. Bonabeau, E., Dorigo, M., Tehraulaz, G.: Swarm intelligence from natural to artificial systems. Oxford University Press, Oxford, UK (1999)
3. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem. IEEE Trans. Evol. Comp. 1, 53–66 (1997)
4. Fischetti, M., Gonzales, J.J.S., Toth, P.: A Branch-and-Cut Algorithm for the Symmetric Generalized Travelling Salesman Problem. Oper. Res. 45(3), 378–394 (1997)
5. Golden, B.L., Assad, A.A.: A decision-theoretic framework for comparing heuristics. European J. of Oper. Res. 18, 167–171 (1984)
6. Grassé, P.-P.: La Reconstruction du Nid et Les Coordinations Interindividuelles Chez Bellicositermes Natalensis et Cubitermes sp. La Thorie de la Stigmergie: Essai dinterpretation du Comportement des Termites Constructeurs. Insect Soc. 6, 41–80 (1959)

7. Helsgaun, K.: An effective implementation of the lin-kernighan TSP heuristic. European Journal of Operations Research 126, 106–130 (2000)
8. Johnson, D.S., McGeoch, L.A.: Local Search in Combinatorial Optimization, chapter The Traveling Salesman Problem: A Case Study in Local Optimization, pp. 215–310. John Wiley & Sons, New York (1997)
9. Johnson, D.S., McGeoch, L.A.: The Traveling Salesman Problem and its Variations, chapter Experimental Analysis of Heuristics for the STSP, pp. 369–443. Kluwer Academic Publishers, Dordrecht (2002)
10. Pintea, C-M., Pop, C.P., Chira, C.: The Generalized Traveling Salesman Problem solved with Ant Algorithms. J.UCS (in press, 2007)
11. Renaud, J., Boctor, F.F.: An efficient composite heuristic for the Symmetric Generalized Traveling Salesman Problem. Euro. J. Oper. Res. 108(3), 571–584 (1998)
12. Snyder, L.V., Daskin, M.S.: A Random-Key Genetic Algorithm for the Generalized Traveling Salesman Problem. INFORMS, San Antonio, TX (2000)
13. Theraulaz, G., Bonabeau, E.: A brief history of stigmergy. Artificial Life 5(2), 97–116 (1999)
14. White, T.: Expert Assessment of Stigmergy: A Report for the Department of National Defence,
http://www.scs.carleton.ca/~arpwhite/stigmergy-report.pdf

Domain Name System as a Memory and Communication Medium

Dušan Bernát

Institute of Computer Systems and Networks,
Faculty of Informatics and Information Technology, STU Bratislava,
Ilkovičova 3, 842 16 Bratislava, Slovakia
dusan.bernat@fiit.stuba.sk

Abstract. This article describes the way how some amount of information can be stored into DNS, particularly in the cache of DNS server. Then it can be retrieved back, possibly by another host in the network. Based on this principle we can construct a communication channel, hidden in the usual traffic, or a memory medium. Considering this kind of media, some basic characteristics and limits, like capacity, transfer speed, error rate, persistence of information, etc., are discussed here. Simple algorithm deciding whether a bit in the memory has been set or not was proposed and implemented. Its performance and optimal setting was examined. The results show that under some circumstances error rates about 0.003, when retrieving the information, can be achieved.

Keywords: DNS, cache, security, covert channel, memory.

1 Introduction

Idea of communication between processes sharing common cache has appeared in [1]. It was based on measuring access time of the processor cache and the difference for cached and uncached addresses. This approach can be generalised to other types of shared cache (not only hardware cache of processor). Reference to DNS cache appeared in [2] and [3]. Based on these ideas we have proposed and implemented algorithm, which proved the ability to transfer some amount of information between processes on different subnetworks. It is based entirely on the response time examination and does not consider content of the DNS response. We have also analysed properties of the algorithm and find the optimal setting of algorithm parameters minimising error rate. This value strongly depends on server type and configuration as well as variety of network conditions. As a result, dependency of error rate on the parameter, perhaps drawn in a graph, is an unique fingerprint of each DNS server. Nevertheless it was not our intention to provide complete analyses of all possible factors. The aim is just to show that the communication in this way is possible and that the described algorithm is useful for examining some of the DNS server properties.

First a brief summary of basic DNS properties and principles is given, as well as a description of memory model. Details about the algorithm and properties

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 560–571, 2008.

of resulting communication media follow. Finally we provide results based on measurements for a couple of name servers and its respective error rates.

1.1 Domain Name System

As people usually prefer using some symbolic name rather than numeric address, we need some mechanism for translating between these two representations. In the Internet, Domain Name System (DNS) serves for this purpose. It is extensible, hierarchical, distributed system which provides host or domain name to address mapping as well as some other functions (e.g. denotes mail exchanger for the domain).

A domain name consists of several parts delimited by the dot. Not only hierarchy of names may reflect organisational structure of companies or institutions, but also the address name space is distributed among corresponding DNS servers in hierarchical manner. Consequently, several name servers may be involved in resolution of a single name. In order to limit the traffic among name servers and improve the response time for DNS requests, caching in DNS is widely deployed [4].

If the required record is stored in the cache (or if the server is authoritative for given name), server responds immediately. Otherwise, some other server has to be consulted (regardless of iterative or recursive approach). Each record in the cache has specified a *time to live* (TTL). Records older than its TTL must not be used again. They are released from the cache and must be obtained from other servers. This provides the only coherence mechanism for DNS cache. If someone changes record in the authoritative server, its new value is not visible until cached record expires. Thus the TTL value makes a trade off between performance and update period. Although `RFC 1034` says *"Access to information is more critical than instantaneous updates or guarantees of consistency"* [4], requirements of mobile applications and dynamic DNS tend to decreases TTL value of corresponding records significantly.

There is also substantial amount of requests for non-existent domain names, resulting in *name error* or *no data error*. Caching these negative answers, i.e. the knowledge that a name does not exist, can decrease the traffic and reduce the response time too. Hence `RFC 2308` proposed the negative caching of DNS queries not to be optional [6]. This mechanism and the fact that cache is shared among all clients allowed to access the server, provide basis for construction of a potentially hidden communication channel.

1.2 Memory Model

Storing of information is fundamental to all computing systems. Memory medium (M) can be usually viewed as a function[1], i.e. a set of pairs, where the first part is an address and the second one is a value bound to this address $M \equiv \{(a_i, v_i); \forall i \in I\}$. Perhaps there are some exceptions, e.g. auto-associative memory, which answers the data based upon incomplete input value, without explicitly specified

[1] Moreover, it is a *sequence*, as the address space (i.e. the domain) is ordered.

address. We restrict further considerations to practical cases of finite memory, where $|I| \in \mathbb{N}$. Moreover the memory address space $A \equiv (\{a_i; \forall i \in I\}, <)$, represented by ordered set of all addresses, as well as the value range $V \equiv \{v_i; \forall i \in I\}$, is usually constituted by a continuous integer region, starting from zero with the length equal to some power of two. However, in general, we can think of address space and value range as arbitrary sets.

To operate the memory we need some methods to store and read data values from particular address. We denote this functions

$$\texttt{read}(a) \rightarrow v_i, i : a_i = a,$$

and

$$\texttt{write}(a, v), v_i := v, i : a_i = a.$$

If the memory medium is shared among processes and the read and write operations can be performed by different ones, exchange of information, i.e. communication, is possible. It would be nice if we always can read the same value which we have written before under particular address. But in reality, we can occasionally observe that retrieved value doesn't match the original one. If the number of mismatched write and read operations performed on certain address during N read-write cycles is denoted e, then we say that error rate E is

$$E = \frac{e}{N}.$$

We are going to consider some other characteristics of memory and its physical implementation later.

1.3 Covert Channel

Communication in an unusual way, using a shared medium which was not intended for this purpose, possibly breaking security policy, is typical characteristics of a covert channel. It is not only hiding the data to be transferred, but also the fact that there is any communication between the two processes. DNS cache of a server, willing to accept connections from the outside world, provides appropriate shared medium. If one process can force the record of particular domain name to be loaded into the server cache and another process, possibly running on another host in the network, can find this out, hidden communication is possible.

Any caching name server can be used, without the need to change its configuration or take control over it. Communicating processes sends only regular DNS requests to specified server. Since such a communication is realised indirectly via some server, communicating processes need not to know about their network (IP) addresses. Rather the domain name is used in the role of address, where we can store one bit of information. The record for this name is either cached or not. Nevertheless, the two communicating processes must be aware of common address space, i.e. the sequence of domain names, beforehand. So they have to exchange the information about which addresses will be used throughout the transfer.

It is also possible to create a tunnel by inserting data into the fields of DNS protocol request and response packets. But this is outside the scope of this article.

2 Implementation of Memory Algorithm

2.1 Principle

To create a memory medium, we need a realisation of our `read` and `write` methods. If we make a standard DNS query to a server which is not authoritative for requested record, it must either search for corresponding record on another server (recursive) or replies with a name of server which the client should contact instead (iterative). Anyway, communication with other servers is required. We can measure the time interval between sending the query and obtaining the response. In typical occasion this may range from milliseconds to seconds. To reduce the time needed to serve subsequent requests for the same name, caching name server can store the record it the cache. Now if we send the same request again, response time may be significantly, and measurably, lower.

Implementation of `write` operation is straightforward. It simply sends DNS request when the bit has to be set, and doesn't send any request for the bit which is left unset. The `read` operation always performs two consecutive requests. Then it has to decide whether the difference in response times (speedup) is caused by storing the record into the cache meanwhile. Otherwise it is caused by a random delay while both replies were served from the cache (possibly due to previous `write`). We can agree that a bit has been set, if the two times are similar, i.e. difference is smaller than some sufficiently small constant value $|t_1 - t_2| < D$. Alternatively, we can check how many times is the second response faster than the first one. This is measured by the ratio $t_2 \cdot M < t_1$. We have chosen the second approach.

Note that there are other ways, some of them perhaps more reliable, to find out whether record has been cached. For more information please see [2], [3]. Algorithm we proposed allows us to investigate some interesting properties of DNS system, thought it is not the best design for real data transfers.

2.2 General Properties of Memory

Capacity. Memory capacity in our model is equal to number of elements in the address space multiplied by amount of information stored in each value. Thus we have

$$C_M = |A| \cdot \log_2 |V|.$$

Here we have the set of addresses equivalent to all legitimate domain names. It is described in `RFC 1034`, chapter *3.1 Name space specifications and terminology* [4]. Domain names are *de facto* case insensitive, restricted to 255 `ASCII` characters in length and neighbouring nodes in hierarchy cannot be identical. Some other limits and details may be found in `RFC 1035` [5]. Anyway, the address space seems to be for all practical purposes inexhaustible[2], regardless of the fact that each address points only to a single bit. On the other hand, the address itself needs several bits to be encoded as we need to communicate the address

[2] Number of possible strings with length 255 would give a rough approximation.

space to the other processes before we can transfer the data. This may seem ineffective, but fortunately we can encode large continuous blocks of addresses very efficiently.

The total range of *address space* is limited by several factors. If we want our communication not to be disturbed by regular DNS traffic, we must chose addresses which are not likely to be used on particular DNS server. Furthermore, if we decide to use negative caching [6], address space is restricted to nonexistent domain names. And if we choose to use reverse records only[3], then our address space shrinks rapidly to the set of legitimate IP addresses. Note that there is a lack of free IPv4 addresses in the past years.

From the practical viewpoint, the *capacity* is limited by the amount of memory which is the server willing to allocate for cache. Naturally, this is implementation dependent. However, the servers usually provide space for at least few tens of thousands of records. Moreover, our address space can be split and deployed on several DNS servers. Thus we can increase available space and in the same time make the detection harder.

Persistence of Information. Each record stored in DNS cache has attached the TTL value, which is a 32-bit integer[4] [5]. After specified amount of seconds, record expires and must not be used again [4]. As we have already mentioned, the specification of DNS protocol prefers accessibility over consistency of information. This yields relatively high values of TTL (in range of days or weeks). On the other hand, when we make a change in DNS record, we don't want to wait a long time, until all cached records expires. This is especially true for applications with mobile addresses or temporal names (e.g. assigned by DHCP). Thus we can usually find a default server TTL ranging from 1 or 3 hours to several days. For changing records the value is about 10 minutes or even less, minute or two. Stable, well known and long running names may have their records TTL as high as few weeks. Default TTL value on the server applies also to negative responses stored in the cache. Actual value for given server can be measured or simply read from DNS response [7].

Of course, this has several implications for our memory medium. If we want to store the information for longer time than is the minimal TTL value, it has to be written again, just when the previous records expired. To accomplish this some synchronisation mechanism would be useful, as it follows that reading the information when it is being renewed, or just before it is renewed, yields invalid results. This is a nice analogy to DRAM (Dynamic Random Access Memory) technology widely used in computer operation memory, which needs a regular read or refresh cycle (during which each memory location is addressed) in order to retain the information.

Another point is that information in the cache is *destroyed* while it is read. After `read` operation completes on a subset of address space, all records corresponding to names within this subset are present in the cache. So subsequent

[3] Names within the domain `.IN-ADDR.ARPA`.

[4] Maximum TTL value spans over 60 years. Zero value means that record should not be cached.

`read` cannot recognise what was written into the cache. This feature resembles quantum mechanical computing systems. In some applications (e.g. security or authentication) the *read-once* behaviour may be quite useful. However, if we could look inside the server carefully, we would see that age value for original records stored during former `write` operation doesn't match the value for records cached during later `read`. After the original records expire, the information appears again, but in inverted manner.

The `write` operation behaves similarly. It destroys (sets to one) data in all processed addresses. Although this *read-once*/*write-once* behaviour may be impractical for general purpose memory it is quite usual when we consider a communication channel. A bit just placed into the medium (whether it is wire, optics or something else) cannot be reverted. But we can achieve this using appropriate higher level protocol which will resend new value if needed.

If we need to store the information in DNS cache for longer time, we have to choose the server with high value of default TTL, or to read or refresh it regularly. The record stored in cache may also be released before expiration if the server is heavily loaded (or flooded) and needs space in the cache for new records. Server restart destroys all of its content as well.

Access Time. It always takes some time to store or retrieve information from the memory. Here it depends mainly on the time of DNS request processing. If the average response time for uncached domain names is t_u seconds and the probability of occurrence of symbol `1` is p, than we will need $T_w = N \cdot t_u \cdot p$ seconds to write a message N bits long, assuming that all of N operations are executed sequentially.

Although several DNS requests per one bit will be always needed, the throughput can be increased by launching several DNS requests in parallel by independent threads. Thus we can eliminate the process blocking while waiting for particular response. Moreover, we would not need to block on waiting for the result of `write` request, if for some reason the error status doesn't matter. Thought in principle we can access all addresses of our memory in one instant (there are no dependencies imposed by DNS protocol) in reality we are always limited (in lower layers) by number of threads we can run. Also the server cannot answer at once when it is flooded with multiple requests. In the same time, this will generate more unusual communication which is more likely to be detected. This is the usual trade-of between transfer speed and visibility.

Nevertheless DNS response time per request ranges from milliseconds to seconds (depending on the server setting and network conditions) and sequential writing of $1kB$ block can take about ten minutes.

Read operation takes $T_r = N \cdot [(1-p) \cdot t_u + (1+p) \cdot t_c]$. For the optimal case $p = 0.5$, where each symbol bears one bit of information, we get $T_w = N/2 \cdot t_u$ and $T_r = N/2 \cdot (t_u + 3 \cdot t_c)$. Thus if access to cached records is M times faster, then `read` operation takes $1 + 3/M$ times longer than the `write`,

$$T_r = \left(1 + \frac{3}{M}\right) \cdot T_w.$$

If the response times of cached and uncached record t_c and t_u respectively are constant and known, it should be possible to make decision in `read` operation based only on one response time, thus reducing overall read time to $T_{r1} = N/2 \cdot (t_u + t_c)$, yielding $T_{r1} = T_w \cdot \left(1 + \frac{1}{M}\right)$. But the assumption need not to be met in general, as the response time includes erratic network delay.

Observed delay of DNS response may have several reasons. If the server which we are asking doesn't have required record for `abc.example.com`, but knows that name of another server in hierarchy which should know about it, is `ns2.dns.net`, the name of this server must be resolved, before the original resolution process can continue. This can be avoided by providing *glue*[5] for `ns2.dns.net`. However, if the glue is missing in the server configuration, resolution takes additional time to resolve name of `ns2.dns.net`. Moreover, some older implementations of DNS server (BIND 8.2.x) encounter problem resolving subsequent requests without a glue. In BIND 8.3.0, if a glue for the server `ns2.dns.net` is missing, it will be resolved, but server doesn't reply to the original request for `abc.example.com`. Just after resolver times out, it resends its original request again. This time the address of `ns2.dns.net` is present in the cache, hence request for `abc.example.com` can be completed. But resolver time out period can take several seconds. If the server configuration has more levels of gluelessness than the number of times the resolver resends the query after time out, the name could not be resolved at all, thought it is reachable. This issue was pointed out by Bernstein in [8] and also recognised by others, e.g. [9].

3 Measurements and Results

3.1 Response Analyses

We have implemented program, with asynchronous `read` and `write` operations as described in Section 2.1. It generates a random block of data D which is then scattered over the defined address space. Subsequently the block is read back, bit by bit, from the same address space and all response times are saved. Finally, the error rate is evaluated for a parameter values M ranging over some interval. This gives us the graph of $E(M)$ for particular server (assuming that whole address space is located in one server). This procedure was repeated typically 1000 times, for blocks with length N varying from one to 64 bytes on various servers.

Evaluating the relational expression

$$B_1(M, t_1, t_2) \equiv \neg(t_2 \cdot M < t_1)$$

for given value of parameter M and for all pairs of measured response times (t_1, t_2) respectively, gives the read data block

$$D_r(M) \equiv (B_1(M, t_1[i], t_2[i]))_{i=1}^{N} .$$

[5] Glue is the address of referred server, which should be stored in server config, in order to avoid additional requests.

Simply put, B_1 is true, and evaluates to 1, if measured responses corresponds to reading value 1 from memory for given value of parameter M. It evaluates to 0 otherwise. Now we can compute the error rate as follows:

$$E(M) = \frac{|D, D_r(M)|_H}{N},$$

where $|.,.|_H$ stands for Hamming distance[6].

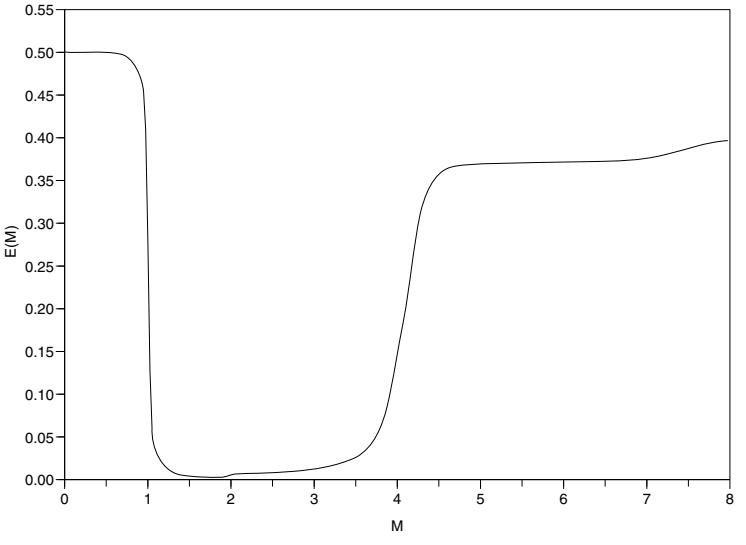


Fig. 1. Error rate as a function of algorithm parameter. Block size is 1 byte, 1000 repetitions.

We can see the graph of error rate as a function of parameter M in the figure 1. We can readily recognise three regions there. For $M < 1$ the relation B_1 is satisfied only when the second response time is greater than the first one. Clearly this makes no sense as it breaks design of the algorithm and it can randomly happen only as a result of various delays or errors in the network. Thus

$$M < 1 \Rightarrow B_1(M, t_1, t_2) \rightarrow 0, \forall(t_1, t_2),$$

as it is quite improbable that $t_2 > t_1$ unless a rare error occurs. This situation has no practical importance. Since the algorithm doesn't work, memory is unusable and we only get almost constant value 0 from it. During experiments we have written uniformly distributed random bits, so comparing to constant yields $E \rightarrow 0.5$.

For the value $M = 1$ we can observe a *phase transition* in the memory system. Error rate rapidly falls down as we move to values of $M > 1$. Here is where

[6] Hamming distance here designates the number of distinct bits in corresponding positions in two bit vectors.

the memory is working properly. Its performance, measured through error rate, depends on correct value of parameter

$$M_{opt} : E_{min} \equiv E(M_{opt}) = \min\{E(M); M > 1\}.$$

Optimal value is unique and guarantees minimal average number of errors.

The second disruption in performance is not so clearly defined as the one we found in $M = 1$, since it is not rooted either in our algorithm nor parameters. It is dependent on the behaviour of the other side, i.e. a DNS server, and on the network conditions. However, we can define the width of region ΔM_e where the error rate is less than some acceptable value e. Sometimes we may find it useful to choose this value with concern to minimal error rate, e.g. $e_\delta = E_{min} + \delta$, so we can use ΔM_e and ΔM_δ interchangeably. Now, formally we have[7]

$$\Delta M_e = M_2 - M_1, E(M_1) = E(M_2) = e \wedge (M_2 > M_1).$$

The third region in the graph is now designated by $M \gg M_2$ where the error rate eventually converges to 0.5 again, as for sufficiently large value of M, relation B_1 is always satisfied, regardless of response times. Thus we have

$$M \gg M_2 \Rightarrow B_1(M, t_1, t_2) \rightarrow 1, \forall(t_1, t_2) \Rightarrow E(M) \rightarrow 0.5.$$

Based on the measurements with memory algorithm, it is possible to assign a set of characteristics, as (M_{opt}, $E_{min}, \delta, \Delta M_e$), to each DNS server. Note that the set is redundant.

3.2 Characteristics Evaluation and Comparison

We can observe distinguishing shape of curve depicted in Fig. 1 also in responses from other servers. For the sake of comparison, several of them are drawn in Fig. 2. The test set contained some randomly chosen DNS servers from the Internet in order to find out what will be the value of error rate for distant (non-local) servers. Thus we cannot provide detailed information about their settings. Anyway, for each we can find out optimal parameter value and corresponding error rate. It is summarised in the following table.

Server	M_{opt}	E_{min}	δ	ΔM_δ
`f1`	1.325	0.0055 ± 0.0131	0.05	0.425
`f2`	1.375	0.0519 ± 0.0674	0.05	0.600
`f3`	1.100	0.0363 ± 0.0952	0.05	0.325
`f4`	1.225	0.1244 ± 0.0742	0.05	0.275
`f5`	1.300	0.0043 ± 0.0170	0.05	1.025
`l1`	1.975	0.0049 ± 0.0135	0.05	1.850

Procedure of obtaining the graph of $E(M)$ can be used for evaluation purposes. From the table with results and picture of $E(M)$ we can see that measurement on `f1`, `f5` and `l1` shows small minimal error rate and `f2`, `f5` and `l1`

[7] In theory, it would suffice to write $\Delta M_e = M - 1, E(M) = e \wedge (M > 1)$.

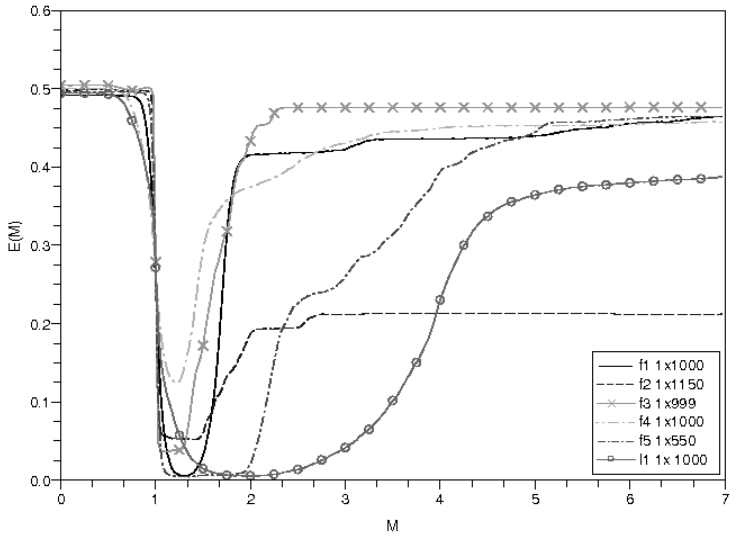


Fig. 2. Comparison of the error rate functions for several DNS servers. Block size is 1 byte.

have relatively wide range of ΔM_e. It is worth to mention that `l1` is one of our local DNS servers, so network weather doesn't influence it too much. Measurements on larger blocks show even smaller error rates about 0.003, under similar conditions.

Evaluating Communication Channel. Considering a selection of appropriate DNS server for communication, as described in section 2.1, we can first obtain the graphs of candidates[8] (by procedure described in section 3.1) and then choose the one with smallest value of E_{min}. In turn, we can conveniently fix the value of parameter to M_{opt}, to guarantee the lowest possible average error rate. Since its value is known, we could use some additional error correcting mechanism in higher layer in order to achieve desired performance (this goes on the expanse of decreased transfer speed).

On the other hand, the width ΔM_e not only says for how large interval of parameter values will be low error rate retained, but conversely, it shows how much may response times vary for a fixed value of M, while the error rate remains at acceptable level. Thus the higher the value of ΔM_e is, the more robust is the data transfer.

Evaluating DNS Servers. The same criteria described above can be used if we are just looking for a good performance DNS server, somewhere in the Internet, with no regards of hidden communication. The low error rate essentially means that with high probability we can find a record in the cache, if we had already

[8] Perhaps measurement should be performed from both potential sides of communication.

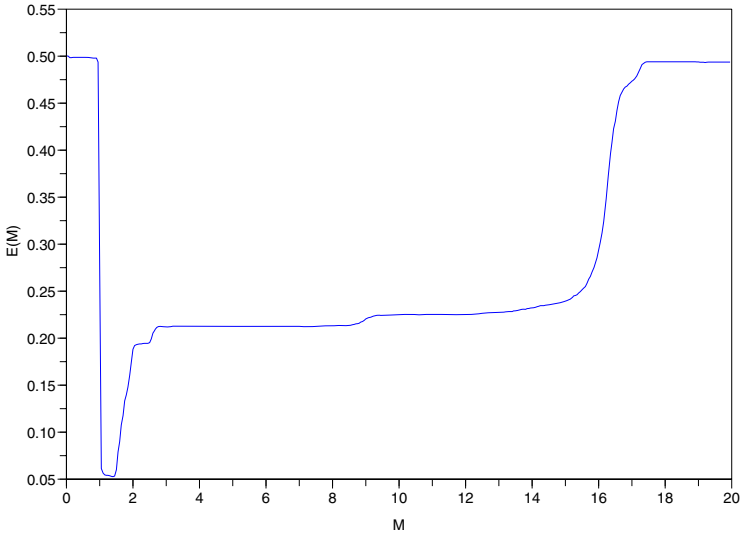


Fig. 3. Distinguishing shape of response from DNS server designated `f2`

requested it before. The low values of $E(M)$ shows efficiency of caching in the particular server. The ΔM_e parameter says, how many times can be the response of cached record faster, compared to uncached one. Thus we want this value for our server to be as large as possible.

Distinguishing the Servers. Beside the numerical quantities presented above, each server possess a unique curve *shape*. Good example of this is the graph designated `f2`, in Fig. 2, which seems to converge to some value less than 0.5 in given range. But evaluating the graph further on, shows the step towards higher values (see Fig. 3), in accordance with results from section 3.1. Also other curves show individual characteristic features which might be used for the purpose of server identification. Perhaps the shape is connected to implementation details of particular server. This interesting effect is beyond the scope of the article, but should be the matter of future research.

4 Conclusions

We have used the procedure of storing and retrieving the data into the cache of DNS server. A brief reasoning revealed some basic properties of such a medium. It is distinguished by relatively high capacity but rather slow access, especially in sequential mode. This might be overcome by using several concurrent threads for access. The relation between write and read access time was given too. The information stored is not persistent. Medium allows indirect and possibly hidden communication of nodes in the network. However, the external ad hoc

synchronisation used in experiments is not sufficient for general purpose transfers. Similarly, the error rates for bare channel are still quite high.

We showed the way in which the optimal parameter setting for algorithm may be found, bringing the lowest possible error rate. However, this algorithm was not intended for real data transfer as there are perhaps more reliable ones. Instead we have used the resulting graphs to evaluate various DNS servers and compare their general caching properties, based only on the responses obtained by our algorithm (without *a priory* knowledge about the server). Another application may be in distinguishing the servers based on this fingerprint.

Acknowledgements. Thanks goes to Slovak Science Agency VEGA, which supported this work by the project No. 1/3104/06.

References

1. Percival, C.: Cache Missing for Fun and Profit 2005 (August 22, 2007), http://www.daemonology.net/papers/htt.pdf
2. Kaminsky, D.: Black Ops 2004 @ LayerOne 2004 (August 22, 2007), http://www.doxpara.com/bo2004.ppt
3. DNS Covert Channels and Bouncing Techniques, (designated as Phrack, Volume 0x0b, Issue 0x3d, but not included in Phrack archive) (May 22, 2007), http://archives.neohapsis.com/archives/fulldisclosure/2005-07/att-0472/p63_dns_worm_covert_channel.txt
4. Mockapetris, P.: Domain Names - Concepts and facilities, STD 13, RFC 1034 (November 1987), http://www.ietf.org/rfc/rfc1034.txt
5. Mockapetris, P.: Domain Names - implementation and specification, STD 13, RFC 1035 (November 1987), http://www.ietf.org/rfc/rfc1035.txt
6. Andrews, M.: Negative Caching of DNS Queries (DNS NCACHE), RFC 2308 (March 1998), http://www.ietf.org/rfc/rfc2308.txt
7. dnstracer online manual (August 23, 2007), http://www.mavetju.org/unix/dnstracer.php
8. Bernstein, D.J.: Notes on the Domain Name System (August 21, 2007), http://cr.yp.to/djbdns/notes.html
9. Minda, M.: Using In-bailiwick Nameservers (February 2005) (August 23, 2007), http://www.nanog.org/mtg-0501/pdf/minda.pdf

Strong Authentication over Lock-Keeper

Feng Cheng and Christoph Meinel

Hasso Plattner Institute, University of Potsdam,
P.O.Box 900460, 14440, Potsdam, Germany
{feng.cheng, christoph.meinel}@hpi.uni-potsdam.de

Abstract. Based on the principle that "the ultimate method to secure a network is to disconnect it", the Lock-Keeper technology has been known as an efficient approach to guarantee the high-level security and prevent online network attacks by physically separating the protected hosts or networks. Because of its simple idea and extensible architecture, the Lock-Keeper system can be easily and seamlessly integrated with other security methods or solutions to provide thorough protection for most actual network-based applications. This paper will propose an advanced strong authentication framework based on the Lock-Keeper. Thanks to Lock-Keeper's physical disconnection, all the credentials, privacies and policies required by the authentication mechanism can be securely stored and manipulated by being completely isolated with both the external and the internal networks. The whole authentication procedure can be performed in the clean and trusted Lock-Keeper GATE component. Based on the proposed framework, a prototypical platform is implemented in the Lock-Keeper to enhance the security of the Lock-Keeper Web Service module, which is one of important Lock-Keeper application modules, and can be applied to secure most web applications in Service-Oriented-Architecture environment.

1 Introduction

The levels of authentication strength rely mostly on the value or sensitivity of the system and information that are protected [1]. Unlike traditional knowledge based authentication, e.g. user-password authentication, which remains pervasive in spite of its known shortcoming and vulnerability to be attacked, strong authentication meets the increased security requirements and enables organizations to verify user identities with high degree of certainty to intensify the online trust. Some credentials-based, multi-factor authentication approaches, such as X.509 [2], SAML-Token [3], physical token like smart cards [4] and even biometric authentication [5], etc., have been proposed as strong authentication solutions and widely used in business and industry [6], [7].

Within these implementations of strong authentication, a centralized Identity and Access Management (IAM) system [8] is usually used to store the credentials related components, e.g. user profiles, privacies or certificates, and provides high efficient authentication services [9]. Unfortunately, the possibility to attack the IAM and in depth the protected resources comes along while the IAM

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 572–584, 2008.

host exposes connections to outside. Stealing the privacy as well as influencing the practical authentication process have been attractive goals for hackers. To protect the IAM system and its hosted authentication procedure against those malicious attacks has been a main challenge for most of strong authentication approaches.

On the other hand, complete physical separation with the external world is recognized as an alternative for most organizations with high-level security requirements to protect their sensitive IT infrastructures. The idea of "*Physical Separation*" that "*the ultimate method to secure a network is to disconnect it*" [10] is simple and easy to understand. The main task is to separate the private network at both logical and physical levels, and simultaneously permit secure data exchange. To provide services through such a special "*Physical Separation*" mechanism, a feasible strong authentication is demanded.

Driven by these requirements, an advanced authentication strategy, which combines both the software-based "*Strong Authentication*" technology and the hardware-based "*Physical Separation*" technology, is proposed in this paper. Lock-Keeper, which is a new implementation of "*Physical Separation*", is used to host the IAM system and perform the authentication process. Such new ideas as "*Offline Authentication*" and "*Offline Maintaining*" are realized by this Lock-Keeper Strong Authentication Framework. To present the applicability and usability of our idea, a secure Web Service authentication platform is implemented in which the strong authentication IAM system, the Web Service Server and its host network are protected well while Web Services are provided normally.

The paper is organized as follows. Section 2 provides some background knowledge. The Lock-Keeper architecture is introduced as well in this section. Section 3 describes the architecture and working principles of our Lock-Keeper Strong Authentication Framework. In Section 4, an implemented Web Service authentication platform is presented to demonstrate the applicability and usability of the proposed Lock-Keeper Strong Authentication Framework. We conclude the paper and preview future works in the last section.

2 Background

In this section, we briefly introduce some new challenges of "*Strong Authentication*" and then discuss the necessities to deploy "*Strong Authentication*" in the case of "*Physical Separation*" protected scenarios.

2.1 Strong Authentication

As an example of the most popular weak authentication method, the traditional user-password authentication, which only requires the information of "*something you know*" from the user, is vulnerable to many attacks [7], including Keystroke Monitoring, Dictionary Attacks, Network Sniffing, Man-in-Middle attack, Social Engineering attack, etc. In comparison, the advanced strong authentication demands many additional information, such as knowledge on "*something you have*" and "*something you are*". For the instance of the security model established for

ATM cards and machines, the PIN ("*something you know*") and possession of the bank card ("*something you have*") are both necessary for a user to pass through the authentication procedure and access the banking services (e.g. drawing or transferring money). To this effect, "*Strong Authentication*" is also defined as two or multiple factors authentication [6].

With the rapid growth of e-Commerce, e-Banking and e-Government, more and more enterprises, financial sections and governmental organizations have deployed strong authentication solutions to protect their online resources and provide secure services to end users. To realize and perform strong authentication in the network environment, some new assistant techniques and methods have been proposed in recent years. For example, X.509, which describes standard ways to formulate and validate Public Key Infrastructure (PKI) certificates required by online strong authentication, has been used for many software companies and services vendors to guarantee the security while providing the XML-based Web Services [2]. Security Assertion Markup Language (SAML) has also been suggested as a standard for exchanging strong authentication data between different domains, which probably belong to different organizations, in the federated Service Oriented Architecture (SOA) environment [3]. IAM system is used to hold the authentication credentials and manage the authentication procedure at the edge of the protected target, i.e. the border of a sensitive network [8]. Therefore, how to securely set up and protect the online IAM system as well as reliably perform the strong authentication in the loosely coupled SOA environment has been one of new challenges.

2.2 Physical Separation

As a simple but intuitive security concept, the principle of "*Physical Separation*" is to find a way to transmit data between two different networks without having to establish a direct and physical connection. Currently, there are many different "*Physical Separation*" implementations, such as e-GAP-based Intelligent Application Gateway (IAG) [11], DualDiode [12], NRL Pump [13], etc. Lock-Keeper is also implemented based on this "*Physical Separation*" idea. It works as a sluice to guarantee that hackers and malign data have no opportunities to break into the internal network by any means of online attacks. we use a SingleGate Lock-Keeper system as an prototype to briefly explain what the Lock-Keeper is and how it works [10], [14].

As shown in Figure 1, a SingleGate Lock-Keeper system consists of three independent Single Board Computers (SBCs): INNER, OUTER and GATE, which are connected using a patented switch unit. This hardware based switch unit restricts the connection so that GATE can be connected with only one partner at any time, either INNER or OUTER. Besides these hardware components, there are also Lock-Keeper Secure Data Exchange (LK-SDE) software running in the Lock-Keeper system. LK-SDE software includes several application modules located on INNER and OUTER, which work as interfaces and provide popular network services to outside users. Currently, there are four LK-SDE application modules implemented, i.e. File eXchange (File-X) Module, Mail

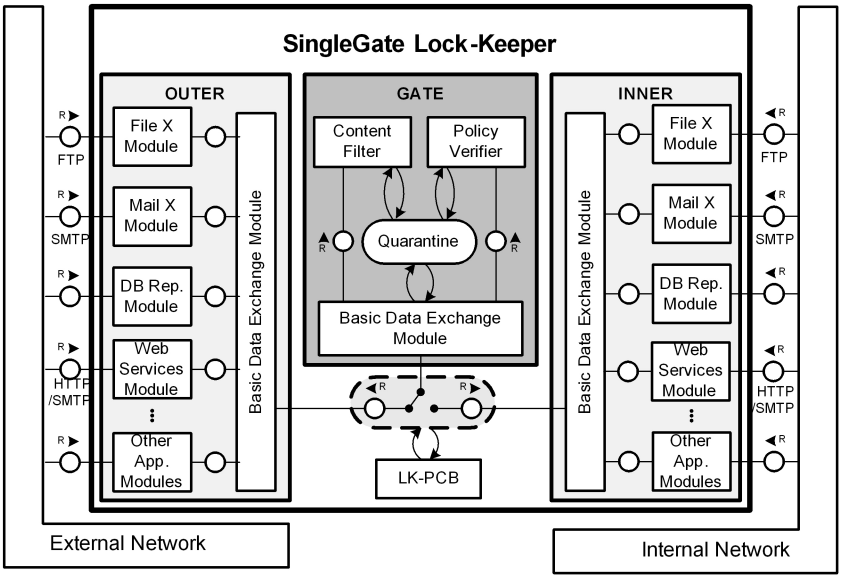


Fig. 1. Conceptual Architecture of the SingleGate Lock-Keeper

eXchange (Mail-X) Module, Database Replication (DB-Rep) Module and Web Service (WS) Module. Normal communication protocols, such as FTP, SMTP, HTTP, etc., are stopped and analyzed respectively by these application modules. Then, standard file-based Lock-Keeper Message Containers (LKMC) can be created to carry the data for the received network traffic. These LKMCs will be transferred to the other side by "*Basic Data Exchange Module*". In particular, because GATE is also a normal PC, it is possible to integrate Third-Party security software, e.g. virus scanning software, mail analysis tools, or content filtering methods, etc. into LK-SDE architecture, which help to check data traffic and prevent offline attacks, e.g. virus, malicious codes, etc.

2.3 Our Motivations

According to previous discussions, we are motivated to propose an advanced authentication framework by integrating "*Strong Authentication*" into the Lock-Keeper system. Many benefits are expected. Firstly, credentials required by the strong authentication, such as user information databases, privacy database and certificate store, etc., can be saved safely on GATE and are impossible to be directly accessed by both internal and external users. Secondly, the authentication methods or cryptography algorithms can be flexibly deployed on GATE and are impossible to be changed or abused. Thirdly, the authentication operations and procedures can be performed unaffectedly in an isolated environment, i.e. Lock-Keeper GATE, which is called as "*offline authentication*". Fourthly, the internal resources including all the internal hosts and the network infrastructure are protected well while normal network services are provided simultaneously.

At last, based on the Lock-Keeper Authentication Framework, it is possible for Lock-Keeper to support more web based applications, especially most newly appeared SOA applications, and proffer protection for more practical scenarios, which can significantly improve the Lock-Keeper's usability.

3 The Lock-Keeper Strong Authentication Framework

As shown in Figure 2, there are two main components in this framework, i.e. Authentication Proxies on INNER/OUTER and an IAM system on GATE. "*Authentication Proxy*" works like other Lock-Keeper application modules to analyze and parse the incoming network traffic. The IAM system composes of an "*Authentication Management Engine*" and a "*Credential Virtual Machine*" where the credential related databases and stores can be safeguarded.

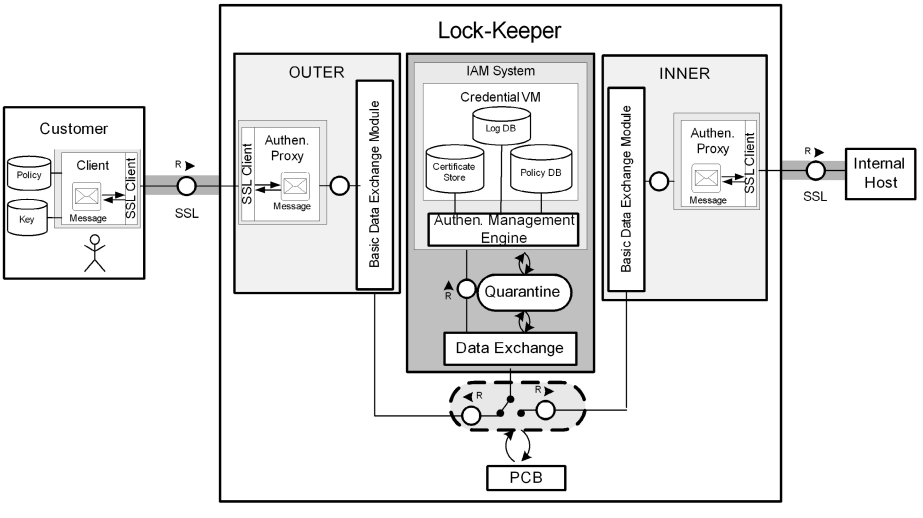


Fig. 2. The Lock-Keeper Strong Authentication Framework

3.1 Authentication Proxy on INNER and OUTER

External users communicate with the "*Authentication Proxy*" using normal network connections. For instance, a customer can send a signed and encrypted request to the Lock-Keeper "*Authentication Proxy*" through SSL-based tunnel, e.g. using HTTPS. The proxy will parse the data traffic belongs to this received request, i.e. network-level packets, and then reconstruct it into the application-level LKMC, which includes a message body and a message header with several additional information required by communication and security. As shown in Figure 3, the header of a standard LKMC consists of a "*Routing Section*", an "*Authentication Section*" and an additional section for containing other information, which may probably be used in next steps. The architecture of LKMC is

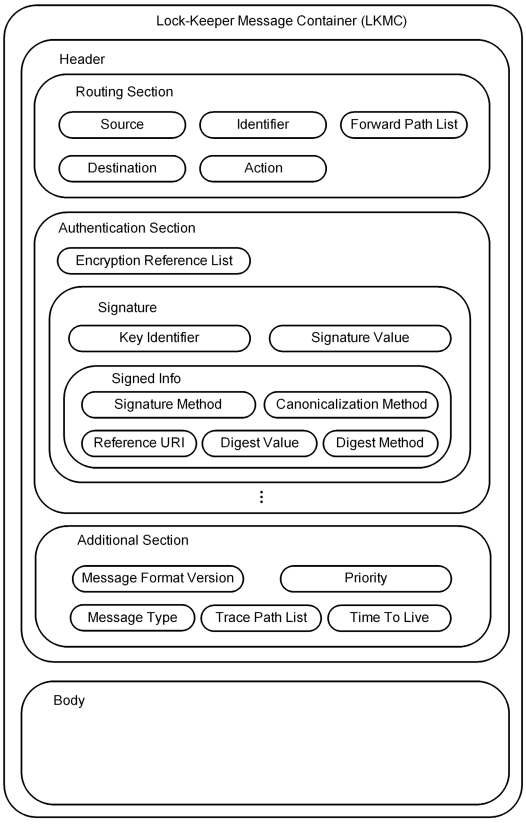


Fig. 3. The Lock-Keeper Message Container for Strong Authentication

flexible to be easily extended to satisfy other requirements. The "*Authentication Proxy*" on OUTER is also responsible for forwarding responses, either successful or failed, back to end customers or potential hackers.

After being preprocessed on OUTER, the LKMC is transferred to GATE through the Lock-Keeper "*Basic Data Exchange Module*". As soon as it completely arrives at GATE, the "*Authentication Management Engine*" will pass it to the IAM system.

The "*Authentication Proxy*" on INNER forwards the LKMC message to the protected service host (i.e. the internal server), which is located in the internal network. If necessary, the request will be verified again by the server and then the response message can be generated after the invocation of requested applications. Similar to the incoming request message, the outgoing response message is also required to be issued, signed, and encrypted by the internal server with the specified policy, which is identical to security policies on both GATE and the client side so that it can successfully pass through the same authentication procedure. The data traffic for the response will be translated into LKMC message

by the "*Authentication Proxy*" on INNER when it arrives at Lock-Keeper on its way back.

3.2 IAM System on GATE

The LKMC message is required to be authenticated by the IAM System on GATE, which is designed based on "*Strong Authentication*". The whole authentication procedure is started and controlled by the "*Authentication Management Engine*", which decrypts the LKMC and extracts authentication information. Then the further authentication operations can be performed by communications between the "*Authentication Management Engine*" and the corresponding credential components. Besides, the content of message body can also be scanned using other application-level security software to prevent offline attacks. For this purpose, a content scanning software with its affiliated virus pattern library should be integrated. The "*Authentication Management Engine*" can also be implemented according to the different practical application scenarios.

As shown in Figure 2, a "*Certificates Store*", a "*Policy DB*" and a "*Log DB*" are contained in the *Credential VM* on the GATE. As normal authentication approaches, these components need to be updated regularly. However, the Lock-Keeper's special switch mechanism makes it impossible to remotely access and update these sensitive parts because any kinds of normal connections have been prevented. So we proposed a method, called "Offline Maintaining", which makes it possible for administrators to easily configure and update these credentials using the Virtual Machine (VM) technology [15]. Thanks to some helpful features of such VM implementions as User-Mode-Linux (UML) [16], we can realize the "*Credential VM*" by a main VM file and a Copy-On-Write (COW) VM file. The main VM file contains a pure operating system where the basic storage architecture is installed for carrying authentication credentials. A same copy of this file must be backed up in the administrator side. If new updates or configurations are required, the administrator firstly makes modifications on the backup VM locally and generates the modified COW file, which is usually not too large (about 20-50 MB), and then sends it to GATE to replace the currently used VM COW file. However, all the operations must be done offline, i.e. at the moment when the Lock-Keeper is not working and completely disconnected. We only provide the privilege for the administrator to access the GATE locally by traditional I/O devices such as keyboard and monitor. As an alternative, a mobile harddisk can also be installed on GATE so that the administrator can flexibly take it out if necessary and plug it in again after updating. By such an implementation of *Authentication Credential Storage*, the security of sensitive authentication data is enhanced again because the instinctive feature of the VM technology on "OS Isolation" provides another layer of security.

If the LKMC message is verified to be invalid, a fault message with a rejection indicator will be generated and then sent back to the customer through OUTER, who is probably a hacker in this case. When the LKMC passes the verification process, it will be transferred to INNER.

4 Security Enhancement of Lock-Keeper Web Service Module

In this section, a secure Web Service providing platform, which is applicable for most SOA application scenarios with high-level security requirements [17], is implemented based on the previously proposed Lock-Keeper strong authentication framework. As shown in Figure 4, the Lock-Keeper Web Service providing platform is basically composed of a consumer, Lock-Keeper Web Service Module and a provider.

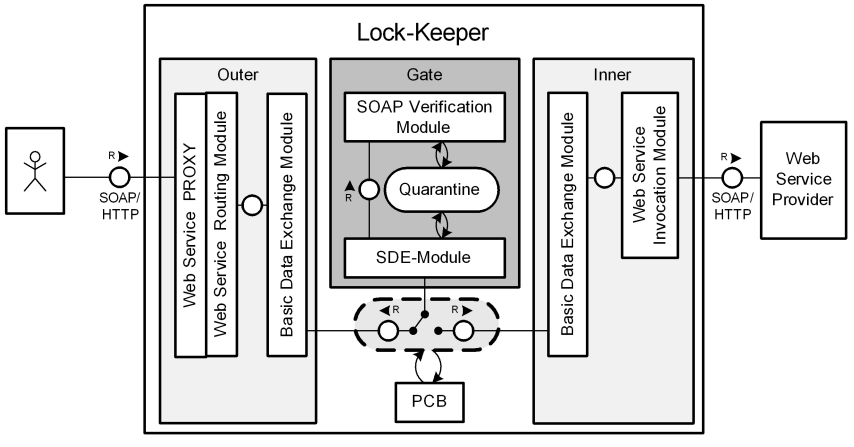


Fig. 4. A Secure Platform for Providing Web Services over Lock-Keeper

The consumer acts as a legally registered user who is supposed to have all the necessary security materials required by the strong authentication. Figure 5 shows a Web Service client on the consumer side. Using this client, consumers can import and manage registered certificates on the "*Certificates Panel*", create and compose security policies using the imported certificates or the supported encryption algorithms on the "*Policy Panel*". For example, there are two composed policies shown in the Figure 5. One of them is called as "*secure*", which requires encryption using certificate "*gate*" and signature using certificate "*client*". This client also provides the interface to create, edit, send out the Web Services request and later receive the response. These tasks will be managed on the "*Monitor Panel*", explained in the next section.

The "*Web Service Provider*", indicated in Figure 4, offers a desired service. As usual Web Service server, it receives the Web Service request, triggers Web Methods and creates the Web Service response.

4.1 A Secure Lock-Keeper Web Service Providing Platform

The Web Service Module inside the Lock-Keeper system consists of a "*Web Service Proxy*" and a "*Web Service Routing Module*" on OUTER, which co-realize

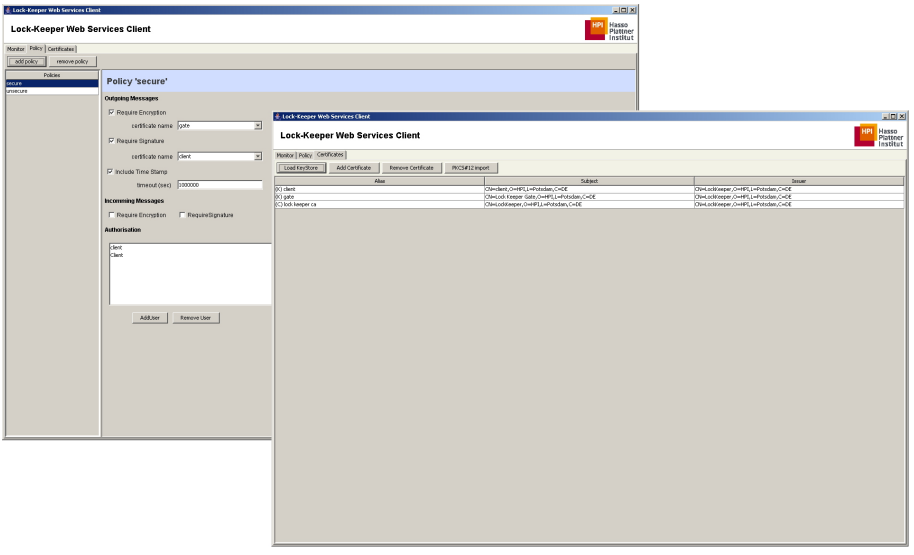


Fig. 5. Policy Panel and Certificates Panel in the Web Services Test Client

the functionality of "*Authentication Proxy*" described in the previously proposed framework, a "*SOAP Verification Module*" on GATE to actually act as IAM system, and a "*Web Service Invocation Module*" on INNER, which works as another "*Authentication Proxy*". The "*Web Service Proxy*" exposes the Web Methods hosted on the internal provider to the external network. The incoming SOAP requests are accepted over HTTP or SMTP by the "*Web Service Proxy*" and then translated into the LKMC. The "*Web Service Routing Module*" forwards the LKMC to the Lock-Keeper "*Basic Data Exchange Module*". On GATE, the "*SOAP Verification Module*" performs the concrete authentication procedure. The architecture of the "*SOAP Verification Module*" is illustrated in the Figure 6. It has realized functionalities of both "*Trust Management*" and "*Threat Protection*", specified in most popular WS-Security standards (see [18] and [19]). The "*Policy Store*" provides essential information to guide decisions and actions performed by the "*Trust Management*" and the "*Threat Protection*". Specified in a certain policy, certificates or private keys are required, which are available from the *Certificate Store*. The verified LKMC is transferred to INNER. For failed verifications, the original request, supposed to be malicious, will be removed as soon as possible and a response with a error message will be sent back to the consumer over OUTER. The information during the whole process is recorded into the "*Logging File*".

After receiving the successfully verified SOAP request, the "*Web Service Invocation Module*" on the INNER invokes the corresponding Web Method hosted in the Web Service provider.

This platform is implemented based on our proposed Lock-Keeper Strong Authentication Framework. The design of the whole architecture does exactly

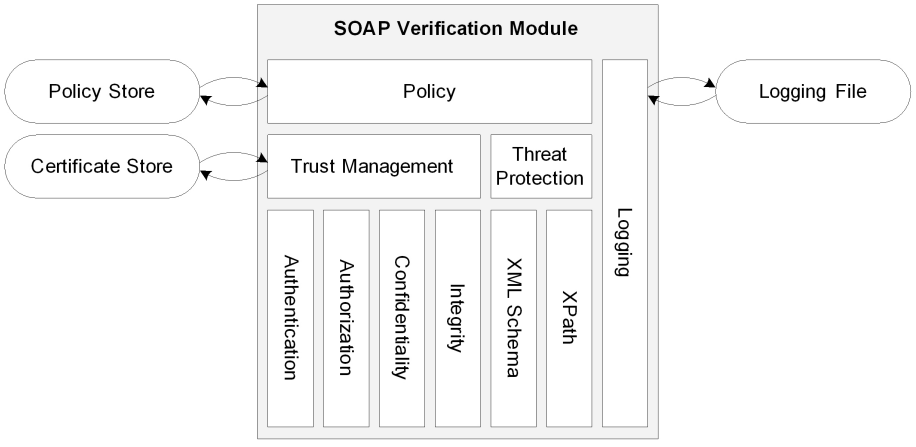


Fig. 6. SOAP Verification Module on GATE

satisfy the well-known Web Services security model proposed in [19], which divides the Web Services security into three layers: "*Secure Network*", "*Secure Web Service Host*" and "*Secure Web Service Message*".

4.2 Experiment Result

The proposed platform has been successfully deployed in a real world "*Online-Police-Station*" project. As a sensitive governmental department, the police has the high-level security requirement to physically separate its internal network with Internet. However, there should be a secure way provided for original citizens to send such reports as traffic accidents, information on new residential places, criminal cases, etc.

In our experiment, the "*Online-Police-Station*" is realized on an internal police Web Service server, which is "*connected*" with Internet through Lock-Keeper. As a simple example, we just install a Web Method, called "*String*", on this server, which is used to file the incoming case report and send acknowledges back to citizens. The Web Service client, especially the "*Monitor Panel*", which has been mentioned before and is practically realized on the police's public portal outside Lock-Keeper, offers citizens an interface to create, sign and encrypt the SOAP request as well as check the response.

As indicated in the Figure 7, the original SOAP message, shown in the top-left dialog box, embeds a simple request with a case report, e.g. "*My Car was stolen*". After enforcing a composed policy called "*secure*", which would be required by later strong authentication on Lock-Keeper, a secured SOAP request is generated in the top-right box. By clicking the button "*Invoke Soap Request*", the encrypted SOAP request can be sent to the *Web Service Provider*. Then it will be processed by the strong authentication process. After a couple of seconds (the required Round-Trip-Time mostly depends on the length of the Lock-Keeper switch interval [10]), the secured response can be received in the bottom-right

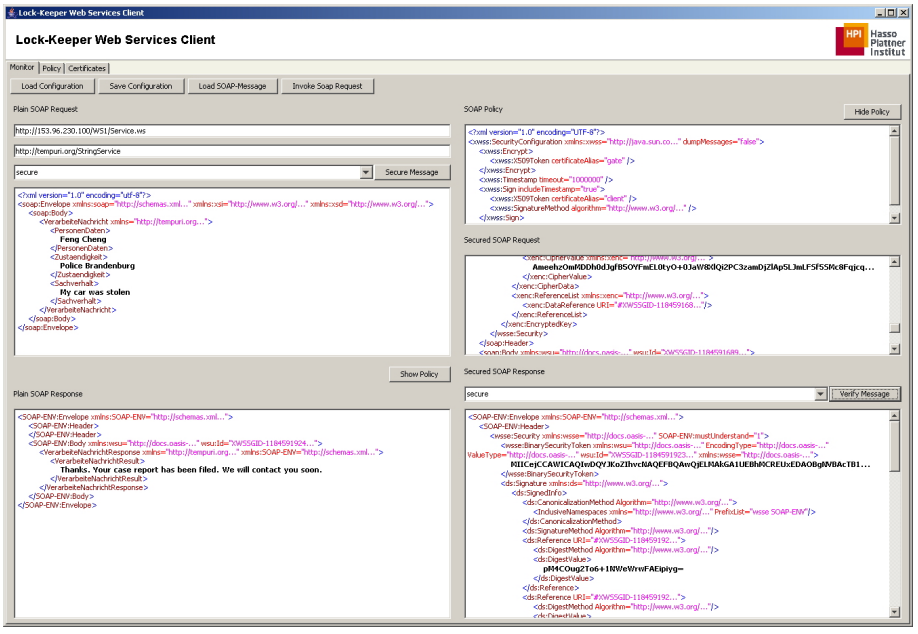


Fig. 7. A Web Service Request and its successful Response shown in Web Service Client

box. The plain-text result (i.e. "*Thanks, Your case report has been filed. We will contact you soon*") will be displayed after being decrypted by enforcing the corresponding policy "*secure*". In the case of failed authentications due to any kinds of errors, faults or probable attacks, the SOAP request is impossible to pass through the Strong Authentication system on GATE and intrude into the internal network of the police. A fault response with the indicator, such as "*The Request is Illegal*", can be generated and sent back to the client from Lock-Keeper.

5 Conclusions

In the paper, we present several benefits on the combination of the "*Strong Authentication*" technology with "*Physical Separation*" technology and propose an advanced Lock-Keeper Strong Authentication Framework. Through this framework, all the authentication components, user profile, privacy and policy are protected well on GATE, which are impossible to be actively accessed from outside. The whole authentication procedure is performed on GATE and impossibly to be affected, which demonstrates the idea of "Offline Authentication". The *Credential VM* in the integrated IAM system can be easily and securely updated, which realizes the concept of "Offline Maintaining" as well. Both the incoming request and the outgoing response are required to be verified by the authentication module so that the insider attacks can be prevented. Meanwhile, an

applicable *Strong Authentication* system, realized in the "*Physical Separation*" device, i.e. Lock-keeper, can significantly improve the usability of this high-level security solution. The secure Web Service Providing platform shown in this paper illustrates the applicability and usability of our proposed Strong Authentication Framework. However, there are still many open issues to be solved around this topic. The Lock-Keeper can be used as a suitable host for the federated authentication proxy to exchange and translate the different authentication information required by different organizations. A Lock-Keeper-based federated authentication approach would be a promised security solution for SOA applications. Other special authentication [20] and access control schemes can also be integrated in Lock-Keeper to enhance security of existing applications. Development of a unified authentication client, e.g. a plugin or extension for normal web browsers, also makes great senses to popularize this idea.

References

1. Zviran, M., Haga, W.J.: A Comparison of Password Techniques for Multilevel Authentication Mechanisms. Computer Journal 36(3), 227–237 (1993)
2. Housley, R., Ford, W., Polk, W., Solo, D.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF - Network Working Group, The Internet Society, RFC 2459 (January 1999)
3. Cantor, S., Moreh, I.J., Philpott, S.R., Maler, E.: Metadata for the OASIS Security Assertion Markup Language (SAML), V2.0. OASIS SSTC, oasis-open.org (2005)
4. Rankl, W., Effing, W.: Smart Card Handbook, 3rd edn. John Wiley and Sons, Ltd., Hoboken, NJ (2003)
5. Wayman, J.L.: Fundamentals of Biometric Authentication Technologies. International Journal of Image and Graphics 1(1), 93–113 (2001)
6. RSA Security, Inc. Strong Authentication: An Essential Component of Identity and Access Management. White Paper, RSA Security, Inc.: SA-WP-0804 (2004)
7. Lobel, M.: Case for Strong User Authentication White Paper, TRS, PricewaterhaouseCoopers: CSUA-WP-0200 (2005)
8. Witty, R.J., Wagner, R.: The Growing Need for Identity and Access Management. White Paper, Gartner, Inc.: AV-21-4512 (2003)
9. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications. Springer, Berlin, Germany (2004)
10. Cheng, F., Meinel, Ch.: Research on the Lock-Keeper Technology: Architectures, Applications and Advancements. International Journal of Computer & Information Science 5(3), 236–245 (2004)
11. IAG 2007 website in Microsoft (2006-2007), www.microsoft.com/iag
12. Menoher, J.: Owl Computing Product Overview: Secure One-Way Data Transfer Systems. White Paper, Owl Computing Technologies, Inc. (2007)
13. Kang, M.H., Moskowitz, I.S.: A Pump for Rapid, Reliable, Secure Communication. In: CCS 1993. Proceedings of 1st ACM Conference on Computer & Communications Security, Fairfax, VA (1993)
14. Lock-Keeper WebSite of Siemens Switzerland (2005-2007), www.siemens.ch
15. Cheng, F., Meinel, C.: Deployment Virtual Machines in Lock-Keeper. In: WISA 2006. LNCS, vol. 4298, Springer, Heidelberg (2006)

16. User Mode Linux Core Team: User Mode Linux HOWTO, http:user-mode-linux.sourceforge.net
17. Cheng, F., Menzel, M., Meinel, Ch.: A Secure Web Services Providing Framework based on Lock-Keeper. In: APNOMS2007. LNCS, vol. 4773, Springer, Heidelberg (2007)
18. ForumSystems: Forum Xwall - XML Firewall Product Data Sheet (2005), www.forumsystems.com/papers/
19. Curphey, M., Scambray, J., Olson, E., Howard, M.: Improving Web Application Security: Threats and Countermeasures. Microsoft Press, Washington (2003)
20. Neuman, C., Yu, T., Hartman, S., Raeburn, K.: The Kerberos Network Authentication System Kerberos RFC4120, kerberos.info (July 2005)

Short Ballot Assumption and Threeballot Voting Protocol*

Jacek Cichoń, Mirosław Kutyłowski, and Bogdan Węglorz

Institute of Mathematics and Computer Science, Wrocław University of Technology
jacek.cichon@pwr.wroc.pl, miroslaw.kutylowski@pwr.wroc.pl
pungabw@wp.pl

Abstract. We analyze the Threeballot voting system proposed recently by R. Rivest. We investigate the relation between the number of the candidates in a race and effectiveness of Strauss' attack. We also show that in a reasonable scenario it is impossible to reconstruct voters' preferences for a single race with two candidates.

Keywords: e-voting, anonymity.

1 Introduction

Recently, there has been a lot of interest in paper based voting schemes that offer additional features such as verifiability of the results. This is due, among others, by recent problems with voting machines (see e.g. California Report [13]). In this situation, paper based methods with no black box electronic devices may contribute to more social acceptance and transparency of elections. New schemes should also provide better resilience to election frauds. This concerns also classical paper based methods, where verifiability is quite limited, once the ballots have been exchanged in a ballot box. New paper based schemes should prevent this: if a voter gets a receipt of the ballot cast, then removing his ballot from the ballot box and replacing it with a different one might be detected. However, we get a new problem: the receipt must not show the voting preferences of the voter. Failing this requirement could have profound consequences, like enabling vote-selling or forcing a voter to vote in a certain way.

It has also been pointed out that a voting scheme and its security mechanism should be understandable for an average voter. Furthermore, Rivest [9] argues that it is desirable to avoid any kind of cryptographic encoding on the receipts. There are at least two reasons for that: cryptographic methods can be broken so that a receipt may betray the voter's choice. Second, some voters may believe that cryptographic codes may leak information in a malicious way (via a subliminal or kleptographic channel [6]).

Somewhat surprisingly, it has turned out that major improvements over the traditional schemes are possible. A number of schemes have been designed: Punchscan [8], Prêt à Voter (see [4] and later publications), Threeballot [9], VAV and TWIN [10].

Unfortunately, none of these schemes is yielding the best solution in all aspects. For Punchscan and Prêt à Voter, a voter can doubt whether her or his vote can be retrieved

* Partially supported by Polish Ministry of Science and Education, grant 3 T11C 011 26.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 585–598, 2008.

once one of the sheets is destroyed. In this situation some people may believe that the elections are simply fake.

Also TWIN can be hard to accept from psychological point of view. Even if at the beginning some voters leave the polling station without receipts, the first voters may feel uncomfortable knowing that the next person may get a copy of one part of his ballot. Note that nontrivial information can be revealed if an organized group of voters is coming to a polling station together: they may derive at lot of information on the contents of the ballot box at the moment when they arrive. So a conscious voter may prefer to vote at the end of an election day. Finally, some fraction of voters will not appear at the polling station.

For VAV also some sociotechnical attacks are possible. The voters may catch the idea that taking an antivote as a receipt guarantees that no information about her or his preferences can be derived no matter how other people vote. If all voters follow this strategy, then the tallying committee can match the votes with the antivotes and modify some number of votes that do not match to any antivote. As for Threeballot, it might be problematic to check that a ballot has the desired form. If control is ineffective, a voter may vote twice for his candidate and revoke a vote for another candidate.

The main problem with Threeballot is necessity to adopt Short Ballot Assumption (SBA) [10]: if the list of candidates in a race is long, or there are many races on the same ballot, then privacy of a vote is endangered. This has been observed shortly after presenting Threeballot [11] (for details see Section 3). The other problem is complexity of procedure; understanding of how to vote might be a non-trivial issue for some voters [3]. Problems of this kind are discussed also in [12]. Other problems have been pointed out in [1]. The idea of the attack performed in the case of two candidates race (say, with candidates Alice and Bob) is to pay for a specific behavior. Then the attacker exchanges some number of ballots containing a mark for Alice with ballots containing a mark for Bob. In [5] it was observed that if the voters fill the ballots at random (except for the final mark that indicates the voter's choice), then the conditional probabilities for voting preferences change, if the receipt of a voter is taken into consideration. This observation was further generalized in [7]. Similar problems have been reported in [2].

Main Problem. Our goal is to find analytically appropriate parameters for Short Ballot Assumption and make Threeballot immune against the attack from [11]. Of course, a rough approximation can be obtained from the experiments (like in [11]), but this does not apply to security margin: failure of an attack observed many times is not a strong argument in favor of a protocol. This problem was already considered for the case of 2 candidate races and multiple races in one ballot [7]. We focus on the problem of a single race on one ballot, since for security reasons the races should be separated.

Results. In Section 2 we show that even in the simplest situation (all but one voter choose the same candidate), it is impossible to say which receipt is owned by the voter who has chosen a different candidate, as long as the number of voters is reasonable.

In Section 3 we develop almost exact formulas for the expected number of valid 3-ballots with a vote for a given candidate that can be reconstructed from the published set of ballots and the receipt held by a voter. The main outstanding problem is to estimate the variation of this number.

Overview of Threeballot System. Each paper ballot, called *3-ballot*, contains three columns and as many rows as the candidates in a race (see Fig. 1). Each row corresponds to one candidate. In a row there are three "bubbles", one bubble per column.

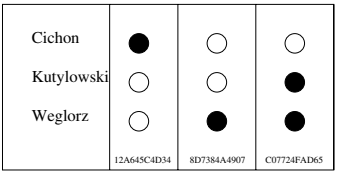


Fig. 1. A 3-ballot with candidates Cichon, Kutylowski and Weglorz, a vote cast for Weglorz

In order to vote for a candidate A the voter has to fill exactly 2 bubbles in the row corresponding to A. In each of the other rows the voter must fill exactly one bubble. The choice which bubbles to fill in a row is arbitrary. A ballot that does not obey these rules is rejected by a checker device. If a ballot is correct, an ID is printed in each column; the ID's in different columns are unrelated and random. Then the columns are separated; each column (together with the ID) forms a *ballot*. The voter chooses one of them and gets its copy.

Finally, the voter casts all his ballots into the ballot box. After opening the ballot box all ballots found inside are published on a bulletin board. The number of the votes for the candidate A is computed as $m - n/3$, where m is the number of ballots containing a filled bubble in the row of A and n is the total number of ballots in the ballot box.

The most important feature of Threeballot is that no matter which candidate has been chosen by the voter, the receipt may have any possible pattern of filled bubbles. So a receipt preserves privacy in the information-theoretic sense as long as a single receipt is concerned.

2 Two Candidates Case

Assume now that Threeballot is used in a race with two candidates, say $\mathcal{A}$ and $\mathcal{B}$, and a small group of voters (say, for electing a chairman in a small faculty). A ballot can take one of the forms: $\frac{\bullet}{\bullet}$, $\frac{\bullet}{\circ}$, $\frac{\circ}{\bullet}$, $\frac{\circ}{\circ}$; a 3-ballot is a triple, for instance $(\frac{\bullet}{\bullet}; \frac{\bullet}{\circ}, \frac{\circ}{\circ})$, where the first element denotes the ballot copied into a receipt, and the order of the second and the third elements is immaterial. We assume that we know the receipt of each voter- the scheme was designed to offer anonymity even if the receipts become revealed. Also, all ballots cast are known to the public (otherwise the election results cannot be checked).

Assume that person $\mathcal{P}$ has voted for candidate $\mathcal{A}$. Then:

- If $\mathcal{P}$ has a receipt $\frac{\bullet}{\bullet}$, then the other columns of his 3-ballot are $\frac{\bullet}{\circ}$ and $\frac{\circ}{\circ}$.
- If $\mathcal{P}$ has a receipt $\frac{\bullet}{\circ}$, then the other columns of his 3-ballot are either $\frac{\circ}{\bullet}$, $\frac{\bullet}{\circ}$, or $\frac{\bullet}{\bullet}$, $\frac{\circ}{\circ}$.
- If $\mathcal{P}$ has a receipt $\frac{\circ}{\bullet}$, then the other columns of his 3-ballot are $\frac{\bullet}{\circ}$, $\frac{\bullet}{\circ}$.
- If $\mathcal{P}$ has a receipt $\frac{\circ}{\circ}$, then the other columns of his 3-ballot are $\frac{\bullet}{\bullet}$, $\frac{\bullet}{\circ}$.

If $\mathcal{P}$ votes for candidate $\mathcal{B}$, then the situation is dual.

The above restrictions and data available after elections (the number of votes cast for each candidate, the ballots cast, and the receipts held by the voters) may be used to deduce some information on the voters' preferences. The most promising case seems to be the election outcome in which there is a single vote for candidate $\mathcal{B}$ - in this case one might be tempted to determine who voted against $\mathcal{A}$. The attack should be regarded as successful even if we merely eliminate some people as possible voters for $\mathcal{B}$ and thereby reduce the anonymity set of the voter who has chosen $\mathcal{B}$.

Assume that Alice is the only voter who has chosen $\mathcal{B}$ and all other voters have chosen $\mathcal{A}$. First consider the case that Alice has prepared her vote in the form ($\frac{\circ}{\bullet}$; $\frac{\bullet}{\bullet}$, $\frac{\circ}{\circ}$), where $\frac{\circ}{\bullet}$ is her receipt. We would like to find a set of 3-ballots for which the same information would be available for an external observer, but the voter who has chosen $\mathcal{B}$ holds a receipt different from $\frac{\circ}{\bullet}$. First we make the following steps:

Step 1: Replace the 3-ballot of Alice by ($\frac{\circ}{\bullet}$; $\frac{\bullet}{\circ}$, $\frac{\bullet}{\circ}$).
In this way we have used two extra ballots $\frac{\bullet}{\circ}$, $\frac{\bullet}{\circ}$ and we have extra ballots $\frac{\bullet}{\bullet}$, $\frac{\circ}{\circ}$ which are not assigned to any voter. Also, at this moment we have no voter that has chosen $\mathcal{B}$.

Step 2: We find a voter that has chosen 3-ballot ($\frac{\bullet}{\circ}$; $\frac{\bullet}{\circ}$, $\frac{\circ}{\bullet}$) (with receipt $\frac{\bullet}{\circ}$). We change his 3-ballot to ($\frac{\bullet}{\circ}$; $\frac{\bullet}{\bullet}$, $\frac{\circ}{\circ}$). So this voter still chooses option $\mathcal{A}$. We have used the extra ballots $\frac{\bullet}{\bullet}$, $\frac{\circ}{\circ}$ from Step 1, one ballot $\frac{\bullet}{\circ}$ that was removed in this step can be used to cover deficit of $\frac{\bullet}{\circ}$ ballots from Step 1. So after Step 2 we have still a deficit of one ballot $\frac{\bullet}{\circ}$ and a surplus of one ballot $\frac{\circ}{\bullet}$.

Step 3: Execute one of the steps:

Step 3A: We find a voter $\mathcal{X}$ with 3-ballot ($\frac{\circ}{\circ}$; $\frac{\bullet}{\bullet}$, $\frac{\bullet}{\circ}$) and change it to ($\frac{\circ}{\circ}$; $\frac{\bullet}{\bullet}$, $\frac{\circ}{\bullet}$).

Step 3B: We find a voter $\mathcal{Y}$ with 3-ballot ($\frac{\bullet}{\circ}$; $\frac{\circ}{\bullet}$, $\frac{\bullet}{\circ}$) and change it to ($\frac{\bullet}{\circ}$; $\frac{\circ}{\bullet}$, $\frac{\circ}{\bullet}$).

Step 3C: We find a voter $\mathcal{Z}$ with 3-ballot ($\frac{\bullet}{\bullet}$; $\frac{\circ}{\circ}$, $\frac{\bullet}{\circ}$) and change it to ($\frac{\bullet}{\bullet}$; $\frac{\circ}{\circ}$, $\frac{\circ}{\bullet}$).

After Step 3, there is no deficit and no surplus of ballots, and we have changed the vote of, respectively, person $\mathcal{X}$, $\mathcal{Y}$ or $\mathcal{Z}$ from $\mathcal{A}$ to $\mathcal{B}$. Since the set of ballots in the ballot box as well as the receipts of the voters do not change during the transformations described, we cannot exclude any person from being a voter of $\mathcal{B}$. The only technical prerequisite is that there is at least one voter with 3-ballot ($\frac{\circ}{\circ}$; $\frac{\bullet}{\bullet}$, $\frac{\bullet}{\circ}$).

Similarly, if Alice votes with ($\frac{\circ}{\bullet}$; $\frac{\circ}{\bullet}$, $\frac{\bullet}{\circ}$), then we may transform the 3-ballots as follows:

- we replace the 3-ballot of Alice by ($\frac{\circ}{\bullet}$; $\frac{\bullet}{\circ}$, $\frac{\bullet}{\circ}$),
- we find a voter with 3-ballot ($\frac{\bullet}{\circ}$; $\frac{\bullet}{\circ}$, $\frac{\circ}{\bullet}$) and change it to ($\frac{\bullet}{\circ}$; $\frac{\circ}{\bullet}$, $\frac{\circ}{\bullet}$).

Let us consider an undirected graph G where nodes are the possible 3-ballots with a vote for $\mathcal{B}$, namely v_1 =($\frac{\circ}{\bullet}$; $\frac{\bullet}{\bullet}$, $\frac{\circ}{\circ}$) v_2 =($\frac{\circ}{\circ}$; $\frac{\bullet}{\bullet}$, $\frac{\circ}{\bullet}$), v_3 =($\frac{\bullet}{\circ}$; $\frac{\circ}{\bullet}$, $\frac{\circ}{\bullet}$), v_4 =($\frac{\bullet}{\bullet}$; $\frac{\circ}{\circ}$, $\frac{\circ}{\bullet}$), v_5 =($\frac{\circ}{\bullet}$;$\frac{\circ}{\bullet}$,$\frac{\bullet}{\circ}$) (we ignore here the ordering of the second and the third ballot in a 3-ballot). We say that there is an edge between nodes u, v, if one of the transformations described above starts with 3-ballots such that the voter for $\mathcal{B}$ uses 3-ballot u and afterward the voter of $\mathcal{B}$ uses 3-ballot v. Let us observe that due to Steps 3A, 3B, 3C in the above procedure the graph G contains the following edges: (v_1, v_2), (v_1, v_3), (v_1, v_4) and (v_5, v_3) (due to the last remark). Hence G is connected, and so a few transformations can convert the set of 3-ballots chosen by the voters to a set of 3-ballots with the

same receipts and the same set of ballots where a person voting for $\mathcal{B}$ holds any specified receipt. So for an external observer, the anonymity set of the voter of $\mathcal{B}$ remains the whole set of voters. The only prerequisite is that the number of voters must be reasonably large to contain the following votes for $\mathcal{A}$: ($\frac{\bullet}{\circ}$; $\frac{\bullet}{\circ}$, $\frac{\circ}{\bullet}$), ($\frac{\circ}{\circ}$;$\frac{\bullet}{\bullet}$,$\frac{\bullet}{\circ}$), ($\frac{\bullet}{\circ}$;$\frac{\circ}{\bullet}$,$\frac{\bullet}{\circ}$), ($\frac{\bullet}{\bullet}$;$\frac{\circ}{\circ}$, $\frac{\bullet}{\circ}$).

Observe that if this is not true, anonymity can be broken in some cases. For instance, if there are just two voters and they choose, respectively, ($\frac{\bullet}{\bullet}$; $\frac{\circ}{\circ}$, $\frac{\circ}{\bullet}$), and ($\frac{\bullet}{\circ}$; $\frac{\bullet}{\bullet}$, $\frac{\circ}{\circ}$), then from ballots $\frac{\bullet}{\bullet}$, $\frac{\bullet}{\bullet}$, $\frac{\bullet}{\circ}$, $\frac{\circ}{\bullet}$, $\frac{\circ}{\circ}$, $\frac{\circ}{\circ}$ and receipts $\frac{\bullet}{\bullet}$, $\frac{\bullet}{\circ}$ an observer can reconstruct the votes:

- receipt $\frac{\bullet}{\bullet}$ cannot occur with another $\frac{\bullet}{\bullet}$ in the same 3-ballot, so one ballot $\frac{\bullet}{\bullet}$ occurs with receipt $\frac{\bullet}{\circ}$.
- for $\frac{\bullet}{\circ}$ and $\frac{\bullet}{\bullet}$ inside the same 3-ballot, the only option for the third ballot is $\frac{\circ}{\circ}$.

So we see that we can reconstruct completely the 3-ballots used.

Extension to the Three Candidates Case. As before, we consider the case that all voters but Alice vote for $\mathcal{A}$. We show that for any receipt W we can find a set of 3-ballots with the same set of ballots in the ballot box and the same receipts such that the vote for $\mathcal{B}$ was cast by a voter with receipt W. Let the 3-ballot cast by Alice for $\mathcal{B}$ have receipt V. Let us fix an arbitrary row of the 3-ballot of Alice, say z (if W and V have the same value in some row, then we choose one of these rows). Consider a set $\mathcal{Z}$ of all 3-ballots where the row z is filled as the row z of the 3-ballot of Alice.

Let us notice that the transformations for the two candidates case have the following property: a ballot removed from a column i of a 3-ballot is inserted into the same column i of a 3-ballot of another voter. So we can perform analogous operations on 3-ballots from the set $\mathcal{Z}$ and the resulting 3-ballots will be in $\mathcal{Z}$.

First, we make transformations on the 3-ballots from $\mathcal{Z}$ such that after the change the 3-ballot for $\mathcal{B}$ has a receipt U that agrees with W in the rows different than z. During the second stage (which is necessary if $U \neq W$), we fix a row $s \neq z$ and consider a set $\mathcal{S}$ of all 3-ballots which have row s filled in the same way as the 3-ballot for $\mathcal{B}$ in the transformed set of 3-ballots. Then on $\mathcal{S}$ we perform transformations so that finally the 3-ballot for $\mathcal{B}$ has a receipt that agrees with W in the rows different from s. Since row s does not change during transformations, finally the 3-ballot for $\mathcal{B}$ has receipt W.

One can perform all described transformations provided that $\mathcal{Z}$ and $\mathcal{S}$ contain certain 3-ballots (as for the 2-candidate case). Each of these sets contains on average $\frac{1}{3}$ of all 3-ballots, so they contain the 3-ballots necessary for transformations if the number of voters is big enough.

A similar procedure can be applied for races with more than 3 candidates, but the number of voters necessary to perform the transitions grows exponentially with the number of candidates in a run (at least for the procedure just sketched).

3 Stochastic Analysis for SBA

From now on we shall consider Threeballot scheme used for elections with a single race (for elections with different races, a voter may get different paper ballots for each race). By k we will mean the number of candidates in a race, N will denote the number of votes cast into a ballot box. We assume that N is a relatively small number (in Poland

the average value is 420, but sometimes it can drop to about 100). We assume that the list of the ballots cast is published separately for each ballot box.

For the sake of clearness of presentation we assume that the process of filling the bubbles by a voter consists of two phases. During the first phase the voter chooses one bubble per row uniformly at random, independently from the choices in different rows. Then the voter fills the chosen bubbles. During the second phase the voter fills one bubble in the row corresponding to the preferred candidate – which of two bubbles to fill is chosen uniformly at random. This bubble is called *additional*.

We shall use notation (A, B, C) for a 3-ballot consisting of the ballots A, B, C, if we do not indicate which of them serves as the receipt, and $(A; B, C)$ if A serves as the receipt.

Overview of Strauss Attack. The attack from [11] considers ballots as published on the bulletin board and a single receipt. Assume that we have a receipt R of Alice and that it contains a filled bubbles. (Of course, a is not fixed in advance; we shall see that possibility to recover the choice of Alice depends slightly on a.)

We call a ballot B *related to* ballot A, if the set of rows of B containing filled bubbles is disjoint with the set of rows of A containing filled bubbles. Let $\mathcal{R}(R)$ be the set of all ballots related to R from the list published by the poll site of Alice. Note that $\mathcal{R}(R)$ contains one of the ballots of the 3-ballot of Alice different from R.

Given a receipt R of Alice and a ballot $B \in \mathcal{R}(R)$ we look for a ballot C that has filled bubbles in all rows where R and B have no filled bubble, and that has one more filled bubble in the remaining rows. Then obviously $(R; B, C)$ is a candidate for a 3-ballot filled by Alice. The point is that it might be quite difficult to find such a C in a pool of random ballots. If there are several rows in a ballot, then the probability to fill the bubbles at random in the way described is quite low. Consequently, there are not many triples (R, B, C) of this kind, but one of them is the 3-ballot of Alice!

Even if there is more than one candidate 3-ballot for receipt R, we still have a chance to recover exactly the 3-ballot of Alice. Namely, we consider each receipt. If there is a receipt such that there is only one valid triple containing it, we know that this is a 3-ballot cast by some voter and we may remove its ballots from the list of ballots. If a removed ballot U occurred only once there, we may disregard all candidate triples computed before that contain U. Now may be for some voter only one triple remains. So we may continue in this way until we cannot find a receipt for which there is only one valid triple.

Experiments from [11] show that for an appropriate choice of parameters, if we manage to start removing ballots, then most 3-ballots can be reconstructed.

The attack of Strauss is based on the fact that two random ballots are unlikely to fit into the same 3-ballot. Obviously, this approach fails, if the number of candidates in a race is small. Failure of the attack for a small number of candidates has been already reported in [11] (green fields in the diagrams).

The main question considered here is to find a bound for the number of candidates in a race up to which Threeballot protocol is still secure regrading voter's privacy. We formulate the following necessary condition on voting secrecy:

Definition 1. *We say that the set of ballots $\mathcal{B}$ provides weak anonymity for a receipt R, if for every candidate x there are ballots $B, C \in \mathcal{B}$ such that $(R; B, C)$ is a valid 3-ballot with a vote for candidate x.*

Of course, weak anonymity is a necessary condition – if it is violated, then we can say something about preferences of at least one voter. It is not a sufficient condition, since it is not necessarily the case that there is a global assignment of ballots to 3-ballots that contains $(R; B, C)$. Sufficient conditions were investigated in Section 2.

Our main goal now is to analyze influence of the number of candidates k and the number of voters N on weak anonymity. We develop formulas that enable us to compute the expected number of 3-ballots mentioned in Definition 1. This is a step towards understanding quite intriguing combinatorial nature of the Strauss' attack. This is necessary since we observe experimentally that anonymity level drops rapidly with k and cannot be compensated easily by the increase of the number of the voters N.

The process of estimating required probabilities is quite delicate and tedious, there are many subtle dependencies that make it difficult. Therefore the way to final estimations goes through several auxiliary results. However, these lemmas show certain surprising phenomena that can be confirmed experimentally.

Analysis. For the rest of this section, we consider elections with a single race with k candidates. We also assume that the receipt R held by Alice contains a filled bubbles.

There are two ways to build a valid 3-ballot $(R; B, C)$ for a receipt R: if B and C do not belong to the same 3-ballot generated by some voter, we call $(R; B, C)$ an *incidental* 3-ballot. Otherwise we call $(R; B, C)$ a *non-incidental* 3-ballot.

Probability of Non-Incidental 3-Ballots

Lemma 1. *Let (A, B, C) be a random 3-ballot. Then the probability that R together with two ballots from (A, B, C) forms a valid 3-ballot with a vote for candidate x equals*

$$q_R = \frac{2^{k-a}}{3^{k-1}} \cdot \frac{k-a+2}{k} \ ,$$

if R contains a filled bubble in the row of x. If R does not contain a filled bubble in the row of x, this probability equals

$$q_{nR} = \frac{2^{k-a-1}}{3^{k-1}} \cdot \frac{1}{k} \ .$$

Proof. First consider the case that R contains a filled bubble in the row of x. W.l.o.g. we may assume that R contains bubbles in the first a rows and the row of x is a.

Consider the first phase of generating (A, B, C) during which a voter fills exactly one bubble in each row. W.l.o.g we may assume that A contains a filled bubble in row 1. Of course, A cannot be used together with R to form a vote for x. So we have to consider only $(R; B, C)$. If it is a vote for x, then B and C contain no filled bubbles in rows $2, \ldots, a-1$, and therefore A must contain filled bubbles in these rows. Similarly, rows $a+1, \ldots, k$ must contain bubbles in either B or C. Probability of such a configuration equals $(\frac{1}{3})^{a-2} \cdot (\frac{2}{3})^{k-a}$.

Now consider the additional bubble filled to indicate the candidate chosen by the voter while creating (A, B, C). There are two subcases. The first one is that row a already contains a filled bubble inside A. Then, in order to get a vote for x from (R, B, C),

the additional bubble must be placed in row a (hence either in ballot B or C). In the second subcase row a contains a filled bubble either in B or in C. So the additional filled bubble cannot be placed in rows 1 through $a-1$ (it would be placed either in B or in C!). If the additional bubble is placed in row a, then it should be placed in A. If it is placed in the rows $a+1$ through k, then again it should be placed in ballot A.

It follows that the probability that after filling the additional bubble we get a 3-ballot (R,B,C) as a vote for x equals

$$(\tfrac{1}{3})^{a-2}\cdot(\tfrac{2}{3})^{k-a}\cdot\left(\tfrac{1}{3}\cdot\tfrac{1}{k}+\tfrac{2}{3}\cdot(\tfrac{1}{k}\cdot\tfrac{1}{2}+\tfrac{k-a}{k}\cdot\tfrac{1}{2})\right)=\tfrac{2^{k-a}}{3^{k-1}}\cdot\tfrac{k-a+2}{k}\quad .$$

The second case is that R does not contain a filled bubble in the row of x. During the first phase of creating (A,B,C) within the rows 1 through a only the bubbles from A can be filled (otherwise the vote (R,B,C) would be not for x). Similarly, in the rows $a+1$ through k no bubble can be filled inside A. It happens with probability $(\frac{1}{3})^{a-1}\cdot(\frac{2}{3})^{k-a}$. In the second phase the voter must choose candidate x. There are two ballots with unfilled bubble in this row - A and either B or C. We succeed, if the additional ballot is not placed in A. This happens with probability $\frac{1}{k}\cdot\frac{1}{2}$. □

After considering non-incidental 3-ballots we shall see that the probability given by Lemma 1 does not contribute much to the overall probability of composing a 3-ballot from R with a vote for x. We can also see that q_{nR} is substantially smaller than q_R for the parameters values of practical interest.

Probability of Getting an Element from $\mathcal{R}(R)$. Our first goal is to estimate probability that from a random 3-ballot we get an element of $\mathcal{R}(R)$. First we inspect quite carefully the case that exactly one ballot falls into $\mathcal{R}(R)$, later we shall see that the opposite case occurs with a much lower probability.

Lemma 2. *Let (A,B,C) be a random 3-ballot. The probability that exactly one of the ballots A,B,C belongs to $\mathcal{R}(R)$ and contains exactly b filled bubbles is at most*

$$\tbinom{k-a}{b}\cdot\tfrac{2^{k-b-1}}{3^{k-1}}\cdot\tfrac{k+3b}{k}\quad .$$

Proof. There are two cases for which it is possible that (A,B,C) has the properties stated. The first one is that during the first phase the voter fills bubbles in rows 1 through a using exactly two ballots. The number of choices to do it in this way is $\binom{3}{2}\cdot(2^a-2)$ against the total number of choices equal to 3^a.

Now let us consider the second phase. W.l.o.g. we may assume that the ballot C does not contain a filled bubble in rows $1,\ldots,a$. Probability that C contains b filled bubbles after the first phase equals

$$\tbinom{k-a}{b}\cdot(\tfrac{1}{3})^{b}\cdot(\tfrac{2}{3})^{k-a-b}=\tbinom{k-a}{b}\cdot\tfrac{2^{k-a-b}}{3^{k-a}}\quad .$$

In this case the additional bubble has to be added in a row where C has already a bubble (in this case C will not be changed for sure) or in the remaining rows and C must be not chosen for filling a bubble. So the second phase does not change C with probability

$$\tfrac{b}{k}+\tfrac{k-b}{k}\cdot\tfrac{1}{2}=\tfrac{k+b}{2k}\quad .$$

After the first phase C may contain $b-1$ filled bubbles. This occurs with probability

$$\binom{k-a}{b-1} \cdot (\tfrac{1}{3})^{b-1} \cdot (\tfrac{2}{3})^{k-a-b+1} = \binom{k-a}{b-1} \cdot \frac{2^{k-a-b+1}}{3^{k-a}} \ .$$

In order to get C with b filled bubbles, all in the rows $a+1, \ldots, k$, the voter must choose one of $k-a-b+1$ rows, and choose a ballot from C for filling. This occurs with probability

$$\frac{k-a-b+1}{k} \cdot \frac{1}{2} \ .$$

We see that finally C contains b filled bubbles, all in rows $a+1, \ldots, k$, with probability

$$\begin{aligned} &\binom{k-a}{b} \cdot \frac{2^{k-a-b}}{3^{k-a}} \cdot \frac{k+b}{2k} + \binom{k-a}{b-1} \cdot \frac{2^{k-a-b+1}}{3^{k-a}} \cdot \frac{k-a-b+1}{2k} \\ &= \binom{k-a}{b} \cdot \frac{2^{k-a-b}}{3^{k-a}} \cdot \left(\frac{k+b}{2k} + \frac{b}{k-a-b+1} \cdot 2 \cdot \frac{k-a-b+1}{2k} \right) \\ &= \binom{k-a}{b} \cdot \frac{2^{k-a-b-1}}{3^{k-a}} \cdot \frac{k+3b}{k} \ . \end{aligned}$$

Taking into account the probability of filling bubbles in exactly two ballots (from A, B, C) in the rows $1, \ldots, a$, we get the final probability of success in the first case considered:

$$\frac{2^a-2}{3^{a-1}} \cdot \binom{k-a}{b} \cdot \frac{2^{k-a-b-1}}{3^{k-a}} \cdot \frac{k+3b}{k} \ . \tag{1}$$

The second case is that during the first phase the voter fills bubbles in only one ballot in rows 1 through a. This happens with probability $\frac{1}{3^{a-1}}$. For the sake of analysis we reverse a little the process of generating (A, B, C). First the voter fills one bubble in each of the rows $1, \ldots, a$; then she chooses where to fill a bubble pointing to the candidate chosen; finally she fills one bubble in each of the rows $a+1, \ldots, k$. So in this case the situation described by the lemma occurs when the voter chooses one of the candidates $1, \ldots, a$ – then the second ballot gets excluded from $\mathcal{R}(R)$. So the overall probability of the event of interest equals

$$\frac{1}{3^{a-1}} \cdot \frac{a}{k} \cdot \binom{k-a}{b} \cdot (\tfrac{1}{3})^{b} \cdot (\tfrac{2}{3})^{k-a-b} = \binom{k-a}{b} \cdot \frac{2^{k-a-b}}{3^{k-1}} \cdot \frac{a}{k} \ . \tag{2}$$

It is easy to see that the sum of expressions from (1) and (2) is smaller than

$$\binom{k-a}{b} \cdot \frac{2^{k-b-1}}{3^{k-1}} \cdot \frac{k+3b}{k} \ . \qquad \square$$

Lemma 3. *Let (A, B, C) be a random 3-ballot. The probability that exactly two of the ballots A, B, C belong to $\mathcal{R}(R)$ and at least one of them has b filled bubbles is at most*

$$\frac{2^{k-a-b}}{3^{k-1}} \cdot \binom{k-a}{b} \cdot \frac{k-a+2b}{k} \ .$$

Proof. Assume that during the first phase the bubbles in rows $1, \ldots, a$ are filled first. They must go to the same ballot, w.l.o.g. assume that it is A. This happens with probability $(\frac{1}{3})^{a-1}$.

We estimate the probability that (A, B, C) fulfills the properties stated and that B contains b filled bubbles. There are two cases: the first one is that during the first phase B gets b filled bubbles. During the second phase, if the voter chooses one of the b rows,

where B already contains filled bubbles, then we get a 3-ballot with the properties stated. If the voter chooses one of the rows $1, \ldots, a$, then either B or C will not belong to $\mathcal{R}(R)$. If the voter chooses one of $k-a-b$ remaining rows, then with probability $\frac{1}{2}$ the voter fills a bubble in A and not in B. So the overall probability of getting into a proper configuration in this case equals

$$(\tfrac{1}{3})^{a-1} \cdot \tbinom{k-a}{b} \cdot (\tfrac{1}{3})^{b} \cdot (\tfrac{2}{3})^{k-a-b} \cdot (\tfrac{k-a-b}{k} \cdot \tfrac{1}{2} + \tfrac{b}{k}) \ . \tag{3}$$

Now assume that during the first phase $b-1$ bubbles are filled inside B. During the second phase one bubble must be filled in B in one of $k-a-(b-1)$ rows. So the overall probability of getting into a proper configuration in this case equals

$$(\tfrac{1}{3})^{a-1} \cdot \tbinom{k-a}{b-1} \cdot (\tfrac{1}{3})^{b-1} \cdot (\tfrac{2}{3})^{k-a-b+1} \cdot \tfrac{k-a-b+1}{k} \cdot \tfrac{1}{2} \ . \tag{4}$$

By summing up the expressions (3) and (4) we get the expression

$$\tfrac{2^{k-a-b}}{3^{k-1}} \cdot \tfrac{1}{2k} \cdot \left(\tbinom{k-a}{b} \cdot (k-a-b+2b) + \tbinom{k-a}{b-1} \cdot (k-a-b+1) \right) \ .$$

Since $\binom{k-a}{b-1} = \binom{k-a}{b} \cdot \frac{b}{k-a-b+1}$, the last expression equals

$$\tfrac{2^{k-a-b}}{3^{k-1}} \cdot \tfrac{1}{2k} \cdot \tbinom{k-a}{b} \cdot (k-a+2b) \ .$$

Since also C may have the form required, we multiply this probability by 2 in order to get the estimation from the lemma. □

Probability of Incidental 3-Ballots. Now we turn our attention into probability of composing a valid 3-ballot from receipt R, a ballot $B \in \mathcal{R}(R)$, and a random 3-ballot (U, V, W).

Lemma 4. *Let $B \in \mathcal{R}(R)$ and B contain b filled bubbles. Let Z be a ballot from a ballot different from those containing originally R and B. Let x be a candidate such that row x contains a filled bubble in either R or B. Then with the probability*

$$\tfrac{2^{a+b-2}}{3^k} \cdot \tfrac{4k-3a-3b+3}{k}$$

(R, B, Z) forms a valid 3-ballot with a vote in row x.

Proof. Let us use the following terminology: By *filled rows* we mean the $a+b$ rows where either R or B contains a filled bubble, except row x. All remaining rows are called *unfilled* rows.

Note that ballot Z must contain filled bubbles in the unfilled rows and no filled bubble in the filled rows. It must also contain a filled bubble in row x. There are two ways to get such a configuration. The first case is that Z gets filled bubble in all unfilled rows and in row x during the first phase of creating (U, V, W). This occurs with probability

$$(\tfrac{1}{3})^{k-(a+b-1)} \cdot (\tfrac{2}{3})^{a+b-1} = \tfrac{2^{a+b-1}}{3^k} \ .$$

Then, during the second phase, the voter must not put an additional bubble in Z, which occurs with probability $1 - \frac{a+b-1}{k} \cdot \frac{1}{2}$.

The second case is that Z receives all but one filled bubble during the first phase, and it gets the missing filled bubble in the second phase. This case occurs with probability

$$(k-(a+b-1))\cdot(\tfrac{1}{3})^{k-a-b}\cdot(\tfrac{2}{3})^{a+b}\cdot\tfrac{1}{k}\cdot\tfrac{1}{2}=\tfrac{2^{a+b-1}}{3^k}\cdot\tfrac{k-a-b+1}{k}\ .$$

We see that together the probability of getting such a ballot Z equals

$$\tfrac{2^{a+b-1}}{3^k}\cdot\tfrac{2k-a-b+1}{2k}+\tfrac{2^{a+b-1}}{3^k}\cdot\tfrac{k-a-b+1}{k}=\tfrac{2^{a+b-2}}{3^k}\cdot\tfrac{4k-3a-3b+3}{k}\ .$$ □

Remark 1. We can immediately see that it is impossible to form a valid 3-ballot vote from R, B for a candidate y that corresponds to an unfilled row.

Now we estimate the probability that we can build an incidental 3-ballot with a vote for x from two random 3-ballots.

Lemma 5. *Let x be a row where R has a filled bubble. Then probability that from two random 3-ballots B_1, B_2 we can build a vote (R,B,C), where $B\in\mathcal{R}(R)$, for candidate x is at most*

$$p_R=\tfrac{2^{2k-3}}{3^{2k-2}}\cdot(4c_0+2c_1\cdot(k-a)-c_2\cdot(k-a)(k-a+1))\ ,$$

where $c_0=(1+\frac{1}{2^{a+1}})\frac{4k-3a+3}{k}$, $c_1=\frac{3(4k-3a+3)}{k^2}-\frac{3}{k}(1+\frac{1}{2^{a+1}})$, $c_2=\frac{9}{k^2}$.

Proof. First observe that according to Lemmas 2, 3 and 4, the probability of getting a 3-ballot (R,B,C) with the properties claimed such that B contains exactly b filled bubbles is at most

$$\begin{aligned}&\left(\tbinom{k-a}{b}\cdot\tfrac{2^{k-b+1}}{3^{k-1}}\cdot\tfrac{k+3b}{k}+\tfrac{2^{k-a-b}}{3^{k-1}}\cdot\tbinom{k-a}{b}\cdot\tfrac{k-a+2b}{k}\right)\cdot\tfrac{2^{a+b-2}}{3^k}\cdot\tfrac{4k-3a-3b+3}{k}\cdot 3\\&=\tbinom{k-a}{b}\cdot\tfrac{2^{k+a-1}}{3^{2k-2}}\cdot(\tfrac{k+3b}{k}+\tfrac{1}{2^{a+1}})\cdot\tfrac{4k-3a-3b+3}{k}\end{aligned}\tag{5}$$

Our goal is to estimate the sum of expressions (5) for all possible values of b:

$$\begin{aligned}&\textstyle\sum_{b=0}^{k-a}\left(\binom{k-a}{b}\cdot\frac{2^{k+a-1}}{3^{2k-2}}\cdot(\frac{k+3b}{k}+\frac{1}{2^{a+1}})\cdot\frac{4k-3a-3b+3}{k}\right)\\&=\tfrac{2^{k+a-1}}{3^{2k-2}}\cdot\textstyle\sum_{b=0}^{k-a}\binom{k-a}{b}\cdot\left(c_0+c_1\cdot b-c_2\cdot b^2\right),\end{aligned}\tag{6}$$

where

$$c_0=(1+\tfrac{1}{2^{a+1}})\tfrac{4k-3a+3}{k},\qquad c_1=\tfrac{3(4k-3a+3)}{k^2}-\tfrac{3}{k}(1-\tfrac{1}{2^{a+1}}),\qquad c_2=\tfrac{9}{k^2}\ .$$

Recall that

$$\begin{aligned}&\textstyle\sum_{i=0}^{n}\binom{n}{i}=2^n,\qquad\sum_{i=0}^{n}\binom{n}{i}\cdot i=n\cdot 2^{n-1},\quad\text{and}\\&\textstyle\sum_{i=0}^{n}\binom{n}{i}\cdot i^2=2^{n-2}\cdot n\cdot(n+1).\end{aligned}$$

Hence the expression (6) can be rewritten as follows:

$$\begin{aligned}&\tfrac{2^{k+a-1}}{3^{2k-2}}\cdot\left(2^{k-a}\cdot c_0+(k-a)\cdot 2^{k-a-1}\cdot c_1+(k-a)(k-a+1)2^{k-a-2}\cdot c_2\right)\\&=\tfrac{2^{2k-3}}{3^{2k-2}}\cdot(4\cdot c_0+(k-a)\cdot 2\cdot c_1+(k-a)(k-a+1)\cdot c_2)\ .\end{aligned}\tag{7}$$

□

A nice feature of approximation (7) is that its main term $\frac{2^{2k-3}}{3^{2k-2}}$ depends neither on a nor on b. This yields a rough estimation of probability of forming an incidental 3-ballot from R.

Now we turn our attention to the case when R does not contain a filled bubble in the row of candidate x. The following lemma can be shown similarly to Lemma 5.

Lemma 6. *Let x be a row where R has no filled bubble. Then the probability that from two random 3-ballots B_1, B_2 we can build a vote (R, B, C), where $B \in \mathcal{R}(R)$, for candidate x is at most*

$$p_{nR} = \frac{2^{2k-4}}{3^{2k-2}} \cdot (4c_0 + 2c_1 \cdot (k-a+1) - c_2 \cdot (k-a) \cdot (k-a+3)) \quad ,$$

where $c_0 = (1 + \frac{1}{2^{a+1}})\frac{4k-3a+3}{k}$, $c_1 = \frac{3(4k-3a+3)}{k^2} - \frac{3}{k}(1 + \frac{1}{2^{a+1}})$, $c_2 = \frac{9}{k^2}$.

Proof. The proof is similar as the proof of Lemma 5. The difference is that instead of (5) we have

$$\binom{k-a}{b} \cdot \frac{b}{k-a} \cdot \frac{2^{k+a-1}}{3^{2k-2}} \cdot (\frac{k+3b}{k} + \frac{1}{2^{a+1}}) \cdot \frac{4k-3a-3b+3}{k}$$

(the term $\frac{b}{k-a}$ is due to the additional condition in Lemmas 2, 3 that the ballot from $\mathcal{R}(R)$ has to contain a filled bubble in row x). So instead of (6) we have

$$\begin{aligned} p_{nR} &= \sum_{b=0}^{k-a} \left(\binom{k-a}{b} \cdot \frac{2^{k+a-1}}{3^{2k-2}} \cdot \frac{b}{k-a} \cdot (\frac{k+3b}{k} + \frac{1}{2^{a+1}}) \cdot \frac{4k-3a-3b+3}{k} \right) \\ &= \frac{2^{k+a-1}}{3^{2k-2}} \cdot \frac{1}{k-a} \cdot \sum_{b=0}^{k-a} \binom{k-a}{b} \cdot \left(c_0 \cdot b + c_1 \cdot b^2 - c_2 \cdot b^3 \right), \end{aligned} \tag{8}$$

Recall that $\sum_{i=1}^{n} \binom{n}{i} \cdot i^3 = 2^{n-3} \cdot n^2 \cdot (n+3)$. Hence

$$p_{nR} = \frac{2^{2k-4}}{3^{2k-2}} \cdot (4c_0 + 2c_1 \cdot (k-a+1) - c_2 \cdot (k-a) \cdot (k-a+3)) \quad . \qquad \square$$

Comparing the results of Lemma 5 and 6 we see that $p_R \approx 2 \cdot p_{nR}$. This has some strange consequences: a conscious voter may be tempted to hide *his* choice and avoid unbalance between p_R and $2 \cdot p_{nR}$ by taking $a = k$. However, if all voters behave like this, then it becomes evident which ballots in the ballot box correspond to the receipts. Then an adversary may safely replace some of the ballots in the ballot box.

Estimating the Number of Possible Votes. The results of the previous subsections lead to quite precise estimations of the expected number of valid 3-ballots composing a vote for a given candidate x.

Theorem 1. *Let R be a receipt with a filled bubbles in k candidate race and N votes cast. If R contains a filled bubble in row x, then the expected number of non-incidental 3-ballots with a vote for x is at most*

$$\frac{2^{k-a}}{3^{k-1}} \cdot \frac{k-a+2}{k} \cdot (N-1)$$

and the expected number of incidental 3-ballots with a vote for x is at most

$$\frac{2^{2k-4}}{3^{2k-2}} \cdot (4c_0 + 2c_1 \cdot (k-a) - c_2 \cdot (k-a)(k-a+1)) \cdot (N-1) \cdot (N-2) \quad ,$$

2 where $c_0 = (1 + \frac{1}{2^{a+1}})\frac{4k-3a+3}{k}$, $c_1 = \frac{3(4k-3a+3)}{k^2} - \frac{3}{k}(1 + \frac{1}{2^{a+1}})$, $c_2 = \frac{9}{k^2}$.

If R does not contain a filled bubble in row x, then the expected number of non-incidental 3-ballots with a vote for x is at most

$$\frac{2^{k-a-1}}{3^{k-1}} \cdot \frac{1}{k} \cdot (N-1)$$

and the expected number of incidental 3-ballots with a vote for x is at most

$$\frac{2^{2k-5}}{3^{2k-2}} \cdot (4c_0 + 2c_1 \cdot (k-a+1) - c_2 \cdot (k-a) \cdot (k-a+3)) \cdot (N-1) \cdot (N-2) \ .$$

Conclusions. The formulas for expected values from Theorem 1 depend on two main terms: $(\frac{2}{3})^k$ and N. While $(\frac{2}{3})^k$ and $(\frac{2}{3})^{2k}$ reduce the number of ballots, the terms $N-1$ and $(N-1)(N-2)$ may compensate for this. We see that the phase transition point is close to $N \approx (\frac{3}{2})^k$, however the neglected terms are important for small parameters.

In the tables below we inspect some values obtained from Theorem 1. We see from Table 1 that non-incidental ballots are negligible for voter's privacy. Also, if we increase the number of filled bubbles on a receipt, then voter's privacy decreases. One can also see that the case $k \geq 9$ is hopeless from privacy point of view. On the other hand, it seems that $k \leq 7$ might provide sufficient privacy level due to high expected values.

Table 1. Upper estimation for the expected number of non-incidental 3-ballots for candidate x for a receipt R with a filled bubbles, when R does not contain a filled bubble in a row x, $N = 100$

	$a=1$	$a=2$	$a=3$	$a=4$	$a=5$	$a=6$	$a=7$
$k=5$	1.96	.98	.49	.24	.12		
$k=6$	1.08	.54	.27	.014	.068	.034	
$k=7$	.62	.31	.16	.077	.039	.019	.0097

Table 2. Upper estimation for the expected number of incidental 3-ballots for candidate x for a receipt R with a filled bubbles, when R does not contain a filled bubble in a row x

	$a=1$	$a=2$	$a=3$	$a=4$	$a=5$	$a=6$	$a=7$	$a=8$	$a=9$	$a=10$
$N=100$										
$k=5$	1250	934	688	494	340					
$k=7$	248	199	160	127	100	76	57			
$k=9$	49	41	34	29	24	20	16	13	10	
$k=10$	22	18.6	15.9	13.6	11.6	9.87	8.27	6.83	5.51	4.41
$N=50$										
$k=7$	60	48	39	31	24	18	14			
$k=9$	11.9	9.97	8.39	7.07	5.92	4.90	3.99	3.18	2.48	

References

1. Appel, A.W.: How to defeat Rivest's ThreeBallot voting system. Draft (October 2006), http://www.cs.princeton.edu/ appel/papers/DefeatingThreeBallot.pdf

2. Araújo, R., Custódio, R.F., van de Graaf, J.: A Verifiable Voting Protocol based on Farnel. In: Proceedings of Workshop On Trustworthy Elections(WOTE) (2007), http://research.microsoft.com/conferences/WOTE2007/papers/07.pdf
3. Belote, G., Jones, H., Juang, J.: Threeballot in the field. Draft (2006), http://theory.lcs.mit.edu/classes/6.857/projects/threeBallotPaper.pdf
4. Chaum, D., Ryan, P.Y.A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
5. Clark, J., Essex, A., Adams, C.: On the Security of Ballot Receipts in E2E Voting Systems. In: Proceedings of Workshop On Trustworthy Elections (WOTE) (2007), http://research.microsoft.com/conferences/WOTE2007/papers/08.pdf
6. Gogolewski, M., Klonowski, M., Kutyłowski, M., Kubiak, P., Lauks, A., Zagórski, F.: Kleptographic Attacks on E-voting Schemes. In: Müller, G. (ed.) ETRICS 2006. LNCS, vol. 3995, pp. 494–508. Springer, Heidelberg (2006)
7. Henry, K., Stinson, D.R., Sui, J.: The Effectiveness of Receipt-Based Attacks on ThreeBallot. Cryptology ePrint Archive, http://eprint.iacr.org/2007/287
8. Hosp, B., Popoveniuc, S.: An introduction to Punchscan. In: Threat Analysis for Voting System. A Workshop on Rating Voting Methods VSRW 2006 (2006), http://vote.cs.gwu.edu/vsrw2006/papers/9.pdf
9. Rivest, R.L.: The Threeballot voting system. Draft, Version 10/1/06 (2006)
10. Rivest, R., Smith, W.: Three Voting Protocols: ThreeBallot, VAV, and Twin. In: EVT. Proceedings of USENIX/ACCURATE Electronic Voting Technology Workshop (2007)
11. Strauss, Ch.: A critical review of the triple ballot voting system. part 2: Cracking the triple ballot encryption. Draft (October 8, 2006), http://www.cs.princeton.edu/~appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf
12. Strauss, Ch.: The trouble with triples: A critical review of the triple ballot (3ballot) scheme, part 1. Draft (October 5, 2006), http://www.cs.princeton.edu/~appel/voting/Strauss-TroubleWithTriples.pdf
13. Top-To-Bottom Review, http://www.sos.ca.gov/elections/elections_vsr.htm

Practical Deniable Encryption*

Marek Klonowski, Przemysław Kubiak, and Mirosław Kutyłowski

Institute of Mathematics and Computer Science,
Wrocław University of Technology
{Marek.Klonowski, Przemyslaw.Kubiak, Miroslaw.Kutylowski}@pwr.wroc.pl

Abstract. A party using encrypted communication or storing data in an encrypted form might be forced to show the corresponding plaintext. It may happen for law enforcement reasons as well as for evil purposes. Deniable encryption scheme introduced by Canetti et al. shows that cryptography can be used against revealing information: the owner of the data may decrypt it in an alternative way to a harmless plaintext. Moreover, it is impossible to check if there is another hidden plaintext.

The scheme of Canetti is inefficient in the sense that it is a special purpose scheme and using it indicates that there is some hidden message inside. We show that deniable encryption can be implemented in a different way so that it does not point to exploiting deniable encryption. Moreover, it is quite straightforward, so it can be used for both good and evil purposes.

Apart from that we show that even the special purpose original scheme can be extended to allow, in some circumstances, any "depth" of deniability.

Keywords: deniable encryption.

1 Introduction

Regular encryption schemes are aimed at hiding contents of encrypted data. Many of them meet very high security against an adversary that is given a ciphertext and would like to learn the corresponding plaintext. However, in the real world situation might be more complex. After gaining access to encrypted data, an adversary can demand presenting decryption keys or the corresponding plaintext (in the latter case the adversary may demand a proof that the plaintext is the right one). This may happen due to law enforcement procedures as well as a part of a criminal action. In many countries citizens are obliged by law to reveal these data on demand of appropriate authorities. One cannot simply refuse claiming that the decryption keys are lost or forgotten – in such a case the person might be considered as guilty by default. For most encryption schemes (ElGamal, RSA, DES ...), neither a receiver (we call him Bob), nor the sender (Alice) of encrypted data can cheat and disclose an incorrect plaintext. An incorrect decryption key would give senseless data.

* The paper is partially supported by EU within the 6th Framework Programme under contract 001907 (DELIS).

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 599–609, 2008.

The main step for solving the problems mentioned is the idea of deniable encryption proposed in [3]. In this scheme, the sender can reveal fake parameters like random strings and private keys that yield a plaintext m_f, instead of the original plaintext m. More precisely, if a ciphertext $c = E(m, r)$ of message m is composed with parameter r (which is the key and, may be, some additional parameters), then the goal is to present r_f and $m_f \neq m$, such that $c = E(m_f, r_f)$. Moreover, m_f should be a reasonable plaintext, in order to convince the adversary. The protocol for finding such m_f and r_f is called a *faking algorithm*, while the whole cryptosystem is named *deniable encryption scheme*. In case of asymmetric schemes, we can distinguish between *sender-deniable*, *receiver-deniable* and *sender-and-receiver deniable* schemes. However, it is shown in [3] that one can convert any sender-deniable encryption scheme into a receiver-deniable scheme, and vice versa. In that conversion the scheme is transformed into a two-phase one, with the inverted roles of the sender and the receiver in the first phase. In the same paper it is also presented how to construct a sender-and-receiver deniable scheme on top of a sender-deniable scheme under some additional assumptions.

Applications. Deniable encryption schemes can be used not only in the basic scenario described. They might be useful for multiparty secure computation [4], electronic voting protocols and many other problems (cf. [3], [2]). Deniable encryption can be also combined with ring signatures [10].

Our Contribution. In this paper we present a few schemes aimed at providing deniable encryption. First, we present a modification of the protocol of Canetti et al. Our modification, if not used on a large scale, allows to hide messages at any depth. We also present other practical solutions – one can be regarded as an extension of the one-time-pad idea, other are based on the ElGamal encryption scheme.

2 Deniable Encryption by Canetti et al.

Public-Key Deniable Encryption. According to [3] public-key sender-deniable encryption scheme should fulfill the following requirements:

1. Only the receiver possesses the decryption key.
2. With overwhelming probability the value decrypted by the receiver contains no flipped bits, i.e. is exactly the same as the one that was being encrypted.
3. The protocol should be semantically secure.
4. The sender should have an efficient *faking algorithm* ϕ such that for a given quadruple (m, r, c, m_f), where c is a transcript of sending ciphertext of m using random parameter r and m_f is an arbitrary, but plausible message, can produce r_f such that c is a transcript of sending message m_f using r_f. In other words, it means that the sender can convince the attacker that m_f might be the message sent.

Shared-Key Deniable Encryption. Although the authors of [3], [2] concentrate on the public-key deniability, they give two examples of shared key encryption. The first one is the obvious one-time pad, the second one (given in [2]) is a *plan-ahead* deniable scheme. That means that the sender chooses m_f (or generally $\ell - 1$ such fake messages, for some $\ell \geq 2$) at the time of encryption, and all these information is put into the resulting ciphertext. Thus the size of the ciphertext is proportional to ℓ. Obviously, plan-ahead shared-key deniable encryption is weaker than the one in which m_f might be chosen at the time of coercion (for convenience we call such a scheme m_f*-ad-hoc* deniable encryption).

A Translucent Set. The public key scheme from [3] is based on so called *translucent set*. $\mathcal{S}_t$ is called a *translucent set* if all the following conditions are satisfied:

- $\mathcal{S}_t \subset \{0,1\}^t$ and $|\mathcal{S}_t| < 2^{t-k}$, for sufficiently large k (say $k = 40$).
- It is easy to find a random element $x \in \mathcal{S}_t$.
- Given $x \in \{0,1\}^t$ and trapdoor information d_t, it is easy to check if $x \in \mathcal{S}_t$.
- Without d_t it is computationally infeasible to decide whether $x \in \mathcal{S}_t$.

Construction of a Translucent Set. The following construction from [3] is based on a trapdoor permutation $f : \{0,1\}^s \to \{0,1\}^s$ and its hard-core predicate $B : \{0,1\}^s \to \{0,1\}$.

Let $t = s + k$. Represent each $x \in \{0,1\}^t$ as $x = x_0||b_1||b_2|| \ldots ||b_k$, where $x_0 \in \{0,1\}^s$ is followed by k bits. Then the translucent set is defined as:

$$\mathcal{S}_t = \left\{ x = x_0||b_1||b_2|| \ldots ||b_k \in \{0,1\}^{s+k} \middle| \, (\forall_{i \leq k})\, B(f^{-i}(x_0)) = b_i \right\}.$$

The trapdoor information d_t plays the role of a private key.

The Parity Scheme. In this sender-deniable scheme n randomly chosen t-bit strings are used to encode a single bit. Let $\mathcal{S}_t \subset \{0,1\}^t$ be a translucent set. According to [3] elements drawn uniformly from $\mathcal{S}_t$ (respectively, from $\{0,1\}^t$) are named $\mathcal{S}$-elements (respectively, $\mathcal{R}$-elements). Note that the probability that an $\mathcal{R}$-element is also an $\mathcal{S}$-element is 2^{-k}.

Encryption: To encrypt bit b a random number $i \in 0, \ldots, n$ such that $i = b \bmod 2$ is chosen. A ciphertext of b consists of i subsequent $\mathcal{S}$-elements followed by $n - i$ $\mathcal{R}$-elements.

Decryption: Return b according to parity of the number of $\mathcal{S}$-elements in the ciphertext.

Honest Opening: The sender reveals random choices used during encoding.

Dishonest Opening: The sender claims that she has chosen $i - 1$ elements instead of i. It changes the parity, and consequently the encoded bit is flipped. If $i = 0$, cheating fails.

It has been proved in [3], [2] that the parity scheme is a $\frac{4}{n}$-sender deniable encryption scheme, which means that the probability of a successful attack of a coercer vanishes linearly in the security parameter n. However, both in [3],

[2] a "flexibly deniable scheme" is given for $n = 2$. Construction of that scheme ensures that the probability of a successful attack is negligible even for such a small value of n. The construction is as follows: for deniable encryption only the sequences $C_1 = \{\mathcal{S}, \mathcal{R}\}$, $C_0 = \{\mathcal{S}, \mathcal{S}\}$ are chosen, and at the time of coercion each of them might be opened as any of $T_0 = \{\mathcal{R}, \mathcal{R}\}$, $T_1 = \{\mathcal{S}, \mathcal{R}\}$ (more precisely, C_0, C_1 will always be opened as T_0, T_1, and honest opening means that C_i is opened as T_i for $i = 0, 1$). Clearly, to make the opening believable the pair T_0, T_1 cannot be precluded from being used for encryption instead of the pair C_0, C_1 – the sender does not have to preserve deniability of a particular plaintext. It is also said in the paper that in general such an approach allows to construct efficient deniable schemes.

3 Our Solutions

Note that for $n = 2$ the above flexibly deniable scheme has some deficiency. Namely, $\{\mathcal{S}, \mathcal{S}\}$ never appears in the opening phase. Accordingly, after opening sufficiently many bits the coercer will be convinced that the flexibly deniable scheme is used.

Assume that going through the pair of plaintexts (m, m_f) and reading the pair (b, b_f) of corresponding bits we get all four cases equally often. Then for $n > 2$ it is easy to design a protocol in which each of $n + 1$ n-element strings $\{\mathcal{R}, \mathcal{R}, \ldots, \mathcal{R}\}$, $\{\mathcal{S}, \mathcal{R}, \ldots, \mathcal{R}\}, \ldots, \{\mathcal{S}, \mathcal{S}, \ldots, \mathcal{S}\}$ occurs equally often in the opening phase (from now on we assume that C_0 contains exactly $2 \cdot \lfloor \frac{n}{2} \rfloor$ $\mathcal{S}$-elements, whereas C_1 contains $2 \cdot \lfloor \frac{n-1}{2} \rfloor + 1$ of them, and exactly one of C_0, C_1 contains a single $\mathcal{R}$-element at the end). For example for $n = 3$ bit $b = 0$ would be encrypted as $C_0 = \{\mathcal{S}, \mathcal{S}, \mathcal{R}\}$, and $b = 1$ as $C_1 = \{\mathcal{S}, \mathcal{S}, \mathcal{S}\}$. Then $b_f = 0$ would be opened as $\{\mathcal{R}, \mathcal{R}, \mathcal{R}\}$ for C_0 and $\{\mathcal{S}, \mathcal{S}, \mathcal{R}\}$ for C_1, and $b_f = 1$ as $\{\mathcal{S}, \mathcal{R}, \mathcal{R}\}$ for C_0 and $\{\mathcal{S}, \mathcal{S}, \mathcal{S}\}$ for C_1. So the coercer cannot distinguish usage of the flexibly deniable scheme from honest application of the parity scheme. The increase of n (reasonable for the scheme that appears to be the parity one) enlarges the volume of the ciphertext. This might be utilized by a solution from Subsect. 3.4.

3.1 A Nested Construction

In this subsection we present a public-key, m_f-ad-hoc sender-deniable encryption scheme that partially extends the protocol from [3]. Our motivation is as follows: this scheme provides deniable encryption, however a coercer also knows that it is a deniable encryption scheme. So after obtaining a plaintext m_f he can continue to demand the real one. In such a situation a good solution for Alice would be to confess to sending a "slightly" banned contents, instead of the original contents. Such an approach seems to be much more convincing than the one from the original scheme.

Description of the Modified Protocol. Assume that $f, f^\star$ are two independent trapdoor permutations such that

$$f : \{0,1\}^{s+k} \to \{0,1\}^{s+k}, \quad f^\star : \{0,1\}^s \to \{0,1\}^s.$$

Denote the trapdoors of f, $f^\star$ as d_t, $d_t^\star$ respectively. Let $t = s + 2k$. Represent each $x \in \{0,1\}^t$ as $x = x_0||b_1^\star|| \ldots ||b_k^\star||b_1|| \ldots ||b_k$, where $x_0 \in \{0,1\}^s$ is followed by $2k$ bits. Then we define translucent sets as:

$$\begin{aligned}\mathcal{S}_t^\star = \{x = x_0||b_1^\star|| \ldots ||b_k^\star||b_1|| \ldots ||b_k \in \{0,1\}^{s+2k} \\ \mid\ (\forall_{i\le k})\ B((f^\star)^{-i}(x_0)) = b_i^\star\},\\ \mathcal{S}_t\ = \{x = x_0||b_1^\star|| \ldots ||b_k^\star||b_1|| \ldots ||b_k \in \{0,1\}^{s+2k} \\ \mid\ (\forall_{i\le k})\ B(f^{-i}(x_0||b_1^\star|| \ldots ||b_k^\star)) = b_i\}.\end{aligned}$$

Define $\mathcal{S}^\star$-elements as elements of $\mathcal{S}_t^\star$ chosen at random. We say that $x \in \{0,1\}^{s+2k}$ is an $\mathcal{R}^\star$-element, if the first $s + k$ bits of x are drawn uniformly at random from $\{0,1\}^{s+k}$. Suppose that the sender knows d_t (to be able to calculate bits b_i when the string $x_0||b_1^\star|| \ldots ||b_k^\star$ is already determined by $f^\star$). Then for each pair of elements $(U, V) = (\mathcal{S}^\star, \mathcal{S}), (\mathcal{S}^\star, \mathcal{R}), (\mathcal{R}^\star, \mathcal{S}), (\mathcal{R}^\star, \mathcal{R})$ she is able to find x such that $x \in U \cap V$.

The construction reminds Russian dolls: $x_0||b_1^\star|| \ldots ||b_k^\star$, that is the internal doll, can be claimed to be a random string used for the original scheme. The whole construction (we will refer to it as to Matryoshka) describes a single element from the Cartesian product $\{\mathcal{S}^\star, \mathcal{R}^\star\} \times \{\mathcal{S}, \mathcal{R}\}$. The external doll, build on the basis of f, is to protect the internal one build on the basis of $f^\star$. As in the original scheme, f must be a *public key* trapdoor permutation, otherwise the coercer would know that the sender could have inverted f (in the case of a public key trapdoor permutation the sender might plausibly deny that she knows f^{-1}). Consequently, $s + k$ must be considerably large, thus s must be relatively large. This gives some space for the extension described subsequently.

Note that some attention must be paid to implementation details. If for example f is a permutation based on the factorization problem, then $f : \mathbb{Z}_N \to \mathbb{Z}_N$ for some composite modulus N. Since $b_i^\star$ are single bits attached to x_0, values $y = x_0||b_1^\star|| \ldots ||b_k^\star$ are usually not uniformly distributed in $\mathbb{Z}_N$. These values are taken as arguments of f to calculate $b_i = B(f^{-i}(x_0))$ when an $\mathcal{S}$-element is to be encoded. Consequently, if x should be opened as an $\mathcal{S}$-element, then y must be revealed. Hence the non-uniform distribution of values y would be evident for the coercer. To hide this we slightly modify the definition of $\mathcal{S}_t$:

$$\begin{aligned}\mathcal{S}_t = \{x = E(x_0||b_1^\star|| \ldots ||b_k^\star)\,||b_1|| \ldots ||b_k \in \mathbb{Z}_N||\{0,1\}^k \\ \mid\quad (\forall_{i\le k})\ B(f^{-i}(E(x_0||b_1^\star|| \ldots ||b_k^\star))) = b_i\},\end{aligned}$$

where E is RSA encryption modulo N with the "public" asymmetric exponent known to the sender. Then the most significant bits of $x_0||b_1^\star|| \ldots ||b_k^\star$ are not available to the coercer (cf. [11, Chap. 4]). Now let us describe details of the scheme:

Preliminaries: At the beginning trapdoor d_t must be established between Alice and Bob. Let m, m_f, $m^\star$, $m_f^\star$ be pairs of plaintexts, where m is only slightly banned, and $m^\star$ is strictly prohibited (m_f, $m_f^\star$ can be devised at the time of coercion). Assume that bit-lengths of the plaintexts are equal. For $n > 2$ define strings $C_0^\star$, $C_1^\star$ like C_0, C_1 from the beginning of Sect. 3, but

use $S^\star$ and $R^\star$ instead of S and R. (In fact, we should not preclude usage of strings like T_0, T_1, but we skip this issue for brevity). Let the length of the strings $C^\star_{b^\star}$, C_b for $b^\star, b \in \{0,1\}$, be the same.

Encryption: To deniably encrypt both $m^\star$ and m the sender encodes $\lfloor \log_2 m^\star \rfloor + 1 = \lfloor \log_2 m \rfloor + 1$ pairs of bits $(b^\star, b)$. Each pair determines a pair $(C^\star_{b^\star}, C_b)$ of strings of length n. The pair $(C^\star_{b^\star}, C_b)$ is encoded as a sequence of n separable Matryoshkas. In each consecutive Matryoshka a single element (belonging to the Cartesian product $\{\mathcal{S}^\star, \mathcal{R}^\star\} \times \{\mathcal{S}, \mathcal{R}\}$) from a consecutive position in the pair of strings is encoded.

Decryption: is straightforward.

Dishonest Opening: In the case of coercion the sender might "reveal" m_f. If she is coerced harder, she can even reveal m (because of the form of C_0, C_1 the coercer is convinced that m is genuine). If despite of this the coercion is still continued, Alice might reveal f^{-1}, the true arguments for encryption E and a fake $m^\star_f$. Note that according to the encoding procedure $m^\star_f$ looks like honest opening of the message encoded according to the parity scheme.

Nested contents. One can easily see that for sufficiently large t it is possible to embed more than two translucent sets into $\{0,1\}^t$ in a straightforward manner. Let us note that we can simply iteratively use the above idea. Then the trapdoors for all except the innermost layer must be shared between the receiver and the sender.

There is one limitation: at each iteration layer we cut off a number of k bits from a sequence that was regarded as a random string for the previous level. So finally we cannot cut off any bits - the s-bit string becomes too short for the basic scheme to work. However, even then the coercer cannot be sure that the innermost layer of nested encryption has been reached. Simply, Alice and Bob may use multiple transmissions and concatenate these short "random" strings to a long string that again has been constructed in the way described. Let us stress that preparing elements belonging to appropriate translucent sets can be performed off-line, i.e. in advance, without knowing messages to be encrypted.

One can easily see that the original scheme as well as its extension share a serious drawback – many bits are needed to encode a single bit of a message intended to be sent. We partially address this problem in Subsect.3.4.

Most importantly, a single addressee should not use the above scheme on a large scale: each sender knowing the trapdoors to external layers would also know content of these layers in messages encrypted by the others. On the other hand, if the addressee would have several public keys, each one dedicated for another sender, then the protocol would be suspicious-looking.

3.2 Postponed One-Time-Pad

This scheme is shared-key, m_f-ad-hoc, sender-or/and-receiver-deniable ("-and-" if they coordinate their stories).

Let $\mathfrak{R}$ be a large finite ring, say $|\mathfrak{R}| \geq 2^{64}$, in which every nonzero element is invertible (so, $\mathfrak{R}$ is a field) or in which a randomly chosen element is invertible with overwhelming probability (e.g. $\mathfrak{R} = \mathbb{Z}_N$, where N is a RSA number).

Preliminaries: Let $E : \mathfrak{R} \to \mathfrak{R}$ be an encryption scheme. If E is a block cipher with the block-length equal to ℓ bits, then it would be convenient to take $\mathfrak{R}$ being a field $\mathbb{F}_{2^\ell}$. Let a_1, a_2 be two distinct elements of the ring $\mathfrak{R}$. Suppose that the sender and the receiver share:

- a secret generator R of pseudorandom numbers $R : \mathfrak{R} \to \mathfrak{R}$; as we shall see, in case of dishonest opening of a ciphertext a fake generator R_f will be presented to the coercer; R_f will be constructed as a polynomial of some degree d_f, hence to make R_f convincing we must assume that the generator R might be, but do not have to be, a random polynomial of some degree d,
- a secret $b \in \mathfrak{R}$ being the current internal state of R,
- $F(a_1)$ being a value at point a_1 of some straight line $F : \mathfrak{R} \to \mathfrak{R}$, F will be freshly determined during encryption of a new message.

Encryption: The sender calculates $b := R(b)$. If $b = a$, then she makes a new update $b := R(b)$. Then she calculates a straight line determined by two points $(a_1, F(a_1))$, $(b, E(m))$. With overwhelming probability F is a bijection on the ring $\mathfrak{R}$, i.e. $E(m) - F(a_1)$ is invertible in $\mathfrak{R}$. The ciphertext is $F(a_2)$.

Decryption: The receiver updates $b := R(b)$, and if necessary repeats the update. Then on the basis of a fixed pair $(a_1, F(a_1))$ and a pair $(a_2, F(a_2))$ with the second coordinate freshly received, determines $F(b)$. The plaintext is $E^{-1}(F(b))$.

Dishonest Opening: Suppose that the receiver (or the sender) has to reveal in a court of law or to the coercer the plaintexts of the ciphertexts obtained. At this point he originates a set of $d_f + 2$ sensible plaintexts $m_f^{(1)}, \ldots, m_f^{(d_f+2)}$ and for each $i \in \{1, \ldots, d_f + 2\}$ calculates $b_f^{(i)} = F_i^{-1}(E(m_f^{(i)}))$, where F_i are straight lines obtained in the last $d_f + 2$ transmissions. Now he can build a chain of $d_f + 1$ pairs $(b_f^{(1)}, b_f^{(2)})$, $(b_f^{(2)}, b_f^{(3)})$,..., $(b_f^{(d_f+1)}, b_f^{(d_f+2)})$ and treating the second coordinate of each pair as a value at point $b_f^{(i)}$ of some polynomial R_f gets this polynomial. Due to pseudorandomness of the output of E Bob might suppose that all $b_f^{(i)}$ are random, and that with overwhelming probability he will get a polynomial of degree d_f. If the degree is smaller, then as a bonus he adds additional points to the chain, making himself even more credible.

Now, by revealing $b_f^{(1)}$ and the polynomial R_f, he can convince of legality of the last $d_f + 2$ ciphertexts $F_i(a_2)$, as well as of legality of any subset of these ciphertexts. If the court of law has earlier straight lines F_i than the F_1, then Bob might testify that the polynomial R_f was generated recently, the earlier one has been destroyed and obviously he does not remember it.

As we see, having received a summons to appear in, Bob generates a key R_f for his one-time-pad encryption of freshly devised messages $m_f^{(1)}, \ldots, m_f^{(d_f+2)}$. Although the one-time-pad appears also in [2], [3], the above approach provides some explanation of the sources of randomness used, i.e. provides the generator R_f. This explanation, however, is of limited plausibility, which depends of the number of ciphertexts that must be proved to be legal.

Last but not least, using the scheme presented can be explained as a security measure: note that it does not reveal directly any ciphertext transmitted. So even known-ciphertext methods cannot be applied directly to break the encryption.

3.3 Deniable Encryption Based on the ElGamal Cryptosystem

In this subsection we present a fairly practical plan-ahead, shared-key, receiver-deniable encryption scheme that is based on the ElGamal cryptosystem. Using it has some advantages over the classical scheme of Canetti et al. First of all, this cryptosystem is widely known – we avoid using obscure scheme obviously aimed at deniable encryption. Another reason is that the scheme generates much less overhead in terms of size of the ciphertext. Moreover, this receiver-deniable scheme is much simpler than the two-phase one depicted in [3].

The ElGamal Encryption Scheme. This algorithm was introduced in [5], [6]. Let us recall it briefly: Consider the multiplicative group of some finite field, i.e. a group $\mathbb{F}^*_{p^r}$, such that *the Discrete Logarithm Problem* (DLP) in this group is hard. In particular p^r-1 must have some large prime factor. Note that $|\mathbb{F}^*_{p^r}|$ must then itself be a large number, what is particularly useful for our purposes. In this subsection, unless otherwise stated, all arithmetic operations are done in $\mathbb{F}^*_{p^r}$. Let g be a generator of $\mathbb{F}^*_{p^r}$. The private key is a number $x \in \{2, 3, \ldots, \mathrm{ord}g - 1\}$ chosen uniformly at random. The public key is a triple: $\mathbb{F}^*_{p^r}$, g, $y = g^x$. To encrypt a message $M \in \mathbb{F}^*_{p^r}$ the sender chooses a number $k \in \{2, 3, \ldots, \mathrm{ord}g - 1\}$ uniformly at random, and then puts $\alpha := g^k$ and $\beta = M \cdot y^k$. The pair (α, β) is a ciphertext of M. To decrypt (α, β) the receiver uses his private key and calculates $\beta \cdot \alpha^{-x} = (M \cdot y^k) \cdot g^{-kx} = M$.

The Deniable Encryption Scheme

Preliminaries: We assume that there is an established secret s shared by Alice and Bob, and that the Bob's private key $x \in \{2, 3, \ldots, \mathrm{ord}g - 1\}$ is known to Alice (the fact of sharing the private key allows to create a broad-band subliminal channel, in [1, p.2] some similar broad-band channel was created in the ElGamal signature scheme). Rest of the settings is exactly as in the ElGamal protocol.

Encryption: To encrypt a message m_f and an illegal message $m \in \langle g \rangle$ a number $k = HASH(s||m_f)$ is computed. We assume that $HASH(\cdot)$ is a regular hash function with values uniformly distributed in the set $\{0, 1, \ldots, \mathrm{ord}g - 1\}$. Then Alice computes

$$\alpha := g^k \cdot m, \quad \beta := (y^k \cdot m^x) \cdot m_f.$$

Note that the pair (α, β) is a regular ElGamal ciphertext of m_f. Moreover, the greater bit-length of the number $|\mathbb{F}^*_{p^r}|$ is, the longer m can be sent subliminally.

Decryption: First of all, the receiver should retrieve the "legal" message m_f:

$$\beta \cdot \alpha^{-x} = (y^k \cdot m^x) \cdot m_f \cdot (g^k \cdot m)^{-x} = m_f \ .$$

Then he can compute $k := HASH(s||m_f)$, and $m := \alpha \cdot g^{-k}$.

Dishonest Opening: Bob, if coerced, can reveal his key x. The coercer can check that (α, β) is in fact a regular, valid ElGamal encryption of the message m_f. So Bob mimics decryption of the fake message m_f.

It is easy to see that the above scheme provides perfect receiver deniability, i.e. the transcript of the procedure of sending m is indistinguishable from sending m_f. Obviously, the above scheme is not sender-deniable: the sender has no effective procedure that for an argument $\alpha = g^k \cdot m$ returns an exponent k' such that $\alpha = g^{k'}$ (otherwise she could solve DLP in $\mathbb{F}^*_{p^r}$). However, the scheme has some limitations: similarly like in Subsect.3.1, a single addresse should not use the scheme on a large scale – key x is known to the sender.

3.4 A Covert Channel Hidden in Deniable Encryption

In Subsect.3.3, a broadband subliminal channel is depicted. Another, very broad subliminal channel can be embedded into McEliece cryptosystem in straightforward manner (as encoding of an error). Let us remind that McEliece Public Key Cryptosystem was introduced in [8]. Detailed discussion about its security can be found for example in [7] as well as references in this paper. Both broadband channels might be used to transfer deniable ciphertext, constructed e.g. according to the flexibly deniable scheme (hence we would get an analogy to the nested construction from Subsect.3.1). However, due to the volume of such ciphertexts they need to be transferred in parts. And again, attention must be paid to implementation issues: for example some freedom of choice could be left in the ciphertexts for the *Bias Removal Method* (cf. [13], [12]) to ensure uniform distribution of the ciphertexts in the domain of the subliminal channel.

Note that another covert channel might *independently* be hidden in the deniable ciphertexts. Hence, if deniable encryption is embedded in some covert channel, we again get a Russian nesting dolls analogy. Deniable encryption might also be performed stand-alone, like for example the nested construction from Subsect.3.1. Nevertheless, a covert channel embedded in deniable encryption can save much of the capacity not used so far.

In general, *all* the plaintexts carried in the underlying deniable scheme might constitute a smokescreen, and the real, true plaintext can be transferred encrypted in a covert channel embedded in the deniable scheme. Let us again take a look at the construction of a translucent set recalled in Sect.2: For each $x \in \mathcal{S}_t$ we have $x_0 = f^k(a)$ and $b_i = B(f^{k-i}(a))$ for some $a \in \{0,1\}^s$. Hence a might itself be a s-bit ciphertext of some plaintext, and as such looks random. Moreover, a might be a public key ciphertext or a result of an application of a hybrid scheme (cf. [9], [14, Sect. 7]). On top of that, each $x \in \{0,1\}^{s+k}$ intended to define an $\mathcal{R}$-element might additionally transfer each of $2^k - 1$ nonzero messages: let $x \notin \mathcal{S}_t$, define $x_0 = f^k(a)$, and $b'_i = B(f^{k-i}(a))$, then the binary string $(b_k \dots b_1)_2$ is defined as $(b'_k \dots b'_1)_2 \oplus (c_k \dots c_1)_2$ for the nonzero additional message $(c_k \dots c_1)_2$. The message should be a part of some ciphertext – it should

look random. Since the receiver can invert f, he can retrieve bits b'_i on the basis of x_0 and will be able to reconstruct $(c_k \dots c_1)_2$. Obviously, such x truly indicates an $\mathcal{R}$-element, but the sender must claim that x is randomly chosen from $\{0,1\}^{s+k}$. This is necessary to preserve deniability in the underlying scheme.

It is easy to see that if the nested construction from Subsect. 3.1 is applied, then the ciphertext a can be transferred in the s-bit kernel of the most nested doll, but the nonzero "random" strings $(c_k \dots c_1)_2$ might be passed in every layer that indicate an $\mathcal{R}$-element from the layer's set $\{\mathcal{R}, \mathcal{S}\}$.

4 Conclusions

We have proposed a few solutions of deniable encryption problem tailored for various scenarios. However, some problems seems to be remained open. For example, let us note that the most interesting public-key m_f-ad-hoc deniable schemes generate significant overhead.

References

1. Anderson, R.J., Vaudenay, S., Preneel, B., Nyberg, K.: The Newton channel. In: Anderson, R.J. (ed.) Information Hiding. LNCS, vol. 1174, pp. 151–156. Springer, Heidelberg (1996)
2. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption (preliminary version) (May 10, 1996)
3. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
4. Canetti, R., Gennaro, R.: Incoercible multiparty computation (extended abstract). In: FOCS, pp. 504–513. IEEE Comp. Soc, Los Alamitos (1996)
5. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
6. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory 31(4), 469–472 (1985)
7. Kobara, K., Imai, H.: Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 19–35. Springer, Heidelberg (2001)
8. McEliece, R.J.: A public-key system based on algebraic coding theory. In: DSN Progress Report 42-44, pp. 114–116. Jet Propulsion Lab (1978)
9. Möller, B.: A public-key encryption scheme with pseudo-random ciphertexts. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 335–351. Springer, Heidelberg (2004)
10. Naor, M.: Deniable ring authentication. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer, Heidelberg (2002)
11. Näslund, M.: Bit Extraction, Hard-Core Predicates, and the Bit Security of RSA. Doctoral Thesis, Royal Institute of Technology, Department of Numerical Analysis and Computing Science, Stockholm (August 1998)

12. Young, A., Yung, M.: Kleptography: Using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997)
13. Young, A., Yung, M.: Malicious cryptography: Kleptographic aspects. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 7–18. Springer, Heidelberg (2005)
14. Young, A., Yung, M.: A space efficient backdoor in RSA and its applications. In: Preneel, B., Tavares, S.E. (eds.) SAC 2005. LNCS, vol. 3897, pp. 128–143. Springer, Heidelberg (2006)

Taming of Pict

Matej Košík

Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
kosik@fiit.stuba.sk

Abstract. This article presents additional necessary measures that enable us to use Pict as an object-capability programing language. It is desirable to be able to assess the worst possible threat that we—users—risk if we run a given program. If we know the threat, we are able to decide whether or not we are willing to risk running the program. The cost of a security audit that reveals such an assessment will be non-zero but it need not to be directly dependent on the size of the whole original program. It is possible to write programs in such a way that this analysis can be reliably performed on a fraction of the original program—on the trusted computing base. This technique does not always give the most accurate assessment but it gives sound and interesting assessment relatively cheaply. It does not prevent usage of other techniques that can further refine the initial assessment.

1 Introduction

There are two different points of view of a computer system. We can view it from an administrator's point of view and from a user's point of view. Users should be regarded as primary because the purpose of computers is not to be administered but to be used. The goal of the administrator is to ensure that none of the users is given excess authority. The goal of the user is (should be) to ensure that each of the processes runs with appropriate authority. Security mechanisms provided by operating system are practical for administrator but they do not help users with their security goals. Microsoft "Immutable" Law #1 states:

> If a bad guy can persuade you to run his program on your computer, it's not your computer anymore.

The problem is, how to decide who is a good guy and who is a bad guy. More importantly, even good guys can make mistakes and their programs can cause damage. The purpose of the computer is that we—users—can run programs on it. This rule basically says that we are safe as long as we do not run any program on it. Let us stop here and think how ridiculous it is.

Noticeable progress has been made in the area of designing programming languages with respect to security. Outstanding example is the E programming language [1]. From the security point of view[1], it is interesting because it enables

[1] The E programming language addresses also other important problems.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 610–621, 2008.

programmers to follow the principle of the least authority (POLA). Multiple aspects of the language contribute to this fact:

- the authority to invoke methods of a particular object is an unforgeable capability
- when some subsystem decides to keep some capabilities as private, there are no language constructs that would enable other untrusted subsystems to "steal" them
- the reference graph can evolve only according to *rules of allowed reference graph dynamics* presented in Section 9.2 of [1]

The contribution of this paper is that it shows how, through a refactorization of the libraries of the Pict programming language [2], the "ambient authority" is reduced to a minimum, and Pict can provide many of the benefits of existing object-capability languages. Provided examples illustrate the technique for determining authority of untrusted subsystems without the need to analyze their code.

2 Related Work

While this article is mostly concerned with taming of Pict—turning Pict into an object-capability programming language—this is not the first work of this kind. See for example: Oz-E [3], Emily [4], Joe-E [5].

The Raw Metal occam Experiment (RMoX) [6] can be regarded as a source of inspiration that languages, based on process calculi, can be used for defining of behavior of various operating system's components and their mutual interaction. Using programming language constructs as a mechanism for isolation of various subsystems from each other instead of relying on awkward hardware support is also one of the points of the Singularity project [7].

3 The Pict Programming Language

The goal of the authors of the Pict programming language was to create a language that could play for the π-calculus a similar role as Haskell plays for the λ-calculus. It is defined in layers, see Figure 1. Syntax of the core language is formally described in the Pict Language Definition [8] in Chapter 3; see rules tagged as C (as *Core*). Some of the syntactically correct programs can be further rejected by the typing rules at compile time. Semantics of the core language is defined in Chapter 13 of that document. It defines:

- structural congruence relation
- reduction relation

These together define behavior of all Pict programs.

Programs written in the core Pict cannot break *rules of allowed reference graph dynamics*. Derived forms make functional and sequential programming in Pict more convenient. By definition, they do not add expressivity to the core

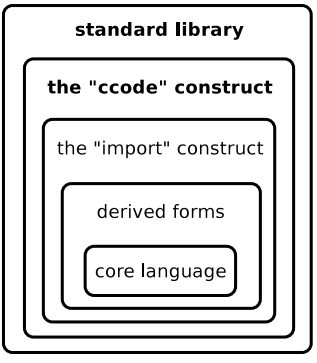


Fig. 1. Layers of the Pict programming language. Programs that are composed solely from core constructs, derived forms and `import` directives are completely harmless because they have minimal authority.

Pict language and thus can be used without concerns that the *rules of allowed reference graph dynamics* could be broken.

The `import` directive is one of the two extralinguistic constructs of Pict. It enables us to split the whole program into multiple, separately compilable modules. These modules are related via `import` construct. This relation forms partial order with the biggest element—it is the main module of a complete program. There is no `export` directive via which the programmer could explicitly specify which bindings he wants to export from a given module. All the variables that are bound in the outer-most scope are automatically exported. The effect of the `import` directive is that all the variables exported by the imported module are visible in the importing module.

The `ccode` construct is the second of the two extralinguistic constructs of Pict. It enables the programmer to inline arbitrary C code into Pict programs. This is very useful and very dangerous at the same time. It is the sole mechanism that Pict programs can use to interact with their (non-Pict) environment such as the operating system. It is also used for implementation of certain operations in an efficient way. This is not absolutely essential[2] but it is pragmatic. Additional rules presented later in the text ensure that this construct cannot be directly or indirectly abused by untrusted modules to gain excess authority. These additional rules ensure that untrusted modules cannot break the *rules of allowed reference graph dynamics*.

The Standard Pict Library [9] provides several reusable components. Figure 2 shows some modules that are part of this library. Some aspects of this original organization are logical and some are not logical. The `import` directive binds them into a partially ordered set. Minimal elements (`Misc`, `Prim`) are shown on the left. Maximal elements (`Random`, `Ref`, `Signals`, `Array2`, `Queue`, `Args`, `IO`) are shown on the right. The $A \prec B$ means that module A is imported by module B.

[2] The core language can model integers, booleans, strings and other values of basic data types together with operations with them.

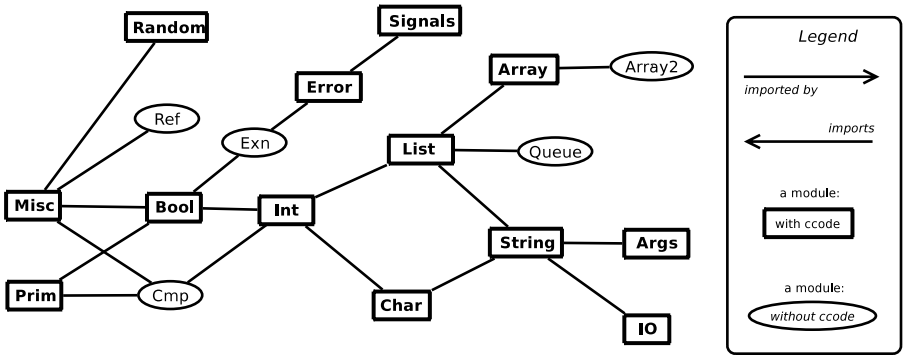


Fig. 2. Partial ordering of modules with respect to the `import` relation. These modules are described in detail elsewhere [9]. This figure is provided only for general impression concerning the structure of modules. Not all the aspects of this original version are completely logical.

Those names that are bound in the outer-most scope of some module are also exported by that module. Let $en(A)$ denote a function that maps a given module A to the set of names that it exports. Then, by definition of the semantics of the `import` directive:

$$A \prec B \;\Rightarrow\; en(A) \subseteq en(B)$$

That is, all the names bound in the outer-most scope of module A are also bound in the outer-most scope of module B that imports A.

When $bn(A)$ denotes the set of all names bound in any scope within module A then for all modules A, by definition of the `import` construct, holds:

$$en(A) \subseteq bn(A)$$

All the names bound in A need not to be, and usually indeed are not, exported.

4 Refactorization of the Original Pict Library

The attempt to minimize the trusted computing base is inherently a good idea. In this light, the organization of the original Pict library is not optimal. Each module that employs the `ccode` construct must be considered as part of the trusted computing base. And, as you can see in Figure 2, there are many such modules. Additionally, the original set of primitives, expressed via `ccode` construct, is not orthogonal. Many of the existing primitives can be rewritten in terms of a pure Pict code. After we removed those superfluous primitives and we concentrated the originally scattered primitives in a few dedicated modules, the situation is different, see Figure 3. Now it has sense to discriminate among trusted and untrusted modules as follows:

- *trusted* modules can contain any (compilable) code
- *untrusted* modules:
  - cannot use the `ccode` construct
  - cannot import any trusted module except for the `Prim` module which provides harmless primitives

Due to the inherent properties of the Pict programming language, these measures are sufficient to ensure that rules of allowed reference graph dynamics hold for all our untrusted modules.

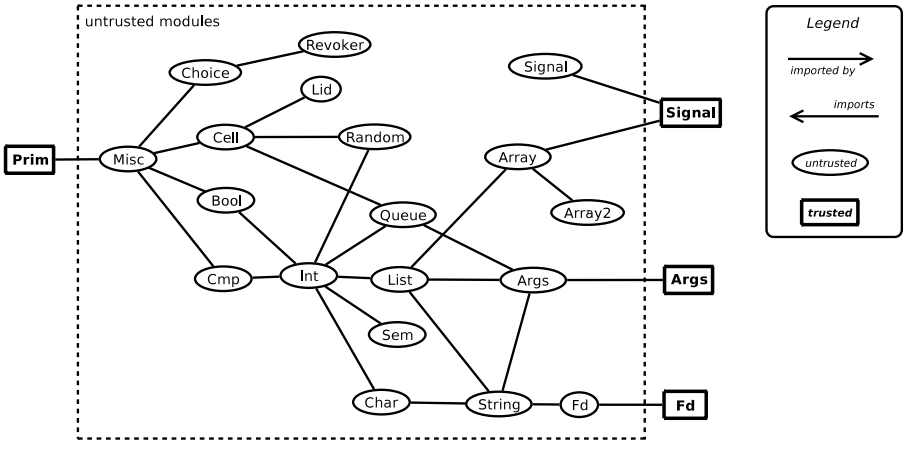


Fig. 3. Refactored standard library of Pict with respect to security

After all these measures, by *minimal authority* of untrusted modules written in Pict we mean:

- Trusted modules have no straightforward way to influence policy how much memory can various sub-components allocate from the common heap of free memory which in Pict is bounded—it's size is by default 1 MB. When untrusted components exhaust it, the Pict runtime prints out a relevant error message and terminates the whole system.
- Trusted modules have no straightforward way how influence the scheduling policy that would define rules for CPU utilization by untrusted modules. At present, there is a scheduler. It ensures fairness of the CPU utilization but this is not what we always want.
- Untrusted modules can make run-time errors (such as division by zero or they might attempt to access non-existent element of some array). These run-time errors are always detected and result in calling the `error` function that performs appropriate actions. At present, this means that some error message will be printed on the screen and the whole system will terminate. So indirectly, untrusted modules have the authority to terminate the whole system.

This situation is not at all fully satisfactory. But this is still far better than ambient authority of trusted modules. To address these remaining issues the whole Pict runtime must be redesigned with these problems in mind.

5 Powerbox

Powerbox is a fundamental security pattern. Its origin can be traced to the DarpaBrowser described in [10] and in [11]. It enables us to dynamically raise the level of authority of untrusted subsystems to a sufficient and acceptable level.

If we are concerned with some single purpose program then we have to identify the authority this program needs. There is nothing inherently wrong that various programs require some authority. As long as it is explicitly declared, users or security auditors can efficiently judge whether it is acceptable for us to grant such authority to the actual program.

To be able to follow POLA, the whole program must be split in at least two modules. The first of them will be trusted and the other one will be untrusted. The purpose of the trusted module is to communicate part of its ambient authority to the untrusted module. The purpose of the untrusted module is to use the authority it is given and to do what we expect from it. It receives required capabilities "by introduction". What kind of capabilities are communicated and through which channel depends on the contract between the trusted and the untrusted part.

Let us show a very simple example. We keep the untrusted module in the `Untrusted/Guest.pi` file and the trusted module in the `Trusted/Host.pi` file. The `Untrusted/Guest` module might look as follows:

```
new contract : ^!String

run contract?logger = ( logger!"0123456789"
                      | logger!"0123456789"
                      | logger!"0123456789"
                      | logger!"0123456789"
                      | logger!"0123456789"
                      )
```

It creates a fresh channel `contract` that can be used for passing values of the `!String` type[3]. The process in the untrusted module blocks until it receives a

[3] Pict is a strongly typed programming language. Each channel has a type. This type determines what kind of values can be communicated over a given channel. An attempt to send a wrong type of value over some channel is detected at compile time. The `contract` capability has a type `^!String`. Our process holds this capability *by initial conditions*. The initial `^` character means that this capability can be used for sending as well as for receiving values of the `!String` type. Our process uses this capability to receive a value from it and binds this value to the `logger` capability. This capability is of `!String` type. That means that the `logger` capability can be used for sending strings along it. It cannot be used for receiving strings from this channel.

value from the `contract` channel. When such value arrives, it will be bound to the `logger` variable. The untrusted guest then has all the authority it needs to do its job. In this case it prints 50 characters on the screen. The above program is not very useful but it could very well perform various simulations and then print out the simulation report. The above code fragment is a mere illustration.

The `Trusted/Host` module is responsible for selecting parts of its ambient authority and communicating appropriate capabilities to the untrusted module. For example:

```
import "Untrusted/Guest"
import "Trusted/Fd"

run contract!print
```

It imports two modules. The first one is `Untrusted/Guest`. This means that it will see the `contract` capability bound in that module. It also imports the `Trusted/Fd` module. It means that it will see the `print` capability bound in that module. It is up to this trusted module to select proper capabilities. In this case it selects the `print` capability and sends it over the `contract` channel. Of course, there may be situations where some guest needs more than one capability. In those cases the trusted host sends an n-tuple of capabilities.

Appropriate makefile for building executable out of these two modules can look as follows:

```
Host: Trusted/Host.pi Untrusted/Guest.px
    pict -o $@ $<

Untrusted/Guest.px: Untrusted/Guest.pi
    isUntrusted $< && pict -reset lib -set sep -o $@ $<
```

Please notice two things:

- `Untrusted/Guest.pi` module is checked with the `isUntrusted` script whether it indeed can be regarded as untrusted[4]
- we compile the `Untrusted/Guest.pi` module with the `-reset lib` flag that inhibits inclusion of the standard prelude[5].

These two actions give us enough confidence to believe that the `Untrusted/Guest` module has initially *minimal authority*. Its authority is later raised to be able to print characters on the standard output. It is not given any other authority. It cannot tamper with files that can be accessed by the user that runs this program. The untrusted module cannot communicate with other processes on your local system. Neither it can communicate over network. It can only print as many characters on

[4] Untrusted modules cannot employ the `ccode` constructs. Untrusted modules cannot import trusted modules except for the `Trusted/Prim` module.

[5] Precise information concerning the "standard prelude" can be found in [12]. Basically, it is a default sequence of `import` directives that is desirable in case of trusted modules but it is undesirable in case of untrusted modules.

the standard output as it wishes. For some programs this kind of authority might be completely sufficient and as you can see it can be trivially implemented.

The same scheme has many variants. The derived forms make certain useful things such as functional programming as well as sequential programming easier. If we express our trusted host and our untrusted guest in so called "continuation passing style" then it would appear that the trusted host gives the untrusted guest the capability to call certain functions. In this case, it is completely up to the trusted host to choose the right set of function-capabilities. The chosen set determines the authority of the untrusted guest.

The Powerbox pattern can also be used in situations when our system consists of multiple untrusted subsystems. In that case, each subsystem will be placed in a separate powerbox. This way, each untrusted component can be given different capabilities and thus we can determine the authority of particular untrusted modules independently. The complexity of the trusted part is determined by the complexity of our security policy. It is independent from the complexity of the untrusted part that does the real job.

6 Experiments in the Kernel Space

Capability-secure languages are useful not only in user-space but they can have interesting applications in the kernel space, too. They can be a precursor to making progress in monolithic (in the traditional sense) kernels. We have a flexible alternative to the classical microkernel-based operating system architecture. In our preliminary experiment we use the Pict programming language because it was easier to adapt to run on a bare metal. From the security point of view Pict is in principle as good as E. From the concurrency point of view, the E programming language is much better. It provides more advanced synchronization mechanisms than Pict so E is a very good alternative for the future. Figure 4 shows the structure of modules with respect to the `import` relationship. This relationship determines the *connectivity by initial conditions* according to the semantics of the `import` directive. Capabilities that are exported from module A are also visible in module B if module A is imported by module B. Modules `Memory`, `IRQ` and `IO` provide various powerful primitives. For example the `Trusted/Memory` module exports a function

```
(memory.write.byte offset value)
```

that enables (those who see this function-capability) to write any `value` of byte size to any `offset` within current data segment (that spans through the whole physical memory). The `Trusted/IO` module exports two functions:

```
(io.write.byte port value)
(io.read.byte port)
```

Those who see the first function can write any `value` (byte) to any I/O `port`. Those who see the second function can read any I/O `port` of byte size.

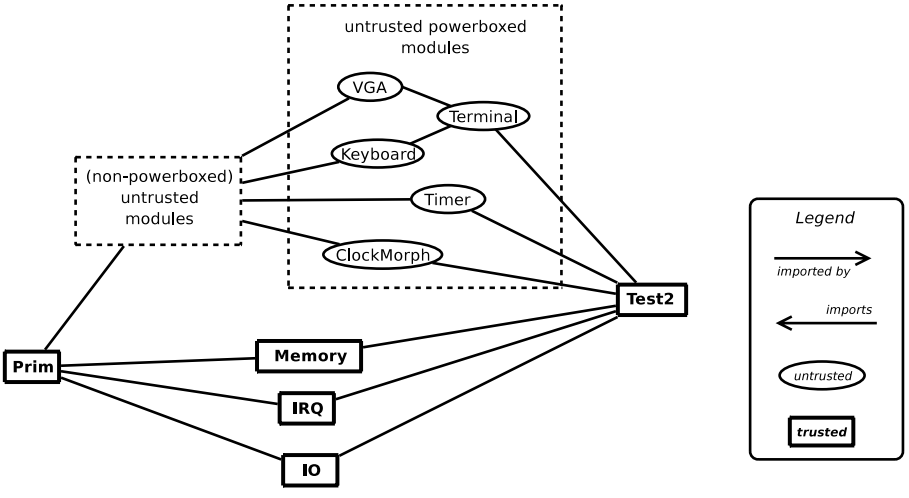


Fig. 4. Structure (with respect to the `import` relation) of modules that form our experimental kernels. `Test2` is the main module.

The trusted `Test2` module has a single task, to disseminate proper capabilities to proper modules via the Powerbox pattern according to POLA. For example, one thing that the `VGA` module needs is the authority to write to the I/O port number 980 (`0x3D4` in hexadecimal system). The `Test2` module sees the `io.write.byte` procedure so it could give the `VGA` module this capability. However, with respect to POLA, it gives it a different capability:

```
\(value) = (io.write.byte 980 value)
```

This abstraction (unnamed function, lambda-expression) is given to the `VGA` module. It gives it the authority to write any `byte` to the I/O port 980. The `Test2` module gives the `VGA` module few other similar capabilities. As a result, the `VGA` module has the authority:

- to write any byte to the I/O port 980
- to write any byte to the I/O port 981
- to read a byte from the I/O port 981
- to write any byte to the Video RAM (nowhere else)

This is enough for the `VGA` module to be able to provide expected services. Various abstractions created by the `Test2` module that act as proxies to more powerful capabilities are denoted as small numbered rectangles in the `Trusted/Test2` module in Figure 5.

If we have a complete and correct[6] knowledge concerning behavior of functions, procedures and processes in all the trusted modules (`Memory`, `IO`, `IRQ`, `Test2`), then we can safely assess the upper bound of authority of all untrusted

[6] This is why we should try to keep the trusted computing base as small as possible.

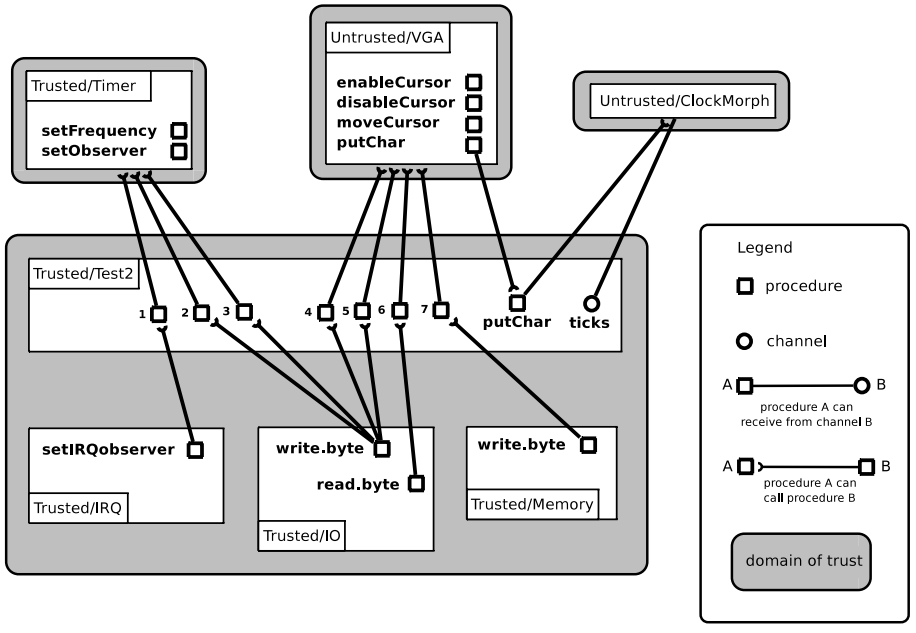


Fig. 5. The actual reference graph in the `Test2` kernel

powerboxed modules (`Timer`, `VGA`, `ClockMorph`) safely only with regard to the reference graph shown in Figure 5. It is possible because rules of allowed reference graph dynamics hold for our untrusted modules. In case of `VGA` and `Timer` modules the situation is trivial. We have complete knowledge about behavior of functions that we give to these two modules. Recall that we give the `VGA` module the following capability.

```
\(value) = (io.write.byte 980 value)
```

Since we have a complete information concerning the `io.write.byte` function, we also have a complete information concerning the behavior of the above abstraction. So the authority of the `VGA` and the `Timer` can be determined precisely; regardless of their actual implementation. These drivers provide various functions. One of them is:

```
(vga.putChar x y ch attribute)
```

When called, it puts any given character `ch` with any `attribute` anywhere on the screen. This is its assumed effect we believe it does.

In a similar way can we also give appropriate authority to the `ClockMorph` component. It is supposed to show the number of seconds from the boot-time in `HH:MM:SS` format. This kind of component obviously needs the authority to print eight consecutive characters somewhere on the screen. If we want to follow POLA also in this case, we have to implement a proxy function that will drop

most of the `vga.putChar` authority and it will provide the ability to change eight consecutive characters on the screen, not more. We have defined the `putChar` function in the `Test2` module that does exactly this. It relies on the `vga.putChar` function, see Figure 5. Regardless how perfectly our trusted proxy function implements additional restrictions, unless we verify the correct behavior of the original untrusted `vga.putChar` function, we cannot claim anything stronger than: "The `ClockMorph` component has as much authority as the `VGA` driver plus it can receive messages from the `tick` channel." But even with this simple technique, without studying the code of particular untrusted modules, we can see that the authority of the `ClockMorph` component is fairly limited.

7 Conclusion and Future Work

Our immediate goal is to address two immediate problems concerning minimal authority:

- there is no way how to give particular untrusted components only limited share of the CPU bandwidth
- there is no way how to give particular untrusted components only limited amount of memory

At present, when some of the untrusted components uses up the whole available memory, the runtime terminates the system. This is a show-stopper for using Pict for writing robust, from the traditional point of view, monolithic operating system kernels.

Sophisticated proof-techniques were developed for proving correctness of functional code, sequential (procedural) code. These can be very useful in analysis of procedures that are made visible to untrusted powerboxed modules. From the formally proved effects of these procedures (or processes) we can determine the authority of untrusted powerboxed modules that are part of the system.

Acknowledgments. This work was partially supported by the Slovak Research and Development Agency under the contract No. APVV-0391-06 and by the Scientific Grant Agency of Slovak Republic, grant No. VG1/3102/06.

References

1. Miller, M.S.: Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control. PhD thesis, Johns Hopkins University, Baltimore, Maryland, USA (2006)
2. Pierce, B.C., Turner, D.N.: Pict: A programming language based on the pi-calculus. In: Plotkin, G., Stirling, C., Tofte, M. (eds.) Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, Cambridge (2000)
3. Spiessens, F., Roy, P.V.: A Practical Formal Model for Safety Analysis in Capability-Based Systems, Revised Selected Papers. In: De Nicola, R., Sangiorgi, D. (eds.) TGC 2005. LNCS, vol. 3705, pp. 248–278. Springer, Heidelberg (2005)

4. Stiegler, M., Miller, M.: How Emily Tamed the Caml. Technical Report HPL-2006-116, Advanced Architecture Program. HP Laboratories Palo Alto (2006)
5. Mettler, A.M., Wagner, D.: The Joe-E Language Specification (draft). Technical Report UCB/EECS-2006-26, EECS Department, University of California, Berkeley (2006)
6. Barnes, F., Jacobsen, C., Vinter, B.: RMoX: A raw-metal occam experiment. In: Broenink, J., Hilderink, G. (eds.) Communicating Process Architectures 2003. Concurrent Systems Engineering Series, vol. 61, pp. 269–288. IOS Press, Amsterdam, The Netherlands (2003)
7. Aiken, M., Fähndrich, M., Hawblitzel, C., Hunt, G., Larus, J.: Deconstructing process isolation. In: MSPC 2006. Proceedings of the 2006 workshop on Memory system performance and correctness, pp. 1–10. ACM, New York (2006)
8. Pierce, B.C., Turner, D.N.: Pict language definition (1997)
9. Pierce, B.C., Turner, D.N.: Pict libraries manual. Available electronically (1997)
10. Wagner, D., Tribble, E.D.: A Security Analysis of the Combex DarpaBrowser Architecture (2002)
11. Stiegler, M., Miller, M.S.: A Capability Based Client: The DarpaBrowser. Technical Report Focused Research Topic 5 / BAA-00-06-SNK, Combex, Inc. (2002)
12. Pierce, B.C., Turner, D.N.: Pict libraries manual. Available electronically (1997)

Classification, Formalization and Verification of Security Functional Requirements

Shoichi Morimoto[1], Shinjiro Shigematsu[2], Yuichi Goto[2], and Jingde Cheng[2]

[1] School of Industrial Technology, Advanced Institute of Industrial Technology
1-10-40, Higashi-oi, Shinagawa-ku, Tokyo, 140-0011, Japan
morimoto-syoichi@aiit.ac.jp
[2] Department of Information and Computer Sciences, Saitama University
Saitama, 338-8570, Japan
{shigematsu, gotoh, cheng}@aise.ics.saitama-u.ac.jp

Abstract. This paper proposes a new hybrid method to formally verify whether the security specification of a target information system satisfies security functional requirements defined in ISO/IEC 15408 evaluation criteria for security. We classify at first the security functional requirements of ISO/IEC 15408 into two classes: static requirements concerning static properties and dynamic requirements concerning dynamic behavior of target systems, and then formalize the static requirements with Z notation and the dynamic requirements with temporal logic. Thus, we can verify static properties using theorem-proving and dynamic behavior using model-checking. As a result, developers can easily use the method to verify whether the security specification of a target information system satisfies both static and dynamic security functional requirements defined in ISO/IEC 15408. The new method is an evolution and improvement of our early verification method where only Z notation was adapted and to verify dynamic behavior of target systems is difficult.

1 Introduction

We have already succeeded in formalization of the security functional requirements, defined in ISO/IEC 15408 common criteria (CC) [8], and formal verification of specifications with them [11]. In the research, we used Z notation, a formal method that produces actual results in the verification of software reliability with theorem-proving [9].

We formalized beforehand all 251 security functional requirements of the CC as formal criteria templates which are necessary to any verification of security specifications. However, it is difficult to formalize some criteria of the CC only in Z, because the CC have many types of security criteria to comply with various security requirements. Some criteria require a system to have various security functions or data for security, while the others define conditions which a system must keep after some specified operations. In the paper [11], we formalized in a sort of way such criteria in Z. This brute force approach makes the formalization

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 622–633, 2008.

and the verification of target systems complicated and difficult. In particular, we failed to discriminate between static properties and dynamic behavior for the formalization and the verification.

Therefore, we classify the security functional requirements into static and dynamic requirements. Moreover, we redefine a new hybrid formal verification method to verify information systems including both static and dynamic security functional requirements defined in the CC. Consequently, we can easily and precisely verify whether a specification of an information system respectively satisfies static and dynamic requirements of the security functional requirements. The classification not only improves the method of [11], but also makes developers easy to implement properly the security functional requirements.

2 Classification and Formalization of Security Functional Requirements

Here we describe the improvements of the verification method.

2.1 Classification of Security Functional Requirements

The CC establish a set of functional components, providing a standard way of expressing the functional requirements for target systems. The CC describe the requirements for functions that implement the security policies of the information system. In other words, the CC offer the functions information systems should use to ensure security, i.e., security functional requirements.

For example, the original text of the security functional requirement, named FIA_SOS.1.1, is as follows [8].

> **FIA_SOS.1** Verification of secrets
> **FIA_SOS.1.1** The TSF shall provide a mechanism to verify that secrets meet [assignment: *a defined quality metric*].

In the original text, TSF refers to a system security function, and the *assignment* is the specification of an identified parameter in the system. FIA_SOS.1.1 specifies that a system must have *a defined quality metric* and *a mechanism to verify that secrets meet it*. Thus, the majority of the requirements inquire whether a system has security functions or data.

On the other hand, some requirements define an action or state which a system should validate with a specific change. For instance, there is the following dynamic requirement [8].

> **FPT_RCV.1** Manual recovery
> **FPT_RCV.1.1** After a failure or service discontinuity, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.

FPT_RCV.1.1 requires that a system must have *a maintenance mode* and shift to the mode after *a failure or service discontinuity*.

Judging from the above, security functional requirements in different aspects are scattered in the CC. Therefore, we classified the requirements into static and dynamic requirements on the basis of the above difference and temporal keywords, e.g., 'when,' 'before,' and 'after.' As a result, it turned out that 77 security functional requirements out of 251 are dynamic requirements. The dynamic requirements are listed in Appendix A.

2.2 Formalization of Security Functional Requirements

In our early research [11], all the 251 requirements of the CC were formalized into inspection formulas in Z notation. The formalized requirements are published in the web site [1]. However, because Z notation is unfit to describe temporal concepts, it is difficult to formalize all the security functional requirements only in Z. The formalization of the dynamic security functional requirements in our early research is somewhat unnatural translation.

Thus, we reformalized the 77 dynamic security functional requirements in temporal logic, after we had extracted their substantive meaning from the context of many public specifications [5] that have been certified by ISO/IEC 15408.

Here we show the formalization of the dynamic security functional requirements of the CC. For example, we formalized FPT_RVM.1.1 in the paper [11] as follows.

TEMPLATE FPT_RVM.1.1

$\forall$ ***System***$''$ | **TSP_enforcement_functions** $\fatsemi$ **TSC_functions**

- present_state $\in S_a \wedge$ present_state$' \notin S'_a \wedge$ present_state$'' \notin S''_a$
  $\wedge$ present_state $\notin S_b \wedge$ present_state$' \in S'_b \wedge$ present_state$'' \notin S''_b$
  $\wedge$ present_state $\notin S_c \wedge$ present_state$' \notin S'_c \wedge$ present_state$'' \in S''_c$

The original FPT_RVM.1.1 specifies that a system must not execute any functions until it executes *TSP enforcement functions*. In the paper [11], we defined S_a as the state prior to *TSP enforcement functions*, and S_b as the state following *TSP enforcement functions*. *TSC functions* are permitted only after performing *TSP enforcement functions*, i.e., when the system's state is S_b. This template formula means that after performing *TSP enforcement functions* and *TSC functions* in the given order, the state of the *System* changes from S_a to S_b and S_b to S_c. That is, we assumed that the state of the system changes with the execution of *TSP enforcement functions* from state S_a to state S_b, and with the execution of *TSC functions* from state S_b to state S_c. We asserted that the transition from S_a to S_b is equivalent with *TSP enforcement functions* being performed. However, the formula is verbose and merely denotes that a state changes in the order.

Therefore, we rewrote all the 77 dynamic security functional requirements in temporal logic. The formalized FPT_RVM.1.1 is reformalized as follows.

$\Box$ ($\neg$ TSC_functions W TSP_enforcement_functions)

The formula □ **p** means that **p** is always true and the formula **p** W **q** means that **q** must be true as long as **p** is false [4]. That is, the formula means that TSP enforcement functions are invariably performed before execution of TSC functions (any function in a system). Thus, the template FPT_RVM.1.1 becomes simple and meaningful.

3 The Hybrid Verification Method

With the reformalization, we also modified the verification procedure of our early method. To verify formally the dynamic requirements, we introduce model-checking into our early verification method. Model-checking can verify that a specification satisfies dynamic behavior formalized in temporal logic [4].

3.1 Outline of the Improved Verification Method

The following is the new verification procedure.

1) Select the formalized criteria (i.e., templates) required in a target system.
2) Formalize the system's static specification in Z notation and its dynamic specification in model-checking tools.
3) Instantiate the selected criteria templates.
4) Verify the formalized specifications against the instantiated criteria.

To verify a security specification, a verifier first selects the required criteria, which we have formalized as templates. Secondly, the verifier formalizes a static specification besides dynamic aspects of a target system in Z and the dynamic aspects in a model-checking tool. After the formalization, each criterion is correctly verifiable.

In the third step, the verifier must instantiate the selected criteria templates to fit them into the target specification. Then he verifies whether the instantiated criteria are deducible from the given axioms and the formalized specification

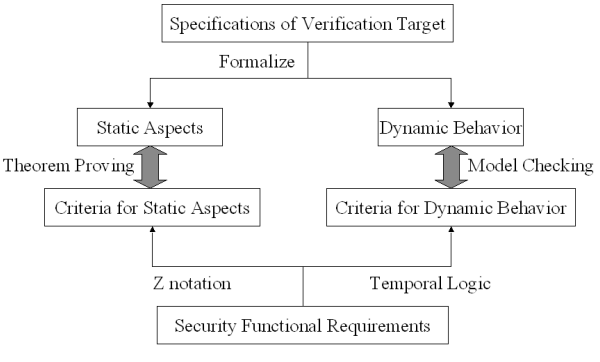


Fig. 1. The new procedure of the verification method

as premises with theorem-proving and model-checking, e.g., the theorem-prover Z/EVES [12] and the model-checkers [2] (cf., Fig. 1). We will demonstrate details of these steps in Section 4, giving a concrete example.

3.2 Separation of Static and Dynamic Specifications

In the second step of the method, verifier must formalize static specifications of a verification target in Z and its dynamic specifications in model-checking tools respectively. If the target system includes the following specification, it is necessary to separate it. The separated dynamic specification can be easily formalized in model-checking tools.

The state machine diagram in Fig. 2 is an excerpt from Chapter 6 of the reference [9].

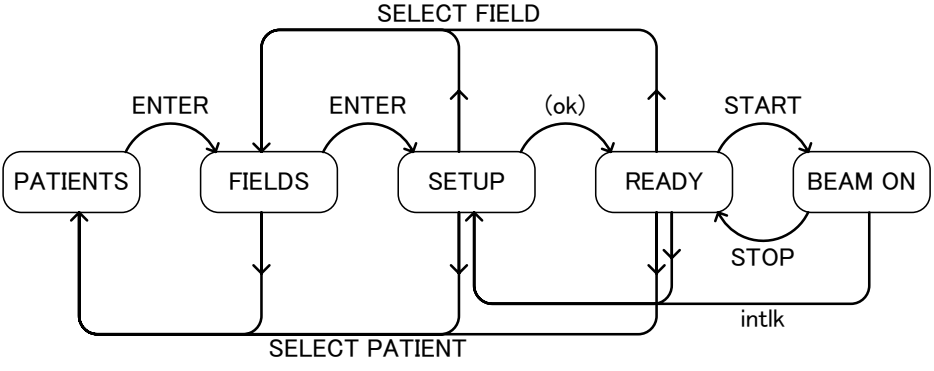


Fig. 2. Therapy control cascade: state transition

We can generally formalize the diagram in Z as follows.

$$STATE ::= patients \mid fields \mid setup \mid ready \mid beam\_on$$
$$EVENT ::= select\_patient \mid select\_field \mid enter \mid start \mid stop \mid ok \mid intlk$$
$$FSM == (STATE \times EVENT) \nrightarrow STATE$$

$$no\_change, transitions, control : FSM$$

$$control = no\_change \oplus transitions$$
$$no\_change = \{s : STATE;\ e : EVENT \bullet (s, e) \mapsto s\}$$
$$transitions = \{(patients, enter) \mapsto fields,$$
$$(fields, select\_patient) \mapsto patients, (fields, enter) \mapsto setup,$$
$$(setup, select\_patient) \mapsto patients,$$
$$(setup, select\_field) \mapsto fields, (setup, ok) \mapsto ready,$$
$$(ready, select\_patient) \mapsto patients,$$
$$(ready, select\_field) \mapsto fields,$$
$$(ready, start) \mapsto beam\_on, (ready, intlk) \mapsto setup,$$
$$(beam\_on, stop) \mapsto ready, (beam\_on, intlk) \mapsto setup\}$$

Although the state transition is not so complicated, the Z description is very verbose. In addition, since general information systems include many specifications

besides state transition, the description will become more complicated. Therefore, our method makes verifiers separate state transition from Z descriptions.

4 Application

To show utility of the improved verification procedure, we present an example of the verification according to the above steps. Using Z notation and model-checking together, we herein briefly verify an example specification with UML diagrams. In this paper we use the theorem prover Z/EVES and model checker NuSMV [3].

The diagrams in Fig. 3 show a part of the certified specifications in a database from [5], which meets the criteria FIA_UID.1.2, FIA_USB.1.1, and FIA_UAU.5.2.

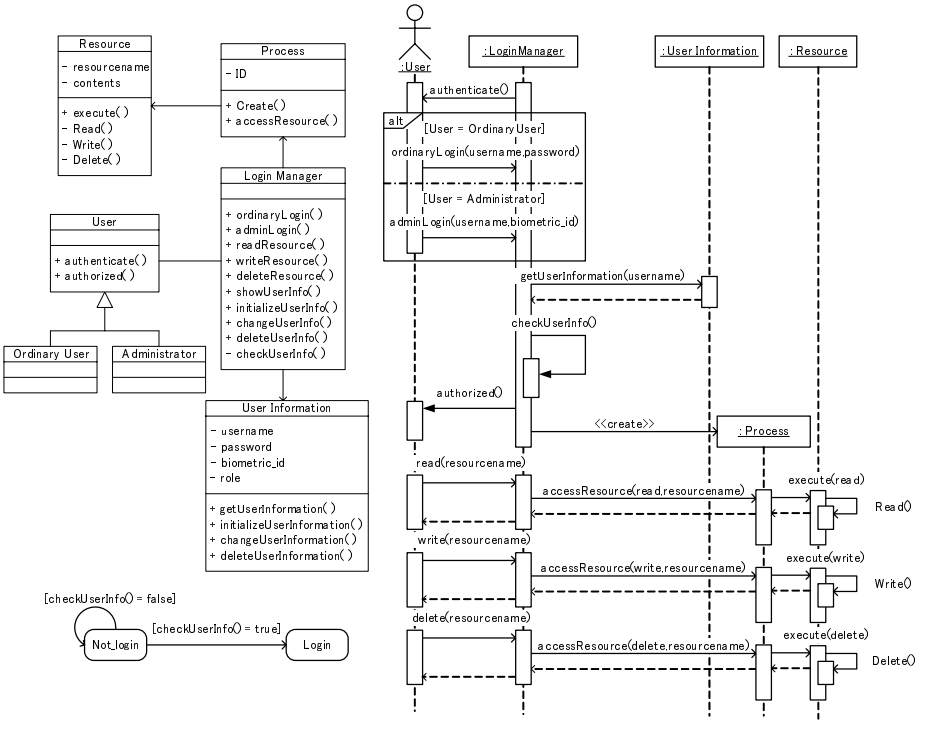


Fig. 3. The example specification in UML

System users are classified according to their role, ordinary users or administrators. An ordinary user is authenticated by a password and an administrator is authenticated by a biometric id. The system allows the authenticated user to change his id. Administrators are permitted to create, change, and delete resources in the database. Ordinary users are only permitted to refer the resources. The database does not permit any operations until a user logs in.

4.1 Example Verification with Theorem-Proving

In order to verify a class diagram with theorem-proving, RoZ was proposed [6]. RoZ can semi-automatically convert a class diagram to skeleton codes in Z. Each class or its operation in a class diagram is formalized as a schema in Z by RoZ. The following is the description which we added the details to the skeleton codes of Fig. 3.

$[Char, PartsInformation, Contents]$
$User ::= Ordinary\_user \mid Administrator$
$Role ::= ordinary\_user \mid administrator$
$Operation ::= read \mid write \mid delete$
$Bool ::= TRUE \mid FALSE$
$String == \mathrm{seq}_1\ Char$
$username == String;\ Password == String$

UserInformation
$password : username \pfun Password$
$biometric\_id : username \pfun PartsInformation$
$role : username \pfun Role$

initializeUserInformation
$\Delta UserInformation$
$Password\_Policy$
$u? : username$
$input\_role? : Role$
$input\_biometric\_id? : PartsInformation$

$password' = \{(u? \mapsto input\_password?)\}$
$biometric\_id' = \{(u? \mapsto input\_biometric\_id?)\}$
$role' = \{(u? \mapsto input\_role?)\}$

getUserInformation
$\Xi UserInformation$
$u? : username$
$output\_password! : Password$
$output\_role! : Role$
$output\_biometric\_id! : PartsInformation$

$u? \in \mathrm{dom}\ password$
$u? \in \mathrm{dom}\ biometric\_id$
$u? \in \mathrm{dom}\ role$
$output\_password! = passwordu?$
$output\_biometric\_id! = biometric\_idu?$
$output\_role! = roleu?$

changeUserInformation
$\Delta UserInformation$
$u? : username$
$new\_password? : Password$
$new\_role? : Role$
$new\_biometric\_id? : PartsInformation$

$u? \in \mathrm{dom}\ password$
$u? \in \mathrm{dom}\ biometric\_id$
$u? \in \mathrm{dom}\ role$
$password' = password \oplus \{(u? \mapsto new\_password?)\}$
$biometric\_id' = biometric\_id \oplus \{(u? \mapsto new\_biometric\_id?)\}$
$role' = role \oplus \{(u? \mapsto new\_role?)\}$

deleteUserInformation
$\Delta UserInformation$
$u? : username$

$u? \in \mathrm{dom}\ password;\ u? \in \mathrm{dom}\ role$
$u? \in \mathrm{dom}\ biometric\_id$
$password' = \{u?\} \rhd password$
$biometric\_id' = \{u?\} \rhd biometric\_id$
$role' = \{u?\} \rhd role$

LoginManager
$Not\_login, Login : \mathbb{P}\ User$

$Not\_login \cap Login = \emptyset$

ordinaryLogin
$login\_user? : User$
$input\_username?, u! : username$
$input\_password? : Password$

$u! = input\_username?$

adminLogin
$login\_user? : User$
$input\_username?, u! : username$
$input\_biometric\_id? : PartsInformation$

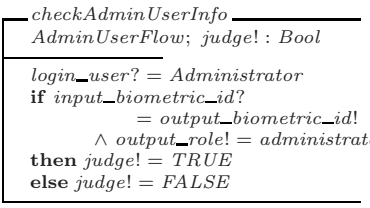
$u! = input\_username?$

$OrdinaryUserFlow \mathrel{\widehat=} ordinaryLogin \gg getUserInformation$

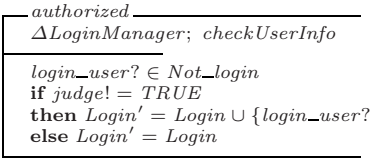
$AdminUserFlow \mathrel{\widehat=} adminLogin \gg getUserInformation$

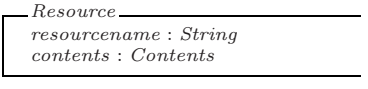
checkOrdUserInfo
$OrdinaryUserFlow;\ judge! : Bool$

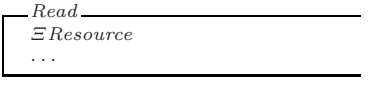
$login\_user? = Ordinary\_user$
if $input\_password? = output\_password!$
$\quad \land\ output\_role! = ordinary\_user$
then $judge! = TRUE$
else $judge! = FALSE$

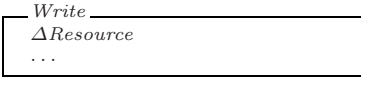


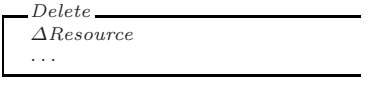
$checkUserInfo \mathrel{\widehat=} checkOrdUserInfo \lor checkAdminUserInfo$

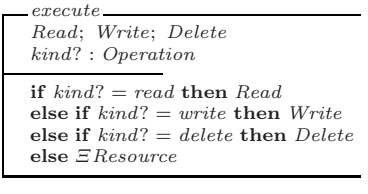


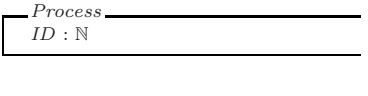


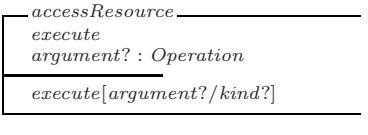


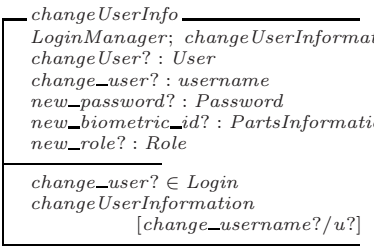


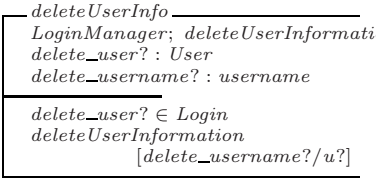


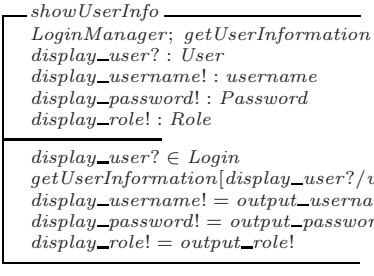


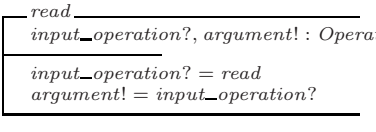


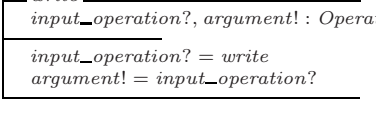


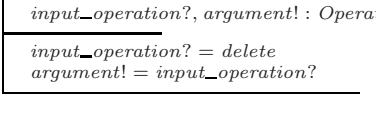












$readResource \mathrel{\widehat=} read \gg accessResource$

$writeResource \mathrel{\widehat=} write \gg accessResource$

$deleteResource \mathrel{\widehat=} delete \gg accessTable$

We now verify whether or not the specification satisfies the static requirements FIA_UAU.5.2 and FIA_USB.1.1 for login functions. The formalized FIA UAU.5.2 as a set of templates is as follows.

TEMPLATE **FIA_UAU.5.2**
$\forall \boldsymbol{TSF'} \mid$ users $\bullet$ **authentication_mechanisms**
$\quad \wedge$ condition_for_authentication $\bullet$ authorized_state
$\forall \boldsymbol{TSF'} \mid$ users $\bullet$ **authentication_mechanisms**
$\quad \wedge \neg$ condition_for_authentication $\bullet \wedge \neg$ authorized_state

The template formulas mean that the state of *users* becomes *authorized state* when *users* satisfy *condition for authentication* according to *authentication mechanisms* by *TSF*. When it does not satisfy the condition, it does not become so. In the specification, since *TSF* corresponds to the schema *LoginManager* which presides over user authentication, *TSF* is replaced by *LoginManager*. The *authentication mechanisms* correspond to the schema *checkUserInfo*, which is the actual authentication (identification) function to check the inputted identification. The *condition or authentication* is that the inputted identification is right. Ordinary users must be authenticated with their password and administrators must be authenticated with their biometric id. If a user is u, the *authorized state* will be that u' is included in the state *Login*. Thus, the templates are instantiated so that they fit into the specification, as follows:

$\forall\, LoginManager' \mid \forall\, u : User \mid u = Ordinary\_user \bullet checkUserInfo$
$\quad \wedge\, input\_password? = output\_password! \bullet u' \in Login$
$\forall\, LoginManager' \mid \forall\, u : User \mid u = Administrator \bullet checkUserInfo$
$\quad \wedge\, input\_biometric\_id? = output\_biometric\_id! \bullet u' \in Login$
$\forall\, LoginManager' \mid \forall\, u : User \mid u = Ordinary\_user \bullet checkUserInfo$
$\quad \wedge\, input\_password? \neq output\_password! \bullet u' \notin Login$
$\forall\, LoginManager' \mid \forall\, u : User \mid u = Administrator \bullet checkUserInfo$
$\quad \wedge\, input\_biometric\_id? \neq output\_biometric\_id! \bullet u' \notin Login$

These four formulas mean that each user is authorized according to the rule of FIA_UAU.5.2 if the right identification is inputted. These are deducible by Z/EVES, thus it can be said that the example satisfies FIA_UAU.5.2.

The formalized FIA_USB.1.1 as a template is as follows.

TEMPLATE **FIA_USB.1.1**
$\forall\, \boldsymbol{\Delta Resource} \bullet$ **a_resource_access_operation**

This formula means that *Resource* is always accessed via *a resource access operation*, because the change of *Resource* can be caused only by *a resource access operation*. In the specification, the schema *Resource* corresponds to *Resource*, and the schema *accessResource* corresponds to *a resource access operation*. The template is instantiated so that it can fit into the specification, as follows:

$\forall\, \Delta\ Resource \bullet accessResource$

This formula is deducible; it can be said that the example satisfies FIA_USB.1.1.

4.2 Example Verification with Model-Checking

The formalized security functional requirements are easily verifiable if the UML diagrams are formalized according to the well-established practices, which verify UML diagrams for behavior with model-checking [10,13]. The following is the description of dynamic behavior in Fig. 3 with NuSMV.

```
MODULE main
VAR
 input_username:boolean;
 input_password:boolean;
 User:{Login,Not_login};
 operation:{nothing,login,display,
                    create,change,delete};
DEFINE
 CheckUserInfo:=input_username
                         & input_password;
ASSIGN
--transition of input_username
 init(input_username):={0,1};
 next(input_username):={0,1}
--transition of input_password
 init(input_password):={0,1};
 next(input_password):={0,1};
--transition of User
 init(User):=Not_login;
 next(User):= case
  operation!=login :User;
  CheckUserInfo    :Login;
  1                :User;
  esac;
--transition of operation
 init(operation):=nothing;
 next(operation):= case
  User=Login  :{nothing,display,
                    create,change,delete};
  User=Not_login:{nothing,login};
  1              :nothing;
  esac;
```

The variables *input_username* and *input_password* are the user's inputs in the example system. Since it is not necessarily the right information, they assume 0 or 1 non-determinatively. The variable *User* denotes users in the example system. The initial state of the *User* is *Not_login*. If the inputted information when login was tried is right, *User* will transit from *Not_login* to *Login*. The variable *operation* denotes operations that users can do. In the state *Not_login*, the operation may choose *nothing* or *login* non-determinatively. In the state *Login*, it may choose *nothing, display, create, change* or *delete* non-determinatively.

The security functional requirement FIA_UID.1.2 specifies that a system must authenticate users before any operation [8]. The formalized FIA_UID.1.2 is as follows:

Template FIA_UID.1.2
$\Box$ ($\neg$ any_actions_occur W authorized_state)

That is, this template means that *any actions occur* is never satisfied until users become *authorized state* after satisfying *condition for authentication*.

In the description, the *authorized state* is that *User* transits from *Not_login* to *Login*. The *any actions occur* is any *operation* from among *display, create, change* or *delete*. Therefore, the formula is instantiated so that it can fit into the description above, as follows:

$$\Box\ (\ \neg\ (operation{=}display \vee operation{=}create \vee operation{=}change \vee operation{=}delete)\ \mathrm{W}\ User{=}Login))$$

This formula is evaluated as *true* by NuSMV. Thus, the example certainly satisfies FIA_UID.1.2.

5 Discussion

In this paper, we showed an example verifying the certified specification by ISO/IEC 15408 with UML diagrams. In the second step of the procedure, verifiers must formalize their system specifications. To mitigate difficulties in formalizing verification targets, we used RoZ & UML and NuSMV descriptions which are intuitive and easy to read [2]. If the target specifications are represented with UML, they are semi-automatically formalized to Z specifications with RoZ [6].

Moreover, recently theorem-proving and model-checking environments are being unified and various hybrid verification tools are being proposed [14]. Thus, we are discussing whether the security functional requirements can be described in CSP-OZ. CSP-OZ is the combination of Communicating Sequential Processes and Object-Z [7]. Our method will probably be able to verify static and dynamic requirements in a lump, if we can describe temporal operators in Z.

6 Concluding Remarks

In this paper, we have proposed a hybrid formal verification method of security specifications, based on the security functional requirements defined in ISO/IEC 15408 common criteria. We classified the requirements into static and dynamic requirements. After the classification, we formalized static requirements in Z notation and dynamic requirements in temporal logic. We also defined a formal verification procedure using theorem-proving with the formalized static requirements and model-checking with the formalized dynamic requirements. Consequently, one can test precisely and strictly whether specifications satisfy not only the static requirements but also the dynamic requirements (i.e., all the security functional requirements of ISO/IEC 15408) respectively. The verification becomes more precise and simple. Additionally, the classification also makes ISO/IEC 15408 easy to implement. We will publish the formalized requirements on our web site [1].

References

1. Advanced Information Systems Engineering Laboratory, Saitama University: Formal Descriptions of ISO/IEC 15408 Part 2, http://www.aise.ics.saitama-u.ac.jp/
2. Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P.: Systems and Software Verification –Model-Checking Techniques and Tools. Springer, Heidelberg (1999)
3. Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: NuSMV: a New Symbolic Model Verifier. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 495–499. Springer, Heidelberg (1999)
4. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (2000)
5. Common Criteria Project: Evaluated Product Files, http://www.commoncriteriaportal.org/public/files/epfiles/

6. Dupuy, S., Ledru, Y., Chabre-Peccoud, M.: An Overview of RoZ: A Tool for Integrating UML and Z Specifications. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 417–430. Springer, Heidelberg (2000)
7. Fischer, C.: CSP-OZ: a Combination of Object-Z and CSP. In: Proceedings of the 2nd IFIP Workshop on Formal Methods for Open Object-Based Distributed Systems, pp. 423–438. Chapman & Hall, Australia (1997)
8. ISO/IEC 15408 Standard: Information Technology - Security Techniques - Evaluation Criteria for IT Security (1999)
9. Jacky, J.: The Way of Z: Practical Programming with Formal Methods. Cambridge University Press, Cambridge (1997)
10. Latella, D., Majzik, I., Massink, M.: Automatic Verification of a Behavioural Subset of UML Statechart Diagrams using the SPIN Model Checker. Formal Aspects of Computing 11(6), 637–664 (1999)
11. Morimoto, S., Shigematsu, S., Goto, Y., Cheng, J.: Formal Verification of Security Specifications with Common Criteria. In: Proceedings of the 22nd Annual ACM Symposium on Applied Computing, pp. 1506–1512. ACM Press, New York (2007)
12. Saaltink, M.: The Z/EVES System. In: Till, D., Bowen, J.P., Hinchey, M.G. (eds.) ZUM 1997. LNCS, vol. 1212, pp. 72–85. Springer, Heidelberg (1997)
13. Schäfer, T., Knapp, A., Merz, S.: Model Checking UML State Machines and Collaborations. Electronic Notes in Theoretical Computer Science 55(3), 357–369 (2001)
14. YAHODA: Verification Tools Database, http://anna.fi.muni.cz/yahoda/

A The Dynamic Security Functional Requirements

FAU_ARP.1.1, FIA_UAU.7.1, FIA_UID.1.1, FIA_UID.1.2, FIA_UID.2.1, FAU_SAA.2.3, FMT_MSA.3.2, FAU_SAA.3.3, FMT_MTD.2.2, FAU_SAA.4.3, FMT_SAE.1.2, FAU_STG.2.3, FAU_STG.3.1, FAU_STG.4.1, FCO_NRO.1.1, FCO_NRO.2.1, FCO_NRR.1.1, FCO_NRR.2.1, FPR_UNO.2.2, FPT_AMT.1.1, FPT_FLS.1.1, FDP_ETC.1.1, FDP_ETC.2.1, FDP_ETC.2.3, FDP_ETC.2.4, FPT_PHP.2.3, FPT_PHP.3.1, FDP_IFF.1.2, FPT_RCV.1.1, FPT_RCV.2.1, FPT_RCV.3.1, FDP_IFF.2.2, FPT_RPL.1.2, FPT_RVM.1.1, FDP_IFF.2.7, FDP_ITC.1.1, FDP_ITC.1.2, FDP_ITC.1.3, FPT_SSP.1.1, FDP_ITC.2.1, FPT_SSP.2.1, FDP_ITC.2.5, FDP_ITT.1.1, FDP_ITT.2.1, FPT_TRC.1.2, FDP_ITT.2.2, FPT_TST.1.1, FRU_FLT.1.1, FRU_FLT.2.1, FRU_PRS.1.1, FRU_PRS.1.2, FRU_PRS.2.1, FRU_PRS.2.2, FRU_RSA.1.1, FRU_RSA.2.1, FRU_RSA.2.2, FDP_SDI.2.2, FTA_MCS.1.1, FTA_MCS.1.2, FTA_MCS.2.1, FTA_MCS.2.2, FTA_SSL.1.1, FTA_SSL.1.2, FIA_AFL.1.1, FTA_SSL.2.1, FIA_AFL.1.2, FTA_SSL.2.2, FTA_SSL.3.1, FTA_TAB.1.1, FTA_TAH.1.1, FTA_TAH.1.2, FIA_UAU.1.1, FIA_UAU.1.2, FIA_UAU.2.1, FTP_ITC.1.2, FTP_ITC.1.3, FTP_TRP.1.2

ONN the Use of Neural Networks for Data Privacy

Jordi Pont-Tuset[1], Pau Medrano-Gracia[1], Jordi Nin[2], Josep-L. Larriba-Pey[1], and Victor Muntés-Mulero[1]

[1] DAMA-UPC, Computer Architecture Dept.
Universitat Politècnica de Catalunya
Campus Nord UPC, 08034, Barcelona, Spain
{jpont,pmedrano,larri,vmuntes}@ac.upc.edu
[2] IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB s/n, 08193, Bellaterra, Spain
jnin@iiia.csic.es

Abstract. The need for data privacy motivates the development of new methods that allow to protect data minimizing the disclosure risk without losing valuable statistical information. In this paper, we propose a new protection method for numerical data called *Ordered Neural Networks* (ONN). ONN presents a new way to protect data based on the use of *Artificial Neural Networks* (ANNs). The main contribution of ONN is a new strategy for preprocessing data so that the ANNs are not capable of accurately learning the original data set. Using the results obtained by the ANNs, ONN generates a new data set similar to the original one without disclosing the real sensible values.

We compare our method to the best methods presented in the literature, using data provided by the US Census Bureau. Our experiments show that ONN outperforms the previous methods proposed in the literature, proving that the use of ANNs is convenient to protect the data efficiently without losing the statistical properties of the set.

Keywords: Perturbative protection methods, Data preprocessing, Artificial Neural Networks, Privacy in statistical databases.

1 Introduction

Managing confidential data is a common practice in any organization. In many cases, these data contain valuable statistical information required by third parties and, thus, privacy becomes essential, making it necessary to release data sets preserving the statistics without revealing confidential information. This is a typical problem, for instance, in statistics institutes.

In this scenario, an intruder might try to re-identify a percentage of the protected individuals by applying *Record Linkage* (RL) techniques [1,2] between some attributes in the protected data set and some attributes obtained from

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 634–645, 2008.

other data sources which includes at least one identifier[1]. Depending on the non-protected attributes obtained by the intruder from other sources, the probability of re-identifying individuals increases, in other words, the larger the number of attributes known, the higher the probability to reveal the identity of the individuals in the protected data set.

Special efforts have been made to develop a wide range of protection methods. These methods aim at guaranteeing an acceptable level of protection of the confidential data. The number of techniques applied to protect data is very large, ranging from simply swapping values of the data set [3] to using complex data models [4].

We present a new type of perturbative protection method (according to the classification presented in [5]) called *Ordered Neural Networks* (ONN). ONN is based on the use of an array of *Artificial Neural Networks* (ANNs) to protect the numerical values of a data set. ONN consists of a set of steps which includes data preprocessing, the learning process using the ANNs array and the protection step. The combination of these parts reproduces the original data in an inaccurate way. This approximation of the real data constitutes our protected values, our goal is to learn a pseudo-identity function.

If we consider the type of data that ONN can protect, ONN is considered a numerical protection method because arithmetic operations can be performed with both original and protected data (*e.g.* age or income). Note that a numerical attribute does not necessarily have an infinite range, as in the case of age. However, categorical data where standard arithmetic operations do not make sense, like academic degree or hair color, are unsuitable for ONN and they have to be protected using other techniques.

This paper is organized as follows. Section 2 presents some ANNs basics and a brief introduction to protection methods. In Section 3, we present a detailed description of the ONN method. Section 4 presents some results. Finally, Section 5 draws some conclusions and presents some future work.

2 Preliminaries

In this section, we introduce the basic knowledge necessary to follow the details of our method. First, we give a brief description of a general ANN. Second, we point out the basic characteristics of the *Backpropagation algorithm* needed to understand this work. Third, we present some protection methods found in the literature. Finally, we describe the score method to evaluate them.

2.1 Artificial Neural Networks

An *Artificial Neural Network* (ANN) is an interconnected network of simple processing elements which are also called *neurons*. Each artificial neuron computes the output by weighting all of its inputs and then applying a final output function

[1] The identifier attributes are used to identify the individual unambiguously. A typical example is the passport number.

called *activation function*. By changing the values of these connection weights the network can collectively produce complex overall behavior. The process of changing these values is called *training*. In this work, we use the *Backpropagation algorithm* presented in [6] for the training phase.

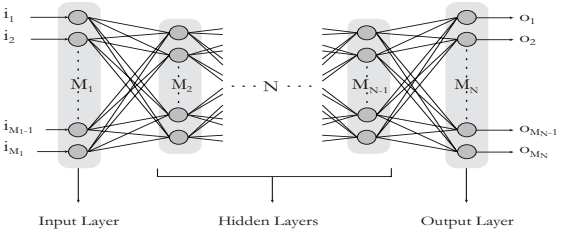


Fig. 1. Artificial Neural Network schema

As we can observe in Figure 1, neurons are arranged in groups called *layers*. Generally, the pth network layer contains M_p neurons and each processing element $(k = 1, ..., M_p)$ computes a single stimulated response as follows:

$$o_k^p = f_k^p(net_k^p) \qquad net_k^p = \sum_{j=1}^{M_{p-1}} w_{jk}^p i_j + \theta_k^p$$

where o_k^p is the final output value returned by each neuron in the pth layer, f_k^p is the activation function, w_{jk}^p is the weight assigned to the connection between neurons j and k, and θ_k^p is called *bias* and it is further detailed in [7]. Note that f_k^p is applied to the net_k^p value that each neuron receives from the neurons of a previous layer $(i_j, j = 1, ..., M_{P-1})$ weighted with a certain w_{jk}^p.

Here, for all layers and for $k = 1, ..., M_p$, we consider a sigmoid activation function:

$$f_k^p(x) = \frac{1}{1 + e^{-cx}} \qquad c \geq 0 \tag{1}$$

where parameter c modifies the shape of this function and it is directly proportional to the slope at the origin of the activation function. Thus, increasing c, the ability of the ANN to separate patterns is higher, but the stability of the iterative algorithm decreases.

Although they have the same structure, layers are classified into three types: the input layer, that receives the external values; the hidden layers, that receive the information from a previous layer in the ANN and pass the results on to the next layer; and the output layer, that returns the results of the ANN. The number of hidden layers is arbitrary and depends on the scenario.

The number of neurons in the hidden layers influences the complexity of the patterns that the net will be able to learn. The larger the number of neurons considered, the larger the complexity of the patterns an ANN is able to recognize. Note that, increasing the number of neurons, the complexity of the ANN structure becomes larger.

We assume that there are neither feedback connections between neurons nor layer-bypassing connections. This means that the inputs of each layer depend exclusively on the previous layer outputs.

2.2 The Backpropagation Algorithm

The *Backpropagation algorithm* allows the ANN to learn from a predefined set of input-output example pairs. The basic idea of this method is to adjust the weights of each processing element iteratively.

After initializing the network weights using uniformly distributed random numbers, the input data are propagated throughout the ANN. An error value is then computed at the output layer (first cycle) and, afterwards, at all the hidden layers (second cycle). In this work, for the lth training vector, the error (E_l) is computed as the sum of the squared difference between the desired and the actual output of each neuron. The overall error (E) for all the training set is then the sum of all these errors ($E = \sum_l E_l$).

Based on this error, connection weights are updated using an *iterative steepest descent* method. To apply this method, we consider the direction $-\nabla E_l$ and the learning-rate parameter η so that weights are updated from step t to step $t+1$ as follows:

$$w_{jk}(t+1) = w_{jk}(t) - \eta \frac{\partial E_l}{\partial w_{jk}} \tag{2}$$

Note that parameter η tunes the norm of the vector that modifies the weight values. This parameter has an effect on the range of the weights explored and the probability of divergence. Weights in hidden layers are similarly adjusted [6].

In this paper, we assume that the learning process is complete when the overall error E is below a certain tolerance factor. When the learning process is complete, the ANN has theoretically internalized the hidden patterns in the provided examples, meaning that it should be able to approximate the given training examples.

2.3 Data Protection Methods

Good surveys about protection methods can be found in the literature [8,9]. The experiments presented in these previous works conclude that, among a wide range of protection methods, *Rank Swapping* (RS-p) [3] and *Microaggregation* (MIC-vm-k) [10] obtain the best protection rates for numerical data. Other protection method like noise addition, lossy compression, data distortion or resampling are usually poorly ranked and, therefore, they are not considered further in this paper.

RS-p sorts the values of each attribute. Then, each value is swapped with another sorted value chosen at random within a restricted range of size p. MIC-vm-k builds small clusters from v variables of at least k elements and replaces original values by the centroid of the clusters that the record belongs to.

In this paper, we compare ONN with these two methods. Specifically, we have chosen the best five parameterizations for RS-p and MIC-vm-k as presented in [9].

2.4 Scoring Protection Methods

In order to measure the quality of a protection method, we need a protection quality measurement that assigns a score to a method depending on its capacity to: (i) make it difficult for an intruder to reveal the original data and (ii) to avoid the information loss in the protected data set. In this paper, we use the score defined in [5] which has been used in several other works.

In order to calculate the *score*, we must first calculate some statistics:

- **Information Loss (IL):** Let X and X' be matrices representing the original and the protected data set, respectively. Let V and R be the covariance matrix and the correlation matrix of X, respectively; let $\overline{X}$ be the vector of variable averages for X and let S be the diagonal of V. Define V', R', $\overline{X}'$, and S' analogously from X'. The information loss is computed by averaging the mean variations of $X - X'$, $V - V'$, $S - S'$, and the mean absolute error of $R - R'$ and multiplying the resulting average by 100.
- **Disclosure Risk (DR):** We use the three different methods presented in [2] in order to evaluate DR: (i) *Distance Linkage Disclosure risk* (DLD), which is the average percentage of linked records using distance based *Record Linkage* (RL), (ii) *Probabilistic Linkage Disclosure risk* (PLD), which is the average percentage of linked records using probabilistic based RL and (iii) *Interval Disclosure risk* (ID) which is the average percentage of original values falling into the intervals around their corresponding masked values. The three values are computed over the number of attributes that the intruder is assumed to know. Disclosure Risk is computed as $DR = 0.25 \cdot DLD + 0.25 \cdot PLD + 0.5 \cdot ID$.
- **Score:** The final score measure is computed by weighting the presented measures and it was also proposed in [5]: $score = 0.5\,IL + 0.5\,DR$.

A simplified version of the score can be used without involving PLD. In this case, the score is $score_{simp} = 0.5\,IL + 0.25\,DLD + 0.25\,ID$.

Due to the large execution cost of PLD, we use the $score_{simp}$ to save execution time of the experiments run in the next section. This simplified score was used in [11] for similar reasons. It is important to highlight that the better a protection method, the lower its score.

3 Ordered Neural Networks

We propose a new protection method called *Ordered Neural Networks* (ONN). ONN uses an array of ANNs for learning the numerical data set to be protected. ANNs themselves would be able to perfectly learn all data using a structure as complex as necessary. However, this is not our goal. We want to obtain an inaccurately learned data set which is similar enough to be representative of the original data set, but different enough not to reveal the original confidential values. In order to fulfill these requirements, our new proposal is based on the use of simple ANN structures combined with preprocessing techniques.

As shown in Figure 2, ONN can be decomposed in several steps, namely (i) vectorization, (ii) sorting, (iii) partitioning, (iv) normalization, (v) learning and (vi)

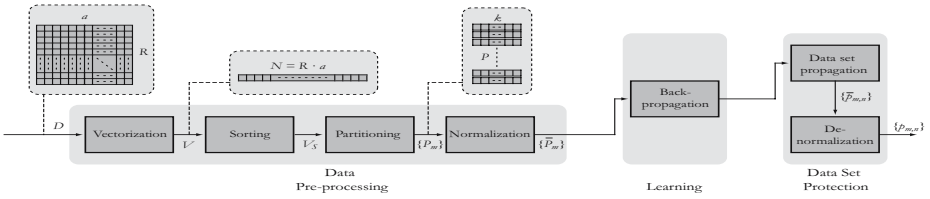


Fig. 2. ONN general schema

protecting data. Steps (i) through (iv) preprocess data in order to facilitate the learning in step (v). Once the learning process finishes, data are protected in step (vi). Following, we go into further detail about the process steps mentioned above.

(i) Vectorization

The vectorization step gathers all the values in the data set in a single vector, independently from the attribute they belong to. This way, we ignore the attribute semantics and, therefore, the possible correlation between two different attributes in the data set.

Formally speaking, let $\mathcal{D}$ be the original data set to be protected. We denote by R the number of records in $\mathcal{D}$. Each record consists of a numerical attributes or fields. We assume that none of the registers contains blanks. We denote by N the total number of values in $\mathcal{D}$. As a consequence, $N = R \cdot a$.

Let V be a vector of size N. ONN treats values in the data set as if they were completely independent. In other words, the concept of record and field is ignored and the N values in the data set are placed in V.

(ii) Sorting

Since the values in the vectorized data set belong to different source attributes, they present a pseudo-random aspect and it becomes very difficult for an ANN to learn patterns from them. In order to simplify the learning process, ONN sorts the whole data set. This way, the accuracy level increases, since the ANN has to learn an easy non-decreasing function instead of a more complex function.

Formally, V is sorted increasingly. Let us call V_s the sorted vector of size N containing the sorted data and v_i the ith element of V_s, where $0 \leq i < N$.

(iii) Partitioning

Once the data in the vector is sorted, we could start the learning process. However, given that the size of the data set may be very large, using a single ANN would make this process very difficult. Because of this, we use an array of ANNs. The main idea is to let each ANN learn a single disjunct chunk of the data set. This way, the learning process is faster and the system is able to fit better the original data set.

We define $1 \leq P \leq N$ as the number of partitions (subvectors) into which V_s is divided. All the partitions contain the same number of values (P must divide N). We call $k = N/P$ the number of values in each partition.

We denote by P_m the mth partition ($0 \leq m < P$). Let $v_{m,n}$ be defined as the nth element of P_m: $v_{m,n} := v_{mk+n} \quad n = 0 \ldots k-1 \quad m = 0 \ldots P-1$.

(iv) Normalization

The ability of an ANN to learn depends on the range of values of the input data set and the activation function. In order to make the learning process possible, it is necessary to normalize the input data set. An input value x is desired to range between two fitted values $-B_{in}$ and B_{in}, where $B_{in} \in \mathbb{R}^+$, so that the output values of the activation function fall between a certain $B1_{out}$ and $B2_{out}$. The basic idea is to adjust the boundaries (B_{in}) in order to make the normalized input values fit in the range where the slope of the activation function is relevant, as explained in [7]. Figure 3 shows the activation function used and the range of values after normalization.

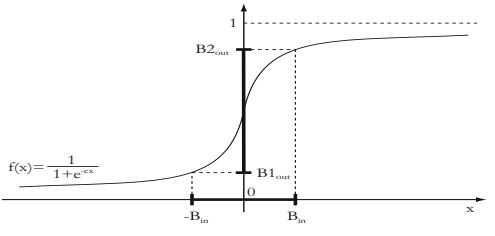


Fig. 3. Activation function and range of values after normalization

More formally, using Equation (1):

$$B1_{out} = \frac{1}{1 + e^{c(B_{in})}} \qquad B2_{out} = \frac{1}{1 + e^{-c(B_{in})}}$$

Let $\max_m$ and $\min_m$ be the maximum and the minimum values in the mth partition:

$$\max_m := \max_{0 \leq i < k} \{v_{m,i}\} \qquad \min_m := \min_{0 \leq i < k} \{v_{m,i}\}$$

Let $\overline{V}_s$ be the sorted vector containing the normalized input data and $\overline{v}_i$ the ith element of $\overline{V}_s$. Analogously, let $\overline{P}_m$ be the mth normalized partition and $\overline{v}_{m,n}$ the nth element of $\overline{P}_m$.

The normalized input values are defined as:

$$\begin{cases} \overline{v}_{m,n} := 2\, B_{in}\, \frac{v_{m,n} - \min_m}{\max_m - \min_m} - B_{in} & \text{if } \max_m \neq \min_m \\ \overline{v}_{m,n} := 0 & \text{if } \max_m = \min_m \end{cases}$$

where $0 \leq m < P$ and $0 \leq n < k$.

Note that $\max_m = \min_m$ means that all the values in the partition are the same. In this case, the normalized value is set to 0, in other words, it is centered in the normalization range.

Analogously, the desired output, which it is denoted by $y_{m,n}$, where $0 \leq m < P,\ 0 \leq n < k$, is normalized between $B1_{out}$ and $B2_{out}$:

$$\begin{cases} \overline{y}_{m,n} := (B2_{out} - B1_{out}) \frac{y_{m,n} - \min_m}{\max_m - \min_m} + B1_{out} \\ \overline{y}_{m,n} := 0.5 \end{cases}$$

where the first component of this expression is used when $\max_m \neq \min_m$ and the second component is used otherwise.

The range of values of the desired outputs is then $(B1_{out},\ B2_{out})$. Notice that, in the input layer, the outputs fall in the same range, making the training process easier.

(v) Learning

Finally, ONN creates an array of ANNs in order to learn the whole data set, where each ANN is associated to a partition. Therefore, the array contains P ANNs. The objective for each ANN is to learn the values in its corresponding partition. However, an specific ANN is not only fed with the values in that partition, but uses the whole data set to learn. In some sense, using the whole data set, we are adding non-linear noise to the learning process by using input data that is not correlated with the data to be learned.

This learning process, in conjunction with the preprocessing techniques previously explained, make the ANNs internalize the data patterns distortedly. This way, the reproduced data are resembling enough to the original, to maintain their statistical properties, but dissimilar enough not to disclose the original confidential values.

Specifically, the ANN that learns the values of a partition $\overline{P}_m$ receives the nth value of that partition, $\overline{v}_{m,n}$, together with the $P-1$ nth values of the remaining partitions. Since the network is intended to learn all values $\overline{v}_{m,n}$ in $\overline{P}_m$, the desired output is set to $\overline{y}_{m,n} = \overline{v}_{m,n}$. This process is repeated iteratively until the learning process finishes.

In our proposal, each ANN contains three layers: the input layer, a single hidden layer and the output layer, which can be described as follows:

Input Layer. The input layer consists of $M_1 = P$ neurons. Each of them takes the data from a different partition as input. That is, input of the ith neuron comes from partition $\overline{P}_i$.

Hidden Layer. The hidden layer has $M_2 = n_h$ neurons. As explained in [7], n_h has an effect on the speed of the learning process and the ability of the network to learn complex patterns.

Output Layer. The output layer consists of one single neuron ($M_3 = 1$).

The structure of the array of ANNs is shown in Figure 4.

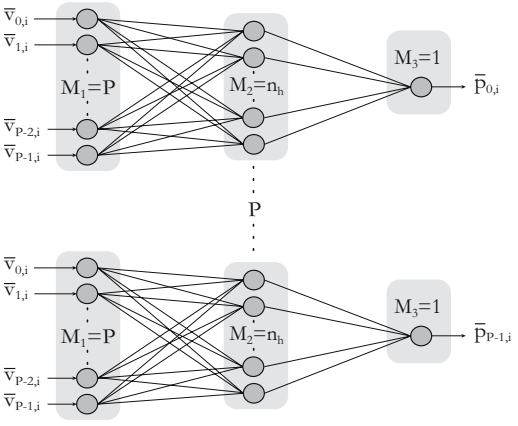


Fig. 4. Array of ANNs used by ONN

All networks learn the original data updating their weights by using the iterative *Backpropagation algorithm* explained in Section 2.2. The quality of the data protection depends basically on the ANN structure and the preprocessing parameters.

(vi) Protecting Data

Once the ANNs have been trained, and therefore the weights updated, the last step obtains the protected values for the data set. This includes the data set propagation and the de-normalization, as explained below.

Let $\overline{p}_{m,n}$ be the protected value for $\overline{v}_{m,n}$. As mentioned before, the mth ANN of the array has been trained to reproduce $\overline{v}_{m,n}$ when the values in the P input neurons is $\overline{v}_{0,n}, ..., \overline{v}_{P-1,n}$. This way, $\overline{p}_{m,n}$ is defined as the output obtained when having $\overline{v}_{0,n}, ..., \overline{v}_{P-1,n}$ as input of the mth already trained ANN.

Finally, the protected value $p_{m,n}$ for $v_{m,n}$ is obtained by de-normalizing $\overline{p}_{m,n}$ as follows:

$$p_{m,n} = \min_m + \frac{(\overline{p}_{m,n} - B1_{out})(\max_m - \min_m)}{B2_{out} - B1_{out}}$$

If $\max_m = \min_m$ the expression used is $p_{m,n} = (\overline{p}_{m,n} + 0.5)v_{m,n}$.

The protected values $p_{m,n}$ are placed in the protected data set in the same place occupied by the corresponding $v_{m,n}$ in the original data set. This way, we are undoing the sorting and vectorization steps.

4 Experiments

In order to test ONN, we use a data set provided by the US Census Bureau, described in detail in [12]. The *Census* data set is used in other works like [13,14]. This data set contains 1080 records consisting of 13 attributes (which is equal to

14040 values to be protected). We compare ONN with the best ranked protection methods presented in the literature, called *Rank Swapping* and *Microaggregation*, described in Section 2.3.

The ONN parameters are: number of attributes used by an intruder to reveal data using RL techniques (V), number of partitions (P), normalization range size (B), learning rate parameter (E), activation function slope parameter (C) and number of neurons in the hidden layer (H).

We have divided our experiments into two different scenarios. First, we assume that the intruder only has half of the original protected attributes ($V = 7$), this scenario was used in [15]. Second, we assume that the intruder is able to obtain all the original attributes ($V = 13$). This scenario could be considered the most favorable scenario for the intruder.

The values selected for each factor were chosen according to empirical results in order to use reasonable and realistic values. The best-score parameters set, for both cases of the V parameter, are presented in Table 1. These sets of configurations are later referred to as ONN-A, ONN-B, etc.

We have run RS-p and MIC-vm-k using their best five parameterizations, extracted from [9], so we can fairly compare to them.

Table 1. ONN parameters used in the experiments

	P	B	E	C	H
A	8	0.8	0.4	4.0	8
B	10	0.8	0.1	3.5	8
C	10	0.8	0.1	3.0	8
D	10	0.8	0.1	4.0	8
E	8	0.8	0.1	4.0	2

($V = 7$)

	P	B	E	C	H
a	8	0.8	0.4	4.0	8
b	8	0.8	0.3	4.0	8
c	8	0.8	0.1	4.0	2
d	10	0.8	0.1	3.5	8
e	8	0.8	0.1	3.5	2

($V = 13$)

Table 2.a shows the scores in the first scenario. As we can observe, the IL when protecting data using ONN is, in general, lower than that obtained using RS-p or MIC-vm-k. This means that ONN is able to fit the data set better than the other two approaches. These results are coherent with the methodology used by ONN to protect data. Since ONN is trained using the data set after being preprocessed, the patterns learned depend on values that come from different individuals in the original data set. Because of this, an intruder should know the values in each partition to be able to understand the learned patterns. Since this information is no longer available after protecting the data, ONN can get lower ILs while preserving relatively good rates of DR.

Regarding DR, the best disclosure risk corresponds to RS-p. These results make sense because when the intruder has a reduced set of variables, it is very difficult to re-identify individuals because, by swapping, RS-p mixes values from different individuals. ONN presents a good DR, better than that obtained by MIC-vm-k. However, due to the RS-p simplicity, a specific record linkage method for *Rank Swapping* can be performed, as presented in [15]. There the authors

Table 2. Average results of IL, DLD, PLD, ID using (a) 7 variables and (b) 13 variables

Method	IL	DR	SCR
RS-14	23.83	24.21	24.02
RS-17	27.40	21.87	24.64
RS-12	21.08	27.83	24.45
RS-15	27.44	23.62	25.53
RS-13	25.39	26.35	25.87
MIC4m17	23.98	31.67	27.82
MIC4m19	26.10	31.09	28.59
MIC4m11	21.27	36.22	28.74
MIC3m20	21.95	35.85	28.90
MIC3m15	18.98	39.33	29.15
ONN-A	20.42	26.25	23.33
ONN-B	20.59	26.95	23.77
ONN-C	20.31	27.26	23.78
ONN-D	20.66	26.96	23.81
ONN-E	22.38	25.65	24.01

(a)

Method	IL	DR	SCR
RS-14	23.83	31.32	27.58
RS-17	27.40	28.43	27.92
RS-12	21.08	35.83	28.46
RS-15	27.44	30.51	28.98
RS-13	25.39	33.79	29.59
MIC4m17	23.98	40.80	32.39
MIC4m19	26.10	40.10	33.10
MIC4m11	21.27	46.45	33.86
MIC3m20	21.95	46.70	34.33
MIC3m15	18.98	50.94	34.96
ONN-a	21.45	33.52	27.49
ONN-b	22.23	32.99	27.61
ONN-c	22.38	33.04	27.71
ONN-d	20.59	34.88	27.73
ONN-e	22.56	33.20	27.88

(b)

show that the real DR of the RS-p is much larger than the values presented in this work using general RL methods.

Observing the scores, ONN shows to be the best protection method among those presented in this work and, therefore, all the methods studied in [9]. The scores obtained by ONN are better than those obtained by RS-p and MIC-vm-k. Note that, although the DR is lower for RS-p, the scores show that ONN is better ranked, meaning that the benefits obtained by avoiding the IL compensate for the increase in the DR.

Table 2.b shows similar results for the second scenario, where the intruder has all the variables. As we can observe, the results are very similar to the first scenario. It is important to notice that the larger the number of variables known by the intruder the more similar the DR presented by RS-p and ONN.

5 Conclusions and Future Work

In this paper, we have presented ONN, a new method for protecting data minimizing the information loss. To our knowledge, no previous attempts had been made to use ANNs for this purpose.

The use of ANNs, combined with other preprocessing techniques, to create a protected data set from the original data has shown to be better than previous techniques. Specifically, we have proven that ONN reduces the combined disclosure risk and the information loss metric beyond the best approaches presented in the literature.

Future directions of this work include the establishment of a set of criteria that allow to automatically tune the parameters of our method.

Acknowledgments

The authors from UPC want to thank Generalitat de Catalunya for its support through grant number GRE-00352 and Ministerio de Educación y Ciencia of Spain for its support through grant TIN2006-15536-C02-02. Jordi Nin wants to thank the Spanish Council for Scientific Research (CSIC) for his I3P grant and the Spanish MEC for its support through ARES CONSOLIDER INGENIO 2010 CSD2007-00004 and eAEGIS-TSI2007-65406-C03-02.

References

1. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. on KDE 19(1), 1–16 (2007)
2. Torra, V., Domingo-Ferrer, J.: Record linkage methods for multidatabase data mining. In: Information Fusion in Data Mining, pp. 101–132. Springer, Heidelberg (2003)
3. Moore, R.: Controlled data swapping techniques for masking public use microdata sets. U.S. Bureau of the Census (Unpublished manuscript) (1996)
4. Burridge, J.: Information preserving statistical obfuscation. Statistics and Computing 13, 321–327 (2003)
5. Domingo-Ferrer, J., Torra, V.: Disclosure control methods and information loss for microdata. Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies, 111–133 (2001)
6. Rojas, R.: Neural Networks: A Systematic Introduction. Springer, Heidelberg (1996)
7. Freeman, J.A., Skapura, D.M.: Neural Networks: Algorithms, Applications and Programming Techniques, pp. 1–106. Addison-Wesley Publishing Company, Reading (1991)
8. Adam, N.R., Wortmann, J.C.: Security-control for statistical databases: a comparative study. ACM Computing Surveys 21, 515–556 (1989)
9. Domingo-Ferrer, J., Torra, V.: A quantitative comparison of disclosure control methods for microdata. In: Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies, pp. 111–133. Elsevier Science, Amsterdam (2001)
10. Domingo-Ferrer, J., Mateo-Sanz, J.M.: Practical data-oriented microaggregation for statistical disclosure control. IEEE Trans. on KDE 14, 189–201 (2002)
11. Sebé, F., Domingo-Ferrer, J., Mateo-Sanz, J.M., Torra, V.: Post-masking optimization of the tradeoff between information loss and disclosure risk in masked microdata sets. In: Domingo-Ferrer, J. (ed.) Inference Control in Statistical Databases. LNCS, vol. 2316, pp. 187–196. Springer, Heidelberg (2002)
12. CASC Project, http://neon.vb.cbs.nl/casc/
13. Domingo-Ferrer, J., Mateo-Sanz, J.M., Torra, V.: Comparing sdc methods for microdata on the basis of information loss and disclosure risk. In: Pre-proceedings of ETK-NTTS 2001, vol. 2, pp. 807–826. Eurostat (2001)
14. Yancey, W., Winkler, W., Creecy, R.: Disclosure risk assessment in perturbative microdata protection. In: Domingo-Ferrer, J. (ed.) Inference Control in Statistical Databases. LNCS, vol. 2316, pp. 135–152. Springer, Heidelberg (2002)
15. Nin, J., Herranz, J., Torra, V.: Rethinking rank swapping to decrease disclosure risk. In: DKE (in press, 2007)

Threshold Privacy Preserving Keyword Searches

Peishun Wang[1], Huaxiong Wang[1,2], and Josef Pieprzyk[1]

[1]Centre for Advanced Computing – Algorithms and Cryptography
Department of Computing, Macquarie University, Australia
{pwang, hwang, josef}@ics.mq.edu.au
[2]Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
hxwang@ntu.edu.sg

Abstract. We consider the following problem: users of an organization wish to outsource the storage of sensitive data to a large database server. It is assumed that the server storing the data is untrusted so the data stored have to be encrypted. We further suppose that the manager of the organization has the right to access all data, but a member of the organization can not access any data alone. The member must collaborate with other members to search for the desired data. In this paper, we investigate the notion of threshold privacy preserving keyword search (TPPKS) and define its security requirements. We construct a TPPKS scheme and show the proof of security under the assumptions of intractability of discrete logarithm, decisional Diffie-Hellman and computational Diffie-Hellman problems.

Keywords: Keyword search, threshold, key distribution, secure index.

1 Introduction

Threshold privacy preserving keyword searches are particularly useful in the following scenario. Assume that users of an organization wish to outsource storage of their sensitive information to a large database server. However, the server storing the data is untrusted, and other members of the organization alone cannot be trusted. Hence, all data have to be submitted in an encrypted form. Only the manager of the organization has the right to access all data, and any member of the organization must collaborate with others to search for the desired data. For example, big intelligence or police organizations (such as CIA and FBI) normally use one central server, where all information is being stored. It is reasonable to assume that the access to the data on the server has to be managed according to the organization security policy. It is clear that all the IT technical staff who maintains the server should not have an access to data stored. However, teams working on specific cases should have access to the relevant data. Each team is normally led by a head officer. The head is responsible for the enforcement of security policy, and depending on the sensitivity of data, she may use secret sharing to control the access to appropriate data. In particular, the head generates

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 646–658, 2008.

the secret and distributes shares of the secret among the members of the team according to the access structure derived from the security policy. Note that now any group of officers belonging to the access structure can access (collectively) a relevant data.

There are a lot of solutions in literature for a single-user and multi-users keyword search over encrypted data, but there is no scheme for threshold access. In particular, we consider the following scenario. Every member uses her share of the secret to generate a share of a trapdoor for a list of keywords. After receiving the encrypted data from the server, all members compute their decryption shares, and then pool them together in order to recover the plaintext. The validity of every member's share has to be checked. Note that the shares of the members must be used in such a way that precludes a leak of any information about the shares. In this paper, we study the threshold privacy preserving keyword search, and provide a practical solution to this problem.

Contributions. The contribution of this paper is three-fold. (1) We define the notion of threshold privacy preserving keyword search (TPPKS) and propose suitable privacy requirements, i.e. data privacy and privacy of the secret and its shares. (2) We construct a TPPKS scheme based on Shamir secret sharing, Boneh and Franklin's ID-based cryptosystem and the group computation, and prove the security under the assumptions of intractability of discrete logarithm, decisional Diffie-Hellman and computational Diffie-Hellman problems. (3) Our scheme has the following attractive properties: shares are verified without leaking any information about them, any invalid share fails the verification, and there is no information disclosed about the shares after they have been used.

Organization. In Section 2, we review the related work. Section 3 provides models and necessary background. In Section 4, we construct a TPPKS scheme. Section 5 proves the security of the proposed scheme. Finally Section 6 includes a conclusion and future research.

2 Related Works

There is a number of research works investigating privacy preserving keyword search over encrypted data. For the single-user setting, the protocols based on single keyword search were studied in [1,6,10] and conjunctive keyword search schemes were proposed in [3,4,7]. For the multi-user setting, there are two protocols published in [9,11].

Song, Wagner, and Perrig [10] introduced a model of privacy preserving keyword search over encrypted data. In their searchable symmetric key encryption scheme, each word is encrypted separately and extra information is embedded into the ciphertext such that it is used in conjunction with a piece of trapdoor for a keyword to test whether the keyword is present in a document. The scheme needs a computation overhead that is linear in the size of a document, fails to deal with compressed data, and reveals statistical information about the distribution

of the plaintext. To address these shortcomings, Goh [6] presented a scheme called secure indices, in which each keyword in a document is processed by a pseudo-random function twice, and the final output is inserted into a Bloom filter. The trapdoor consists of an indicator of what bits in the secure index should be tested. An immediate consequence of application of Bloom filters is that the Goh's scheme introduces false positives. Boneh *et al.* [1] presented public key schemes for keyword search over encrypted data. Their main idea is that each keyword is encrypted under the public key in such a way that the user is able to generate the trapdoor for any particular keyword with her private key. Their schemes require the server to perform a linear scan through all the document contents for every query, need secure channels to transmit trapdoors, and never refresh the trapdoor for a keyword.

Golle *et al.* [7] pioneered the construction of conjunctive keyword search over encrypted data. In their first scheme, the server checks the two hash codes of the group points associated with the keywords to find the desired data. In the second scheme, the server verifies the two outputs of bilinear pairing that takes the keywords as an input and checks if the keyword is in the document or not. Note that there is a substantial overhead related to the transmission of the trapdoors, and their second construction is arguably inefficient, particularly for a large collection of documents. Byun *et al.* [4] designed a conjunctive keyword search scheme in the random oracle model, which requires only constant communication and storage costs. In their scheme, each keyword is hashed to a group, and then mapped into another group by bilinear pairing. On receiving a trapdoor, the server tests two results of bilinear pairing and guesses if the document contains the keywords. Recently Boneh and Waters [3] further developed a public-key encryption scheme for conjunctive keyword search from a generalization of anonymous identity based encryption. The scheme supports comparison queries (such as greater-than) and general subset queries.

Based on Goh's scheme [6], Park *et al.* [9] proposed keyword search schemes for groups. The main idea is to use one-way hash chain in reverse order to make group session keys, encryption keys and index generation keys. All group members use identical secret keys to make secure indices and trapdoors, and a set of new group keys must be generated for each session. Therefore, the size of a query becomes larger as the number of sessions increases. Wang *et al.* [11] address the shortcomings of the schemes from [9], propose a new model called common secure index for conjunctive keyword-based retrieval (CSI-CKR), which is based on dynamic accumulators, Pailliers cryptosystem and blind signatures.

It should be noted that, to prove the security, all existing schemes assume that the number of keywords associated with a document remains fixed. This constraint can be satisfied by simply adding null keywords to the list.

3 Preliminaries

Notation. Throughout this paper, we use the following notation. Let $a \xleftarrow{R} A$ denote that an element a is chosen uniformly at random from the set A. PPT

denotes *probabilistic polynomial time*. For any positive integer k, $[k]$ denotes the set of integers $\{1, \ldots, k\}$. Concatenation of two binary strings x and y is denoted by $x||y$.

3.1 TPPKS

In this section, we introduce the notion of the TPPKS scheme. Basically, a TPPKS scheme consists of five algorithms defined as follows.

System Instantiation: The manager takes a security parameter and generates the system public parameters and the secret keys.

Key Distribution: The manager selects secret keys at random, creates shares for the secret key and distributes the shares among the members. Shares are used to generate the trapdoor for a list of keywords and broadcasts verification keys.

Data Encryption and Secure Index Generation: Each user applies the system public parameters in order to encrypt her data, to generate the corresponding secure index, and to upload them to the server.

Trapdoor Generation and Data Search: First each one of the collaborating members generates a share of a trapdoor for the target keywords. Then the collaborating members verify their shares. After all shares have passed the verification, they are combined to make the trapdoor for the target keywords.

Data Decryption: On receiving the encrypted data from the server, each collaborating member uses her secret share to generate a decryption share of encrypted data. If all decryption shares are valid, the collaborating members can compute the plaintext.

3.2 Security Model

A TPPKS scheme usually has to provide data privacy and privacy of the secret key and shares. Data privacy means that, all queries, secure indices and encrypted data leak no information about the plaintexts. Privacy of secret key and shares means that, a group of members of size less than the threshold is not able to find the secret key, and the use of a share discloses no information about the share. To satisfy these privacy requirements, we need to make some specific assumptions about the security of the components and building blocks used in our protocols.

To prove the security of search protocol, we use the Golle *et al.*'s security game [7] namely, the indistinguishability of ciphertext from limited random string (ICLR). It captures a secure notion – Semantic Security against Adaptive Chosen Keyword Attacks (IND-CKA) [6], which guarantees that an adversary cannot recover the contents of a document from its secure index and the indices of other documents.

Since secure indices and trapdoors only concern the keyword list L in a data, and not the data, for convenience, we denote the secure index of the data by

I_L. We assume that the number of keywords associated with a document is m. Let $\mathrm{Rand}(L, V)$ denote a randomized keyword list formed from the keyword list $L = (w_1, \ldots, w_m)$ by replacing the keywords of L that are indexed by a subset $V \subset \{1, \ldots, m\}$ by random values.

Definition 1. *A trapdoor T is distinguishable for keyword lists L_i and L_j if the result of searching with T on the secure index of L_i is different from that of L_j.*

Definition 2. ICLR *is a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

Setup. *$\mathcal{A}$ adaptively selects a polynomial number of keyword lists, $\{L^*\}$, and requests the secure indices, $\{I_{L^*}\}$, from $\mathcal{C}$.*

Queries. *$\mathcal{A}$ may query $\mathcal{C}$ to get the trapdoor $T_{L'}$ of a keyword list L'. With $T_{L'}$, $\mathcal{A}$ can search on a secure index I_{L^*} to determine if all keywords in the list L' are contained in L^* or not.*

Challenge. *After making a polynomial number of queries, $\mathcal{A}$ decides on challenge by picking a keyword lists L, a subset $V \subset [m]$ and a value $v \in V$ such that $\mathcal{A}$ must not have asked for any trapdoor distinguishing $Rand(L, V)$ from $Rand(L, V - \{v\})$, and sending them to $\mathcal{C}$. Then $\mathcal{C}$ chooses $b \xleftarrow{R} \{0, 1\}$. If $b = 0$, $\mathcal{A}$ is given I_0 of $L_0 = Rand(L, V - \{v\})$. If $b = 1$, $\mathcal{A}$ is given I_1 of $L_1 = Rand(L, V)$. After the challenge of determining b for $\mathcal{A}$ is issued, $\mathcal{A}$ is allowed again to query $\mathcal{C}$ with the restriction that $\mathcal{A}$ may not ask for the trapdoor that distinguishes L_0 from L_1.*

Response. *Eventually $\mathcal{A}$ outputs a bit $b_{\mathcal{A}}$, and is successful if $b_{\mathcal{A}} = b$. The advantage of $\mathcal{A}$ in winning this game is defined as $Adv_{\mathcal{A}} = |Pr[b = b_{\mathcal{A}}] - 1/2|$, and the adversary is said to have an ϵ-advantage if $Adv_{\mathcal{A}} > \epsilon$.*

3.3 Complexity Assumptions

In this section, we briefly review three well-known hardness assumptions, which are Discrete Logarithm (DL), Decisional Diffie-Hellman (DDH) and Computational Diffie-Hellman (CDH).

Definition 3 (DL Assumption). *Given a finite cyclic group $G = \langle g \rangle$ of prime order q with a generator g. For a given random number $x \in G$, the DL problem is to find an integer t $(0 \leq t < q)$ such that $x = g^t$. An algorithm $\mathcal{A}$ is said to have an ϵ-advantage in solving the DL problem if*

$$Pr[\mathcal{A}(g, g^t) = t] > \epsilon.$$

The DL assumption holds in G if no PPT algorithm has advantage at least ϵ in solving the DL problem in G.

Definition 4 (DDH Assumption). *Let $G = \langle g \rangle$ be a cyclic group of prime order q and g a generator of G. The DDH problem is to distinguish between triplets of the form (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where $a, b, c \xleftarrow{R} Z_q$. An algorithm $\mathcal{A}$ is said to have an ϵ-advantage in solving the DDH problem if*

$$|Pr[\mathcal{A}(g^a, g^b, g^{ab}) = yes] - Pr[\mathcal{A}(g^a, g^b, g^c) = yes]| > \epsilon.$$

The DDH assumption holds in G if no PPT algorithm has advantage at least ϵ in solving the DDH problem in G.

Definition 5 (CDH Assumption). *Let $G = \langle g \rangle$ be a cyclic group of prime order q and g a generator of G. The CDH problem is to compute g^{ab} for given $g, g^a, g^b \in G$, where $a, b \xleftarrow{R} Z_q$. An algorithm $\mathcal{A}$ is said to have an ϵ-advantage in solving the CDH problem if*

$$Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}] > \epsilon.$$

The CDH assumption holds in G if no PPT algorithm has advantage at least ϵ in solving the CDH problem in G.

3.4 The Bilinear Pairings

Let G_1, G_2 be two cyclic groups of some large prime order q. A bilinear pairing is defined as a function $e : G_1 \times G_1 \rightarrow G_2$ with the following properties:

1. Bilinear: for all $P, Q \in G_1$ and $a, b \in Z_q$, $e(aP, bQ) = e(P, Q)^{ab}$.
2. Non-degenerate: there exist $P, Q \in G_1$ such that $e(P, Q) \neq 1$, where 1 is the identity of G_2.
3. Computable: for all $P, Q \in G_1$, $e(P, Q)$ is computable in polynomial time.

A Bilinear Pairing Parameter Generator is defined as a polynomial-time algorithm $\mathcal{BPPG}$, which takes as input a security parameter k and outputs a uniformly random tuple (e, G_1, G_2, q) of bilinear pairing parameters.

4 Construction of TPPKS

4.1 System Instantiation Algorithm

1. The manager C runs a $\mathcal{BPPG}$ with a security parameter k to generate bilinear pairing parameters (q, G_1, G_2, e), where G_1 is an additive group of large prime order q with a generator P, $q' = \frac{q-1}{2}$ is also a prime, G_2 is a multiplicative group of order q and the DL and CDH assumptions hold in both G_1 and G_2.
2. C chooses two cyclic groups: a multiplicative group G of prime order q with a generator g, in which the DDH assumption holds, and an additive group $G_0 = \langle P_0 \rangle$ of prime order q', in which the computation is based on the modulus q and the DL assumption holds.
3. C chooses three cryptographic hash functions:

$$H : \{0,1\}^* \rightarrow Z_q^*, H_1 : \{0,1\}^* \rightarrow G_1, \text{ and } H_2 : G_2 \rightarrow \{0,1\}^l,$$

   where $\{0,1\}^l$ is the plaintext space.
4. C chooses $Q \xleftarrow{R} G_1$, and five different values $\lambda, \sigma, r, d, s \xleftarrow{R} Z_q^* \setminus \{1\}$ and computes $P' = \lambda P, Q' = (\lambda - \sigma)Q, g' = g^{\frac{r}{d}}, \tilde{g} = g'^s$ and $u = \frac{s}{d}$.
5. C publishes system's public parameters $\{e, G, G_0, G_1, G_2, q, q', g, g', \tilde{g}, u, P_0, P, P', Q, Q', H, H_1, H_2\}$ and keeps $\{\lambda, \sigma, r, d, s\}$ secret.

4.2 Key Distribution Algorithm

1. Every member M_i $(1 \leq i \leq n)$ has an unique identity number ID_i, and C computes $x_i = H(ID_i)$ $(i = 1, \ldots, n)$.
2. C randomly generates three secret polynomials f_0, f_1, f_2 of degree $t-1$ of the form
$$f_0(x) = r + a_1^{(0)}x + \cdots + a_{t-1}^{(0)}x^{t-1},$$
$$f_1(x) = d + a_1^{(1)}x + \cdots + a_{t-1}^{(1)}x^{t-1},$$
$$f_2(x) = \sigma + a_1^{(2)}x + \cdots + a_{t-1}^{(2)}x^{t-1},$$
where $\{a_j^{(i)}\}$ $(i = 0, 1, 2; j = 1, \cdots, t-1)$ are secretly random numbers in Z_q^*.
3. For every member M_i, C lets $r_i = f_0(x_i), d_i = f_1(x_i), \sigma_i = f_2(x_i)$, and delivers the secret shares r_i, d_i, σ_i to M_i $(1 \leq i \leq n)$ via a secure channel.
4. C computes
$$\nu_i^{(r)} = r_i H_1(ID_i), \nu_i^{(d)} = d_i H_1(ID_i), \text{ and } \nu_i^{(\sigma)} = e(\sigma_i H_1(ID_i), P),$$
and publishes $(\nu_i^{(r)}, \nu_i^{(d)}, \nu_i^{(\sigma)})$ as the verification keys of M_i $(1 \leq i \leq n)$.

4.3 Data Encryption and Secure Index Generation Algorithm

1. A user encrypts her data $\mathcal{M}$ as follows: chooses $\gamma \xleftarrow{R} Z_q^* \setminus \{1\}$, computes
$$X = \gamma P \text{ and}, Y = \mathcal{M} \oplus H_2(e(Q, P')^\gamma),$$
and let $R = (X, Y)$ be the ciphertext of $\mathcal{M}$.
2. The user chooses $\alpha \xleftarrow{R} Z_q^* \setminus \{1\}$ and computes $W' = g^{-\alpha}$ and $\bar{W} = g'^{\alpha}$. For each keyword w_j in $\mathcal{M}$, the user computes $W_j = \tilde{g}^{\alpha H(w_j) P_0}$.
3. The user lets $I = \{W', \bar{W}, W_1, W_2, \ldots, W_m\}$ be the secure index of the data $\mathcal{M}$, and uploads $\{I, R\}$ to the server.

4.4 Trapdoor Generation and Data Search Algorithm

1. The t members $\{M_{i_j}\}_{j=1,\cdots,t}$ with identities $\{ID_{i_i}\}_{j=1,\cdots,t}$ together compute
$$c_{i_j} = \prod_{m'=1, m' \neq j}^{t} \frac{H(ID_{i_{m'}})}{H(ID_{i_{m'}}) - H(ID_{i_j})},$$
choose $\beta \xleftarrow{R} Z_q^* \setminus \{1\}$, and compute $A^{(0)} = \beta P_0$ in G_0.

   For every queried keyword $w'_{m'}$ in the queried keyword list $L' = \{w'_{m'}\}_{m'=1,\ldots,l}$ $(l \leq m)$, they compute $A^{(m')} = (uH(w'_{m'}) + \beta)P_0$ in G_0.

2. Each member M_{i_j} computes in G_0

$$A_{i_j}^{(0)} = c_{i_j} r_{i_j} A^{(0)} \text{ and } A_{i_j}^{(m')} = c_{i_j} d_{i_j} A^{(m')} \ (m' = 1, \cdots, l),$$

and takes them as her share of the trapdoor of L'.

3. The t members verify every member's shares by checking whether it holds for $j = 1, \cdots, t$ that

$$\begin{aligned} e(H_1(ID_{i_j}), A_{i_j}^{(0)} P) &= e(c_{i_j} \nu_{i_j}^{(r)}, A^{(0)} P) \text{ and} \\ e(H_1(ID_{i_j}), A_{i_j}^{(m')} P) &= e(c_{i_j} \nu_{i_j}^{(d)}, A^{(m')} P) \ (m' = 1, \cdots, l). \end{aligned}$$

If it holds for all $j = 1, \cdots, t$, this means that all search shares are valid, then they go to next step. If it does not hold for some j, this means that M_{i_j} provides a invalid search share, then they terminate the protocol.

4. The t members compute

$$A_0 = \sum_{j=1}^{t} A_{i_j}^{(0)} \text{ and } A_{m'} = \sum_{j=1}^{t} A_{i_j}^{(m')} \ (m' = 1, \cdots, l),$$

and sends $(A_0, \{A_{m'}\}_{m'=1}^{l})$ as the trapdoor of L' to the server.

5. On receiving the trapdoor, the server tests on a secure index for every $m' \in [l]$ if there exists some $i \in [m]$ such that

$$W'^{A_0} \cdot \bar{W}^{A_{m'}} = W_i.$$

If so, the server puts the data R in a collection. After all secure indices are checked, if the collection is not empty, the server returns the collection to the member; otherwise, returns No Data Matched to the t members.

4.5 Data Decryption Algorithm

When the t members receive a data $R = (X, Y)$, they do the following.

1. Each member M_{i_j} $(j = 1, \cdots, t)$ chooses $y \xleftarrow{R} Z_q^* \setminus \{1\}$, computes

$$\begin{gathered} y_{i_j}^{(1)} = e(Q, X)^y, y_{i_j}^{(2)} = e(H_1(ID_{i_j}), P)^y, \tilde{p}_{i_j} = H(y_{i_j}^{(1)} || y_{i_j}^{(2)}), \\ z_{i_j} = \tilde{p}_{i_j} \sigma_{i_j} + y \text{ and } v_{i_j} = e(Q, X)^{\sigma_{i_j}}, \end{gathered}$$

and provides $\{y_{i_j}^{(1)}, y_{i_j}^{(2)}, z_{i_j}, v_{i_j}\}$ as her decryption share.

2. The t members compute $\tilde{p}_{i_j} = H(y_{i_j}^{(1)} || y_{i_j}^{(2)})$ $(j = 1, \cdots, t)$ and check whether it holds that

$$e(Q, X)^{z_{i_j}} = v_{i_j}^{\tilde{p}_{i_j}} y_{i_j}^{(1)} \text{ and } e(H_1(ID_{i_j}), P)^{z_{i_j}} = (\nu_{i_j}^{(\sigma)})^{\tilde{p}_{i_j}} y_{i_j}^{(2)}.$$

If it holds for all $j = 1, \cdots, t$, this means that all decryption shares are valid, then they go to next step. If it does not hold for some j, this means that M_{i_j} provides a invalid decryption share, then they terminate the protocol.

3. Finally, the t members compute $D_{i_j} = v_{i_j}^{c_{i_j}}$ $(j = 1, \cdots, t)$, and then output the plaintext

$$\mathcal{M} = Y \oplus H_2(\prod_{j=1}^{t} D_{i_j} \cdot e(Q', X)).$$

Note. Although we use intersection operations to do a conjunctive keyword search in the above scheme, that is, a secure index is tested for one keyword at first, and then, upon the test result, the server decides if it keeps testing for the next keyword. In fact, our scheme can also deal with conjunctive keyword searches in the same way as all existing conjunctive keyword search schemes do, that is, secure indices use keyword fields, and the positions where the conjunctive keywords appear in secure indices are given in a trapdoor. The details are as follows. A member has an above trapdoor $(A_0, \{A_{i_j}\}_{j=1}^{l})$ for a list of keywords $L = \{w_{i_j}\}_{j=1}^{l}$, where i_j is the position where the keyword w_{i_j} appears in the secure index, this means, $\{i_j\}_{j=1}^{l} \subset [m]$. The member computes $A_0^{(c)} = (A_0)^l, A_1^{(c)} = \sum_{j=1}^{l} A_{i_j}$ and sends $\{A_0^{(c)}, A_1^{(c)}, i_1, \cdots, i_l\}$ as the trapdoor of L to the server. The sever checks if it holds that $W'^{A_0^{(c)}} \cdot \bar{W}^{A_1^{(c)}} = \prod_{j=1}^{l} W_{i_j}$ to guess whether all the keywords $\{w_{i_j}\}_{j=1}^{l}$ are in the secure index or not.

5 Security

The following four theorems state the security of the proposed TPPKS scheme.

Theorem 1. *The secret key distribution process in the proposed TPPKS is secure against impersonation and a coalition of up to* $(t-1)$ *adversaries.*

Proof. First we prove two claims.

Claim 1. *Anyone excluding the manager and* t *collaborating members cannot recover the secret keys* r, d, σ *in the key distribution process. The security is unconditional.*

Proof. Let's consider the case of the secret key r firstly, and the cases of d, σ can be discussed in the same way.

We use the Shamir (t, n)-threshold Secret Sharing scheme in the construction of secret key distribution in a straightforward way, that is, the manager selects $t-1$ numbers $(a_1, \ldots, a_{t-1})$ at random along with the secret key r and constructs the polynomial f_0 of degree $t-1$, so

$$f_0(x) = r + a_1^{(0)} x + \cdots + a_{t-1}^{(0)} x^{t-1}.$$

Each member is given the private key s_i over a secure channel, so any PPT adversary excluding the manager and t collaborating members cannot recover the coefficients of the polynomial $f_0(x)$, that means, he cannot compute $r = f_0(0)$.

Next, we consider the security of r when it is used to generate a trapdoor. An adversary can get a polynomial sample of pairs (P, rP) of G_0 by observing the input and output of the trapdoor generation algorithm. However, the security of r is still kept based on the DL assumption.

Hence, it is computationally impossible for any adversary excluding the manager and t collaborating members to reveal the secret key r in the key distribution process.

Claim 2. *Any PPT adversary excluding the manager and t collaborating members cannot computationally forge a member's secret shares r_i, d_i, σ_i in the key distribution process.*

Proof. As in **Claim 1**, we only take r as an example to discuss.

From the **Claim 1**, we know that no body (excluding the manager and t collaborating members) can construct $f_0(x)$. So, it is computationally impossible to compute r_i from $f_0(x)$. When r_i is used to generate a share of a trapdoor, an adversary can get a polynomial of element pairs $(P, r_i P)$ of G_0 by observing the input and output of M_i. Because DL assumption holds in G_0, no PPT adversary excluding the manager and t collaborating members can compute r_i. So, we have the claim.

From the **Claim 1** and **Claim 2**, we have the theorem immediately.

Theorem 2. *The secret share verification algorithms used in the proposed TPPKS are secure under the DL and CDH assumptions.*

Proof. First we check correctness of the verification algorithm for the secret share r_i.

$$e(H_1(ID_i), A_i^{(0)} P) = e(H_1(ID_i), A^{(0)} P)^{c_i r_i} = e(c_i \nu_i^{(r)}, A^{(0)} P),$$

this means, if the member M_i is honest, the verification algorithm gives a positive answer. If M_i replaces r_i with a forged share $\tilde{r}_i$, then we see that

$$e(H_1(ID_i), A_i^{(0)} P) = e(H_1(ID_i), A^{(0)} P)^{c_i \tilde{r}_i} \neq e(H_1(ID_i), A^{(0)} P)^{c_i r_i} = e(c_i \nu_i^{(r)}, A^{(0)} P),$$

the verification algorithm gives a negative answer.

Because the manager publishes every member's verification key, so an adversary can easily get $(H_1(ID_i), r_i H_1(ID_i))$. As DL assumption holds in G_1, so he cannot compute r_i in PPT. If an adversary can get the r_i from the verification algorithm, we can interact with the adversary in the same way as in [12] (Theorem 1: Case 1: Non-interaction with Signer) to break the CDH assumption.

The case of d_i is similar to the case of r_i, we omit the discussion.

Now let's analyze the case of σ_i.

Since the verification key $\nu_i^{(\sigma)}$ is published, an adversary can get a pair of elements in G_2 $(e(H_1(ID_i), P), e(\sigma_i H_1(ID_i), P))$. Similarly, σ_i is secure under the DL assumption. We use the non-interactive zero-knowledge protocol of Fiat and Shamir [5] to construct the verification algorithm for σ_i, so the verification algorithm is secure.

Theorem 3. *The data cryptosystem used in the proposed TPPKS is semantically secure.*

Because the cryptosystem used in our scheme follows from the ID-based cryptosystem of Boneh and Franklin [2] and its variation [8] in a straightforward way, we have this theorem immediately.

Theorem 4. *The search process in the proposed TPPKS is semantically secure under the DDH assumption according to the security game ICLR.*

Proof. To describe conveniently, we omit all information concerning the threshold setting, and only concern the keyword search process, so the member's share (r_i, d_i) and P_0 don't occur in our proof.

Suppose that the scheme is not semantically secure under the security game ICLR. Then there exists an adversary $\mathcal{A}$ that wins the ICLR game with an ϵ-advantage. We build an adversary $\mathcal{A}'$ that uses $\mathcal{A}$ as a subroutine and breaks the DDH assumption with the $\frac{\epsilon}{2m}$-advantage.

Let (g^a, g^b, g^c) be $\mathcal{A}'$'s DDH challenge. $\mathcal{A}'$'s goal is to break the DDH assumption, or in other words to decide whether $c = ab$. $\mathcal{A}'$ guesses a value z for the position v that $\mathcal{A}$ will choose in the phase **Challenge** of the game ICLR, by picking $z \xleftarrow{R} [m]$. Then $\mathcal{A}'$ works by interacting with $\mathcal{A}$ in the ICLR game as follows:

Setup: $\mathcal{A}$ makes a polynomial number of requests for secure indices, which $\mathcal{A}'$ answers as follows. Let one of $\mathcal{A}$'s keyword lists be $L^* = (w_1^*, \ldots, w_m^*)$. $\mathcal{A}'$ chooses $r, d, s, \alpha \xleftarrow{R} Z_q^*$ and lets $g' = g^{\frac{r}{d}}, \tilde{g} = g'^s = g^{\frac{rs}{d}}$ and $u = \frac{s}{d}$, and computes $W' = g^{-\alpha}$ and $\bar{W} = g'^{\alpha}$. For every word $w_j^* \in L^*$ $(1 \le j \le m)$, $\mathcal{A}'$ picks a value $x_j^* \xleftarrow{R} Z_q^*$ and associates it with w_j^*. Then $\mathcal{A}'$ computes $W_j = \tilde{g}^{\alpha x_j^*}$ $(j = 1, \ldots, z-1, z+1, \ldots, m)$ and $W_z = (g^b)^{\frac{rs}{d}\alpha x_z^*}$. Note that $\mathcal{A}'$ is given g^b as part of the DDH challenge, so she can compute all the values. To be consistent across different queries, $\mathcal{A}'$ keeps track of the corresponding pair (w_j^*, x_j^*). Finally, it returns the secure index $I_{L^*} = (W', \bar{W}, W_1, \ldots, W_m)$ to $\mathcal{A}$.

Queries: $\mathcal{A}$ queries for a keyword list $L' = (w_1', \ldots, w_l')$ $(1 \le l \le m)$ with the restriction that she may not make a query that are distinguishing $\mathrm{Rand}(L, V)$ from $\mathrm{Rand}(L, V - \{v\})$. $\mathcal{A}'$ chooses $\beta \xleftarrow{R} Z_q^*$, computes $A_0 = r\beta$ and $A_j = sx_j' + d\beta$ for each word $w_j' \in L'$, where x_j' takes the previously used value if the keyword w_j' previously appeared in any one of queried trapdoors or secure indices, or $x_j' \xleftarrow{R} Z_q^*$ otherwise (also the corresponding pair (w_j', x_j') has to be kept in memory for future use). Finally, $\mathcal{A}'$ returns the trapdoor $T_{L'} = \{A_0, A_1, \ldots, A_l\}$ to $\mathcal{A}$. Because $\mathcal{A}'$ consistently uses the same value for the same keyword, $T_{L'}$ is a valid trapdoor for L'. By searching on the secure indices with $T_{L'}$, $\mathcal{A}$ can get the right result.

Challenge: After making polynomially many index and trapdoor queries, $\mathcal{A}$ decides on a challenge by submitting the challenge keyword list $L = (w_1, \ldots, w_m)$ along with a subset $V \subset \{1, \ldots, m\}$ and a value $v \in V$. If $z \neq v$, $\mathcal{A}'$ returns a random value in reply to the DDH challenge. With probability $1/m$, we have $z = v$ and in that case $\mathcal{A}'$ proceeds as follows. Let $W_v = (g^c)^{\frac{rs}{d}\alpha x_v}$. For $j \in V$ and $j \neq v$, let $W_j = R_j$ for a random value R_j. For $j \notin V$, let $W_j = (g^a)^{\frac{rs}{d}\alpha x_j}$. Where, x_j $(1 \le j \le m)$ takes a value in the same way as in **Queries**. $\mathcal{A}'$ computes $W' = (g^a)^{-\alpha}$ and $\bar{W} = (g^a)^{\frac{r}{d}\alpha}$, and returns $I = (W', \bar{W}, W_1, \ldots, W_m)$ to $\mathcal{A}$. Observe that, although $\mathcal{A}'$ does not know

a, c, she can compute all the values, since g^a, g^c is given as part of the DDH challenge. Check that this index is an encryption of keyword in every position $j \notin V$. If $c = ab$, this index is also an encryption of keyword w in position v; otherwise it is not. Now $\mathcal{A}$ is again allowed to ask for queries with the restriction that $\mathcal{A}$ may not make a query that are distinguishing $\text{Rand}(L, V)$ from $\text{Rand}(L, V - \{v\})$.

Response: Finally, $\mathcal{A}$ outputs a bit $b_{\mathcal{A}}$. If $b_{\mathcal{A}} = 0$, $\mathcal{A}'$ guesses that $\{g^a, g^b, g^c\}$ is not a DDH triplet. If $b_{\mathcal{A}} = 1$, $\mathcal{A}'$ guesses that $\{g^a, g^b, g^c\}$ is a DDH triplet.

Let $\mathsf{triplet}$ be the event that $\{g^a, g^b, g^c\}$ is a DDH triplet. From the definition of DDH we have

$$Pr(\mathsf{triplet}) = Pr(\overline{\mathsf{triplet}}) = \frac{1}{2}.$$

Let $\mathsf{succ}_{\mathcal{A}'}$ and $\mathsf{succ}_{\mathcal{A}}$ be the events the $\mathcal{A}'$ and $\mathcal{A}$ win their respective games. Because $\mathcal{A}'$ returns a random value in reply to the DDH challenge when $z \neq v$, the output must be independent of b, we have

$$Pr(\mathsf{succ}_{\mathcal{A}'} | z \neq v) = \frac{1}{2}.$$

When $z = v$, in the case that the event $\mathsf{triplet}$ occurs, $\mathcal{A}'$ solves the DDH challenge with the same advantage that $\mathcal{A}$ has in winning game ICLR. So we have

$$Pr(\mathsf{succ}_{\mathcal{A}'} | z = v | \mathsf{triplet}) = Pr(\mathsf{succ}_{\mathcal{A}}).$$

When $z = v$, in the case that the event $\overline{\mathsf{triplet}}$ occurs, that is, the input for $\mathcal{A}$ to guess b is a uniformly random element in G. So, the output is independent of b, we have

$$Pr(\mathsf{succ}_{\mathcal{A}'} | z = v | \overline{\mathsf{triplet}}) = \frac{1}{2}.$$

As the event $z = v$ is independent of the event $\mathsf{triplet}$ or $\overline{\mathsf{triplet}}$, so we have

$$Pr(z = v | \mathsf{triplet}) = Pr(z = v | \overline{\mathsf{triplet}}) = \frac{1}{m}.$$

We know that $Pr(z \neq v) = \frac{m-1}{m}$. Putting them together, we have

$$\begin{aligned} Pr(\mathsf{succ}_{\mathcal{A}'}) &= Pr(\mathsf{succ}_{\mathcal{A}'} | z \neq v) Pr(z \neq v) \\ &\quad + Pr(\mathsf{succ}_{\mathcal{A}'} | z = v | \mathsf{triplet}) Pr(z = v | \mathsf{triplet}) Pr(\mathsf{triplet}) \\ &\quad + Pr(\mathsf{succ}_{\mathcal{A}'} | z = v | \overline{\mathsf{triplet}}) Pr(z = v | \overline{\mathsf{triplet}}) Pr(\overline{\mathsf{triplet}}) \\ &= \tfrac{1}{2} \cdot \tfrac{m-1}{m} + Pr(\mathsf{succ}_{\mathcal{A}}) \cdot \tfrac{1}{m} \cdot \tfrac{1}{2} + \tfrac{1}{2} \cdot \tfrac{1}{m} \cdot \tfrac{1}{2} \\ &= \tfrac{2m-1}{4m} + \tfrac{1}{2m} Pr(\mathsf{succ}_{\mathcal{A}}) \end{aligned}$$

Therefore, the advantage of $\mathcal{A}'$ in solving the DDH problem is

$$\begin{aligned} Pr(\mathsf{succ}_{\mathcal{A}'}) - \tfrac{1}{2} &= \tfrac{2m-1}{4m} + \tfrac{1}{2m} Pr(\mathsf{succ}_{\mathcal{A}}) - \tfrac{1}{2} \\ &= \tfrac{1}{2m} Pr(\mathsf{succ}_{\mathcal{A}}) - \tfrac{1}{4m} \\ &= \tfrac{1}{2m} (Pr(\mathsf{succ}_{\mathcal{A}}) - \tfrac{1}{2}) \\ &> \tfrac{\epsilon}{2m} \end{aligned}$$

6 Conclusion and Future Research

We present a definition of threshold privacy preserving keyword searches, called TPPKS, and described its security requirements. We constructed an efficient TPPKS scheme and proved its security. However, in the proposed scheme, the members are fixed, maybe some want to leave or some new members want to join in some cases, so designing the scheme for a dynamic group is still a challenging problem.

Acknowledgments

The work was in part supported by Australian Research Council Discovery grants DP0663452, DP0558773 and DP0665035. Huaxiong Wang's research was in part supported by Singapore Ministry of Education grant T206B2204.

References

1. Boneh, D., Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
2. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
3. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
4. Byun, J.W., Lee, D.H., Lim, J.: Efficient Conjunctive Keyword Search on Encrypted Data Storage System. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 184–196. Springer, Heidelberg (2006)
5. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solution to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–189. Springer, Heidelberg (1987)
6. Goh, E.-J.: Secure indexes, in *Cryptology ePrint Archive*, Report 2003/216 (February 25, 2004), http://eprint.iacr.org/2003/216/
7. Golle, P., Staddon, J., Waters, B.: Secure Conjunctive Search over Encrypted Data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
8. Liu, S., Chen, K., Qiu, W.: Identity-Based Threshold Decryption Revisited. In: ISPEC 2007. LNCS, vol. 4464, pp. 329–343. Springer, Heidelberg (2007)
9. Park, H.A., Byun, J.W., Lee, D.H.: Secure Index Search for Groups. In: Katsikas, S.K., Lopez, J., Pernul, G. (eds.) TrustBus 2005. LNCS, vol. 3592, pp. 128–140. Springer, Heidelberg (2005)
10. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 44–55 (May 2000)
11. Wang, P., Wang, H., Pieprzyk, J.: Common Secure Index for Conjunctive Keyword-Based Retrieval over Encrypted Data. In: SDM 2007. LNCS, vol. 4721, pp. 108–123. Springer, Heidelberg (2007)
12. Zhang, F., Kim, K.: ID-Based Blind Signature and Ring Signature from Pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002)

3D_XML: A Three-Dimensional XML-Based Model

Khadija Ali[1] and Jaroslav Pokorný[2]

[1] Czech Technical University, Faculty of Electrical Engineering, Praha, Czech Republic
alik1@fel.cvut.cz
[2] Charles University, Faculty of Mathematics and Physics, Praha, Czech Republic
jaroslav.pokorny@mff.cuni.cz

Abstract. Much research work has recently focused on the problem of representing historical information in XML. In this paper, we describe an ongoing work to represent XML changes. Our model is a three-dimensional XML-based model (3D_XML in short) for representing and querying histories of XML documents. The proposed model incorporates three time dimensions, *valid time*, *transaction time*, and *efficacy time* without extending the syntax of XML. We use XQuery to express complex temporal queries on the evolution of the document contents. We believe that native XML databases (NXDs) present a viable alternative to relational temporal databases when complex time dependent data has to be manipulated and stored. So NXDs will be our choice.

Keywords: XML, 3D_XML model, transaction time, valid time, efficacy time, XQuery, three-dimensional element, native XML databases.

1 Introduction

Recently, the amount of data available in XML [1] has been rapidly increasing. Much research work has recently focused on adding temporal features to XML [2, 3, 5, 6, 7, 11]. Temporal information is supported in XML much better than in relational tables. This property is attributed to the hierarchical structure of XML which is perfectly compatible with the structure of temporal data. Recently, only few works capture the notion of time explicitly in this context. Technically, to develop an XML temporal data model, it is necessary to extend a XML data model by a time dimension. The problem is that there is more XML data models (e.g. Infoset, XPath data model, XQuery data model, etc.) and more times (usually valid and transaction times). The main aim of this paper is to propose a temporal extension of XML. We propose a new scheme to represent XML changes, and show how temporal queries can be supported on this scheme.

An important issue of each data model is its implementation. There are two different ways to store XML documents in a database: *XML-enabled databases* and *native XML databases* (NXDs) [10]. The former map the data to existing (relational) database systems. The latter are XML-database systems whose inner data representation is XML-compliant. (NXDs) preserve data hierarchy and meaning of XML documents. So

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 659–671, 2008.

(NXDs) will be our choice (particularly *eXist* [12]). In the following points we summarize the motivation of this choice:

1. eXist is open-source, and free to use. It uses the numbering scheme which supports quick identification of relationships between nodes as well as navigation through the document tree.
2. It is schema independent. It is also very user friendly. It has been chosen best XML database for InfoWorld's 2006 Technology of the Year awards. It's a worthwhile open source project for people who are interested in programming, since it's still incomplete.

The paper is organized as follows. After a discussion of related work in the next section, in Section 3 we define formally a new model (3D_XML). In Section 4, we deal with current-time (now), and show how it is supported in 3D_XML. We describe the temporal constructs of 3D_XML in Section 5. In Section 6 we illustrate that XQuery is capable of expressing complex temporal queries, but the expression of these queries can be greatly simplified by a suitable library of built-in temporal functions. Finally, in Section 7, we present our conclusions and future investigations.

2 Related Work

In the following subsections we provide a comparison of some works which have made important contributions in providing expressive and efficient means to model, store, and query XML-based temporal data models [2, 3, 5, 6, 7, 11] according to the following properties: time dimension (valid time, transaction time), support of temporal elements and attributes, querying possibilities, association to XML Schema/DTD, and influence on XML syntax [9].

Time dimension. All the models are capable to represent changes in an XML document by supporting temporal elements, and incorporating time dimensions. Two time dimensions are usually considered: valid time and transaction time. There are several other temporal dimensions that have been also mentioned in the literature in relation to XML. In [7] a publication time and efficiency time in the context of legal documents are proposed.

Temporal elements and attributes. Time dimensions may be applied to elements and attributes. All the models are capable to support temporal elements. In [3] and [11] the temporal attributes are supported. In [3] versions of an element are explicitly associated as being facets of the same (multidimensional) element. Grouping facets together allows the formulation of cross-world queries, which relate facets that hold under different worlds [14].

Influence on XML syntax. Only in [3] the syntax of XML is extended in order to incorporate not only time dimensions but also other dimensions such as language, degree of detail, etc. So the approach in [3] is more general than other approaches as it allows the treatment of multiple dimensions in a uniform manner.

Querying possibilities. The model's power depends also on supporting powerful temporal queries. In [5] and [11] powerful temporal queries expressed in XQuery without extending the language are supported. In [6] a valid time support is added to

XPath. This support results in an extended data model and query language. In [7] querying uses combination of full text retrieval and XQuery extended by some constructs to deal with time dimensions. The other models in [2] and [3] did not discuss the issue of temporal queries; in [2] elements have timestamps if they are different from the parent nodes. This fact complicates the task of writing queries in XPath/XQuery.

Association to XML Schema/DTD. A significant advantage will be added to the model if it is not only representing the history of an XML document but also the history of its corresponding XML schema or DTD as well. In [3], [7], and [11] the temporal XML schema/DTD is supported by extending the existing XML schema/DTD.

3 3D_XML Formalism

We shortly introduce three time dimensions in Section 3.1 as they are usually used in temporal databases. Then in Section 3.2 and Section 3.3, we describe our time and data models.

3.1 Time Dimensions

Three temporal dimensions are considered; valid time, transaction time, and efficacy time.

- *Valid time*: the valid time of the fact is the time when the fact is valid, or true in the modeled reality.
- *Transaction time*: it concerns the time the fact was present in the database as stored data. In other words, the transaction time of the fact identifies the time when the fact is inserted into the database and the time when that fact is removed from the database.
- *Efficacy time*: it usually corresponds to the valid time, but it can be a case that an abrogated data continues to be applicable to a limited number of cases. Until such cases cease to exit, the data continues its efficacy [7].

We extend the above efficacy time definition by assuming that it can be a case that valid data stops to be applicable to a limited number of cases. When such cases cease to exit, the data stops its efficacy.

Example 1: Consider a company database. Suppose the manager `mgr` of `Design` department is `Esra` from `"2002-01-01"` till `"2006-09-25".` Due to some unexpected circumstances, another person started managing `Design` department from `"2002-09-08".` `Esra` stopped managing `Design` department from `"2002-09-08"` till `"2006-09-25";` this case represents when valid data stops to be applicable. Suppose the efficacy time start is the same as the valid time start. The element `mgr` is timestamped by `"2002-01-01", "2006-09-25", "2002-01-01",` and `"2002-09-07"` which represent valid time start, valid time end, efficacy time start, efficacy time end, respectively. In this example, the valid time end `"2006-09-25"` is greater than efficacy time end `"2002-09-07".` Figure 1 represents the valid time interval when `Esra` is a manager of `Design` department, its efficacy time interval, while the last part represents the time interval when `Esra` stopped managing `Design` department. Figure 2 depicts valid and efficacy times

relationship. Valid time is represented by a time interval `(vtStart, vtEnd)`. Efficacy time is represented by a time interval `(etStart, etEnd)`. The relationship between valid time and efficacy time falls into three categories:

1. `(vtStart < etStart)` or `(vtEnd > etEnd)`; it represents a case valid data stops to be applicable.
2. `(vtStart > etStart)` or `(vtEnd < etEnd)`; it represents a case data is applicable although it is not valid.
3. `(vtStart = etStart)`, and `(vtEnd = etEnd)`; it represents the normal case.

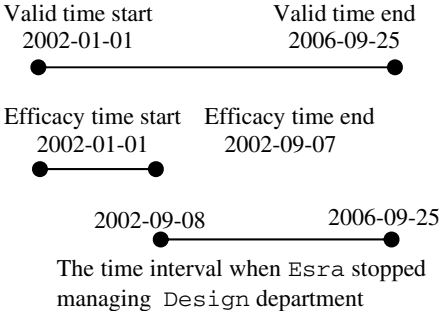


Fig. 1. Valid and efficacy times of `mgr` in Example 1

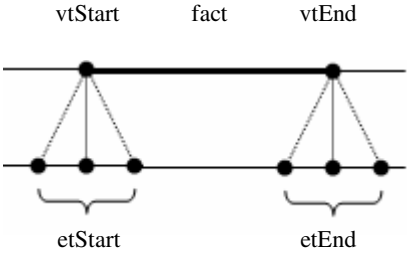


Fig. 2. Valid and efficacy times relationship

3.2 Time Model

In order to represent the changes in an XML document we encode this document as a 3D_XML document in which the syntax of XML is not extended to incorporate the three time dimensions. Instead of retaining multiple instances of the XML document, we retain a single representation of all successive versions of the document. Although time itself is perceived by most to be continuous, the discrete model is generally used. The time can be bounded in the past and in the future. A finite encoding implies bounds from the left (i.e., the existence of time start) and from the right (time end). In any specific application, the granularity of time has some practical magnitude. For instance, the time point of business event, such as a purchase, is associated with a date, so that a `day` is the proper granule for most business transactions.

Our assumptions

- The time domain T is linear and discrete. A time constant $t = [a, b]$, is either a time instant or a time interval. In a time instant constant, $a = b$, whereas in an interval constant $b > a$. It is clear that the time constant is represented with the beginning and ending instants, in a closed representation. In other words a and b are included in the interval. A bounded discrete representation as integer count of the instants since the origin is our option.
- We limit our event measures to dates (granularity = one day).
- *now* is a special symbol, such that $t < now$ for every $t \in T$, representing current time. We will highlight supporting *now* in 3D_XML in Section 4.

- The valid time constant vt = [*vtStart*, *vtEnd*], *vtStart*, and *vtEnd* represent the valid time start and valid time end, respectively. The efficacy time constant et = [*etStart*, *etEnd*], *etStart*, and *etEnd* represent the efficacy time start and efficacy time end, respectively. The transaction time constant tt = [*ttStart*, *ttEnd*], *ttStart*, and *ttEnd* represent the transaction time start and transaction time end, respectively.

Definition 1. *Valid time of an element/attribute in 3D_XML document D* is represented as n valid time constants $vt_1, vt_2, \ldots, vt_n$, where each vt_i represents a time constant when the element/attribute is valid. Each vt_i is called the i^{th} *version* of *D*. For each pair (vt_i, vt_j), $i, j \in [1, n]$ and $i \neq j$, the following constraint is held:

$$vt_i \cap vt_j = \emptyset \qquad \text{(valid time constants disjunction)}$$

Definition 2. *Efficacy time of an element/attribute in 3D_XML document D* is represented as n efficacy time constants $et_1, et_2, \ldots, et_n$, where each et_i represents a time constant when the element/attribute is efficient. For each pair (et_i, et_j), $i, j \in [1, n]$ and $i \neq j$, the following constraint is held:

$$et_i \cap et_j = \emptyset \qquad \text{(efficacy time constants disjunction)}$$

The efficacy time usually corresponds to valid time; in this case, elements/attributes are retrieved by their valid time. Otherwise, if efficacy time is different from valid time, elements/attributes will be retrieved by their efficacy time**.**

Definition 3. (Inheritance constraints). An element e with a valid time constant vt and an efficacy time constant et having m children $e_1, e_2, \ldots, e_m$, where child e_i has k valid time constants $vt_{i1}, vt_{i2}, \ldots, vt_{ik}$, and q efficacy time constants $et_{i1}, et_{i2}, \ldots, et_{iq}$, is consistent if the following conditions hold:

$$\bigcup_{1 \le j \le k} vt_{ij} \subseteq vt \qquad \text{(valid time inheritance constraint)}$$

$$\bigcup_{1 \le j \le q} et_{ij} \subseteq et \qquad \text{(efficacy time inheritance constraint)}$$

Data manipulation system of 3D_XML (left as future work) preserves the above constraints, i.e. inheritance /disjunction constraints, via user-defined functions.

3.3 Data Modeling

A time-varying XML document records a version history, which consists of the information in each version, along with timestamps indicating its lifetime.

Definition 4. A *three-dimensional XML document* (*3D_XML document* in short) is an XML document in which the three time dimensions, valid time, transaction time, and efficacy time are applied to at least one element/attribute.

Definition 5. A *three-dimensional element/attribute* (*3D_XML element/attribute* in short) is an element/attribute whose content depends on all the three time dimensions.

We will show how temporal elements and temporal attributes can be represented in 3D_XML. A temporal element can be specified in DTD notation as one of the following two structures:

```
(1)   <!ELEMENT element_name+>
      <!ATTLIST element_name vtStart CDATA #REQUIRED
                             vtEnd CDATA #REQUIRED
                             ttStart CDATA #REQUIRED
                             ttEnd CDATA #REQUIRED
                             etStart CDATA #REQUIRED
                             etEnd CDATA #REQUIRED>
(2)   <!ELEMENT element_name+>
      <!ATTLIST element_name inherits CDATA #REQUIRED>
```

We can infer from the above two structures the following observations:

1. A temporal element is represented with one or more elements having the same name; each element represents one version.
2. The three time dimensions are added to a temporal element as attributes. For instance, $\texttt{vtStart}_i$, $\texttt{vtEnd}_i$, $\texttt{ttStart}_i$, $\texttt{ttEnd}_i$, $\texttt{etStart}_i$, and $\texttt{etEnd}_i$ represent valid time start, valid time end, transaction time start, transaction time end, efficacy time start, and efficacy time end, respectively, in the i^{th} version, $i \in [1, n]$. The absence of the above three time dimensions implies that the element inherits them from one of its ancestors; the optional special attribute `inherits` = (1, 2, …, n) represents the first ancestor (parent), second ancestor (parent of parent),…, and the root, respectively.

In [2] elements have timestamps if they are different from the parent nodes. This fact complicates the task of writing queries in XPath/XQuery. We preferred to keep track of timestamps of such kind of elements having their timestamps are not different from the parent (or ancestor) nodes by a special attribute `inherits` representing the ancestor's level. The advantage of this approach is obvious; it facilitates the task of writing powerful queries in XQuery, beside supporting a more effective implementation.

To declare a temporal attribute, the following DTD syntax is used:

```
<!ELEMENT temporal_Attribute+>
<!ATTLIST temporal_Attribute name CDATA #REQUIRED
          value CDATA #REQUIRED
          vtStart CDATA #REQUIRED
          vtEnd CDATA #REQUIRED
          ttStart CDATA #REQUIRED
          ttEnd CDATA #REQUIRED
          etStart CDATA #REQUIRED
          etEnd  CDATA #REQUIRED>
```

We infer from the above form:

- A temporal attribute can be supported in our 3D_XML model by representing it by a special empty element `temporal_Attribute`. Representing time dimensions is similar to time dimensions in temporal elements.
- The name and value of the temporal attribute are represented by special attributes `name` and `value`. The transformation from a `temporal_Attribute` element to an attribute is simple and can be implemented in XQuery.

Example 2: Assume that the history of an employee is described in a 3D_XML document called `employee1.xml` as shown in Figure 3, where we shortened `Start` and `End` substrings to `S` and `E`, respectively, due to the space limitations. The element `employee` has five subelements: `emp_no`, `name`, `dept`, `job` and `salary`.

1. `emp_no`, `name`, and `dept` inherit their time dimensions, i.e. valid time, transaction time, and efficacy time from the first ancestor (parent); this fact is represented by assigning 1 to the attribute `inherits` (`inherits` = "1").
2. Notice that `salary` contains a temporal attribute `currency`. Let us assume that the salary is paid in *crown* before `"2015-01-01"`, and in *euro* after that date due to the expected change of currency in Czech Republic. Notice that the used currency *crown* will be valid till 2014-12-31; valid time end of `temporal_Attribute` element (with the value `crown`) is 2014-12-31.
3. `Anas's` job is changed from `Engineer` to `Sr Engineer` on 2005-09-02. Subsequently, his salary is changed from 60000 to 90000, in the same date. In this case the old version of `salary` (`salary=60000`) is definitely no longer applicable, hence efficacy time has been stopped to `"2005-09-01"` like validity.

```
<employee vtS="2000-01-01" vtE="now" ttS="2000-01-01"
ttE="now"   etS="2000-01-01" etE="now">
 <emp_no inherits=”1”>111</emp_no>
 <name   inherits=”1”>Anas</name>
 <dept   inherits=”1”>Design</dept>
 <job vtS="2000-08-31" vtE="2005-09-01" ttS="2000-08-31"
  ttE="2005-09-30" etS="2000-08-31" etE="2005-09-01">
  Engineer</job>
 <job vtS="2005-09-02" vtE="now" ttS="2005-10-01"
  ttE="now" etS="2005-09-02" etE="now">Sr Engineer</job>
 <salary vtS="2000-08-31" vtE="2005-09-01" ttS="2000-08-
  31" ttE="2005-09-30" etS="2000-08-31" etE="2005-09-
  01">60000
     <temporal_Attribute name="currency" value="crown"
       vtS="2000-08-31" vtE="2014-12-31" ttS="2000-09-01"
       ttE="now" etS="2000-08-31" etE="2014-12-31"/>
     <temporal_Attribute name="currency" value="euro"
       vtS= "2015-01-01" vtE="now" ttS="2000-09-01"
       ttE="now" etS="2015-01-01" etE=="now"/>
 </salary>
 <salary vtS="2005-09-02" vtE="now" ttS="2005-10-01"
  ttE="now" etS="2005-09-02" etE="now">90000
     <temporal_Attribute name="currency" value="crown"
       vtS="2000-08-31" vtE="2014-12-31" ttS="2000-09-01"
       ttE="now" etS="2000-08-31" etE="2014-12-31"/>
     <temporal_Attribute name="currency" value="euro"
       vtS="2015-01-01" vtE="now" ttS="2000-09-01"
       ttE="now" etS="2015-01-01" etE=="now"/>
 </salary> </employee>
```

Fig. 3. employee1.xml**:** information about an employee encoded in 3D_XML

4 Supporting for "now"

Now-relative data are temporal data where the end time of their validity follows the current time. Now-relative data are natural and meaningful part of every temporal database as well as being the focus of most queries [13]. Different approaches are used to represent current time in XML temporal databases. A common approach is to represent current time as unrealistic large date most often used "9999-12-31". Due to the nature of XML and native XML databases to store all data as text, it is possible to represent current time by words such as "now" or "UC" or "∞"; "UC" means (untilchanged). We express a right-unlimited time interval as [*t*, *now*]; although "now" is often used in temporal database literature for valid time, we will use it for all the three time dimensions. Usage of the following user-defined function `check-now` ensures that the temporal query yields the correct answer when a right-unlimited time interval [*t*, *now*] is included in the query.

```
declare function check-now ($d )as xs:date
{if ($d = "now") then xs:date(current-date())
                 else xs:date($d)};
```

As "now" can only appear as a time end of an interval, in case of valid and efficacy time intervals it means a fact is valid and efficient until now, respectively, while in the case of transaction time interval it means no changes until now.

5 Temporal Constructs

For simplicity, in all the following temporal constructs, we omitted the above user-defined function `check-now`.

5.1 Get Time Dimensions

The user-defined functions: `get_vtStart`, `get_vtEnd`, `get_etStart`, `get_etEnd`, `get_ttStart`, and `get_ttEnd` retrieve valid time start, valid time end, efficacy time start, efficacy time end, transaction time start, and transaction time end, respectively. The absence of the above time dimensions implies that the element inherits them from one of its ancestors; note that the level of the ancestor is identified by the special attribute `inherits`. Because of space limitation, we define only `get_vtStart`. The other functions can be defined in a similar way.

```
declare function get_vtStart ($s)
{ let $g := string($s/@inherits)
  return if ($g )
         then  xs:date($s/ancestor::node()[$g]/@vtStart)
         else  xs:date($s/@vtStart)};
```

5.2 Fixed Duration

XML and XQuery support an adequate set of built-in temporal types, including date, dayTimeDuration, making the period-based query convenient to express in XQuery. A user-defined function `fixedDuration` is defined as follows:

```
declare function fixedDuration($node, $length as
xdt:dayTimeDuration)
{let $dur := get_vtEnd($node)- get_vtStart($node)
 return (if ( $dur eq $length) then true()
                               else false())};
```

It checks the length of the valid time interval of the element ($node), and returns true if this length equals a given length ($length), and false otherwise.

5.3 Valid/Efficient Times Relationships Constructs

Here we focus on the temporal constructs related to Valid/efficacy times relationship.

```
declare function valid-notEfficient($a)
{if (get_etStart($a) > get_vtStart($a) or get_etEnd($a)<
     get_vtEnd($a)) then true()else false()};
```

`valid-notEfficient` is a user-defined function which checks if the element (`$a`) is valid but not efficient (if `$a/@etStart > $a/@vtStart` or `$a/@etEnd < $a/@vtEnd` (see Figure 1 and Figure 2)).

5.4 Snapshot Data

Snapshot data – in the literature of databases – in the simplest sense, is the database state in a specific time point. The time point can be the current date (now), or any time point in the past, it can also be in the future, if it is expected that some facts will be true at a specified time after now. Next, we define the snapshot function `dataShot` which can be used to construct snapshots of 3D-XML documents.

```
declare function dataShot ($e, $v)
{ if (get_vtStart($e)<= xs:date($v) and
                    get_vtEnd($e) >= xs:date($v) )
  then element {name($e)}
    {$e/text(),$e/@* except
     $e/@*[string(name(.))="vtStart" or string(name(.))=
     "vtEnd" or string(name(.))="etStart"  or
     string(name(.))="etEnd" or string(name(.))="ttStart"
     or string(name(.))= "ttEnd"], for $c in $e/*
     return dataShot ($c, $v)} else () };
```

Here `dataShot` is a recursive XQuery function that checks the valid time interval of the element and only returns the element and its descendants if `vtStart <= $v <= vtEnd`. (`vtStart`, `vtEnd`) represent valid time interval of the element (`$e`) , while `$v` represnts a specific time point. Note that `except` is an XQuery function discarding the attributes: `vtStart`, `vtEnd`, `etStart`, `etEnd`, `ttStart`, and `ttEnd` from the query's result.

5.5 Interval Comparison Operators

A small library of interval comparison operators is defined to help users with interval-based queries. Due to space limitation we define only three interval comparison operators: *Tcontains*, *Toverlaps*, and *TmeeTs*, respectively.

```
declare function Tcontains ($x, $y)
if ( get_vtStart($x)) <= get_vtStart($y) and
get_vtEnd($x)>= get_vtEnd($y) )then true() else false()};
```

`Tcontains` returns true if one element contains another one and false otherwise; it checks if the valid time interval `$x` contains the valid time interval of `$y`.

```
declare function Toverlaps($x, $y)
{if ( get_vtStart($x) <=  get_vtEnd($y)   and
      get_vtStart($y)  <=  get_vtEnd($x) )
 then true() else false()  };
```

`Toverlaps` checks if the element `($x)` overlaps the element `($y)`.

```
declare function TmeeTs ($x as xs:date, $y as xs:date)
{let $d := $y - $x
 return if(compare($d,"P1D")=0)then true()
                                else false()};
```

`TmeeTs` is a user-defined function checks if the first date `($x)` precedes the second date `($y)` by one day; `"P1D"` is a duration constant of one day in XQuery. Note that `compare` is an XQuery function returning -1, 0, or 1, depending on whether the value of `($d)` is respectively less than, equal to, or greater than one day.

5.6 Break Construct

The valid time constants, efficacy time constants belonging to an element/attribute may appear either with breaks, or without breaks. An occurrence of a *break* implies that there exist at least two versions of the element/attribute, *i* and *i*+1, such that their valid time constants are not adjacent.

Definition 6 (Breaks). An element *e* with a valid time constant *vt* and an efficacy time constant *et*, is said to have *breaks* if there exist at least two versions of *vt*, *i* and *i*+1, such that: $\texttt{vtStart}_{i+1}$ - $\texttt{vtEnd}_i$ is greater than one day ("P1D"), ($i \in [1, n-1]$, n represents the number of versions). If no such versions exist, the element *e* is said to have no *breaks*.

```
declare function Tbreak($g)
{  let $c := count($g) - 1
   let $o := for $i in (1 to $c)
             let $j := $i +1
             return if (TmeeTs(get_vtEnd($g[$i]),
                        get_vtStart($g[$j])))
             then() else "break"
   return count ($o)};
```

The function of `Tbreak` is to check if the temporal element `($g)` has breaks. It calls `TmeeTs` to check every two consecutive versions of the element `($g)`. `Tbreak` returns the number of breaks if exist. `TmeeTs` is defined in Section 5.5.

6 Temporal Queries with XQuery

In all next temporal queries, we omitted the user-defined function `check-now`. Note that, collection C1 consists of XML documents as employee1.xml.

Query 1. Find the employees (their names) who when worked as Engineer, their salaries were 60000 at any time during that period.

```
for $j in collection ("C1")//employee/job[.="Engineer"]
for $s in collection ("C1")//employee/salary[.="60000"]
where $j/../emp_no = $s/../emp_no and Toverlaps($j,$s)
return $j/../name
```

Query 1 checks if the time interval of the `job` element with value `Engineer` overlaps the valid time interval of the `salary` element with value `60000`. `Toverlaps` is defined in Section 5.5

Query 2. Retrieve employees assigned as Sr Engineer but they actually started work in the new position later, return also the inefficient period's length.

```
for $s in collection ("C1")//job[.= "Sr Engineer"]
let $diff := get_etStart($s) - get_vtStart($s)
where (valid-notEfficient ($s))
return if (days-from-duration($diff)>0)
       then <name inefficient_period="{$diff}">
              {data($s/../name)} </name> else ()
```

Query 2 returns the employee name along with the inefficient period length if `valid-notEfficient` returns true. Note that `days-from-duration` is an XQuery function which returns an xs:integer representing the days component in the canonical lexical representation of the value of `$diff`.

Query 3. The next query returns the average of Ebtehal's salaries (paid in crown).

```
for $d in collection ("C1")//employee[name="Ebtehal"]
return <avg>{avg($d//temporal_Attribute[@value="crown"
             and Tcontains(., ..)]/..)} </avg>
```

Query 3 checks if the valid time interval of `temporal_Attribute` element (with the value `crown`) contains the valid time interval of its parent (`salary`); note that `temporal_Attribute` is special empty subelements representing the temporal attribute of an element, so the parent covering constraints is not considered here.

Query 4. Return employee's names who have one break in their employment histories (fired and rehired) and their salaries have been changed for the first time at any time when they are assigned in "Design" department as "Sr Engineer".

```
for $t in collection ("C1")//employee/job[.= "Engineer"]
for $dep in  collection("C1")//employee/dept[.="Design"]
where $t/../emp_no = $dep/../emp_no
return if (Tcontains($dep, $t) and Toverlaps($dep/../
salary[1],$t)and Tbreak($dep/../job)= 1) then $dep/../
name else ()
```

Query 4 shows how a complex query can be greatly simplified by using a number of user-defined functions, i.e., `Tcontains`, `Toverlaps`, and `Tbreak`.

7 Conclusions and Future Work

In this paper, we have introduced a new scheme to represent XML changes without extending the syntax of XML. NXDs represent a suitable storage platform when complex time dependent data has to be manipulated and stored, so we chose to implement temporal queries directly in NXDs (particularly DBMS eXist). Although NXDs provide many functionalities to support XML data (particularly temporal XML data), supporting efficiently temporal queries/updates is a challenging issue. XQuery is natively extensible and Turing-complete [8], and thus any extensions needed for temporal queries can be defined in the language itself. This property distinguishes XML temporal querying from that one in relational temporal languages, e.g. TSQL. So, any syntax extension of XQuery towards temporalness, e.g. τXQuery [4], makes only queries easier to write. We conclude that XML provides a flexible mechanism to represent complex temporal data without extending the current standards [9]. The future work is directed to add more temporal constructs in order to support more powerful temporal queries. Many research issues remain open at the physical level, including the support of updates on historical data. Updates will be a real area of future investigation. Also, in order to improve the performance of our system, we plan to evaluate the effectiveness of the temporal queries.

Acknowledgement. This paper was partly supported by the National programme of research (Information society project **1ET100300419).**

References

1. W3C: Extensible Markup Language (XML) 1.1. 3rd edn. W3C Recommendation (February 04, 2004), http://www.w3.org/TR/xml11/
2. Buneman, P., Khanna, S., Tajima, K., Tan, W.: Archiving scientific data. In: Proc. of ACM SIGMOD Int. Conference, pp. 1–12 (2002)
3. Gergatsoulis, M., Stavrakas, Y.: Representing Changes in XML Documents using Dimensions. In: Proc. of 1st Int. XML Database Symposium, pp. 208–221 (2003)
4. Geo, D., Snodgrass, R.: Temporal slicing in the evaluation of XML queries. In: Proc. of VLDB, Berlin, Germany, pp. 632–643 (2003)
5. Wang, F., Zaniolo, C.: XBIT: An XML-based Bitemporal Data Model. In: Proc. of 23rd Int. Conference on Conceptual Modeling, Shanghai, China, pp. 810–824 (2004)

6. Zhang, S., Dyreson, C.: Adding Valid Time to XPath. In: Proc. of 2nd int. Workshop on Database and Network Information Systems, Aizu, Japan, pp. 29–42 (2002)
7. Grandi, G., Mandreoli, F., Tiberio, P.: Temporal Modelling and Management of Normative Documents in XML Format. Data and Knowledge Engineering 54(3), 227–254 (2005)
8. Kepser, S.: A Simple Proof of the Turing-Completeness of XSLT and XQuery. In: Proc. of Extreme Markup Languages, Montréal, Québec (2004)
9. Ali, K., Pokorný, J.: A comparison of XML-based Temporal Models. In: SITIS 2006. Proc. of 2nd int. conference on Signal-Image Technology & Internet–based Systems, Hammamet, Tunisia, December 17-21, pp. 1–12 (2006)
10. Bourret, R.: Going native: making the case for XML Databases, http://www.xml.com/pub/a/2005/03/30/native.html
11. Wang, F., Zaniolo, C.: Temporal Queries in XML Document Archives and Web Warehouses. In: Proc. of 10th Int. Symposium on Temporal Representation and Reasoning, pp. 47–55 (2003)
12. eXist Home page, http://exist.sourceforge.net/
13. Stantic, B., Governatori, G., Sattar, A.: Handling of Current Time in Native XML Databases. In: Proc. of 17th Australian Database Conference, pp. 1–8 (December 2005)
14. Gergatsoulis, M., Stavrakas, Y., Doulkeridis, C., Zafeiris, V.: Representing and querying histories of semistructured databases using multidimensional OEM. Inf. Syst. 29(6), 461–482 (2004)

Visual Exploration of RDF Data*

Jiří Dokulil[1] and Jana Katreniaková[2]

[1] Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
jiri.dokulil@mff.cuni.cz
[2] Faculty of Mathematics, Physics and Informatics,
Comenius University, Bratislava, Slovakia
katreniakova@dcs.fmph.uniba.sk

Abstract. We have developed and implemented [1,2] infrastructure and RDF storage for the Semantic Web. When we filled it with data the need for some tool that could explore the data became evident. Unfortunately, none of existing solutions fulfills requirements imposed by the data and users expectations. This paper presents our RDF visualizer that was designed specifically to handle large RDF data by means of incremental navigation. A detailed description of the algorithm is given as well as actual results produced by the visualizer.

1 Introduction

The RDF [3] is one of data formats of the Semantic Web. In RDF the information is encoded as a set of statements about resources. These statements may abstractly be viewed as a graph. The data storage for RDF data is at the core of the Semantic Web infrastructure that was created at the Faculty of Mathematics and Physics of the Charles University in Prague [4]. Since its creation a lot of RDF data was loaded into the storage and a query API is available to access the data. However, not knowing the exact structure of the data even programmers using the infrastructure find it difficult to create a meaningful query. We have therefore decided that some kind of visualization tool is definitely necessary to support further development.

Working with RDF data brings up several issues. Most important of them is the size of the data. The data can be huge (millions of nodes and edges) and contain nodes with extremely high degree (thousands or even hundreds of thousands). This not only limits the possibilities of drawing the graph but also the acceptable complexity (both time and space) of the drawing algorithm. Traditional graph-based techniques work very well for small graphs. Unfortunately, the difficulty of finding readable layout extremely increases with the size of graph. We have therefore focused on finding an approach that is effective both from complexity and user point of view. One possibility to partially overcome the

* This research was supported in part by the National programme of research (Information society project 1ET100300419) and VEGA 1/3106/06.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 672–683, 2008.

problem with large data is an incremental navigation [5]. We decided to use the incremental navigation enhanced by our novel *node merging* technique so that we can draw even nodes with large degree. To make the drawing easily readable we proposed a *triangle layout* algorithm [6,7].

The structure of the paper is as follows. Section 2 gives the overview and comparison of known layout algorithms. It also gives a detailed description of our triangle layout. Implementation issues—including the node merging—are described in Section 3. Closing remarks appear in Section 4.

2 Visualization Algorithm

Since the RDF data can be extremely large, some kind of incremental exploration and visualization technique [5] is necessary. The user is given the possibility to explore the neighborhood of the displayed subgraph by extending the displayed part of the graph by one (or more) nodes. This way a *navigation tree* for the data is created. This tree stores the nodes and edges that are currently displayed to the user. We focus on drawing of the navigation tree. The non-tree edges can easily be drawn as lines between corresponding vertices.

2.1 Comparison

There are more approaches to drawing of trees. One of the common techniques is layered drawing where nodes are placed on layers that contain nodes with the same depth. This layers can have different shapes (lines, circles, squares, ...). Examples include:

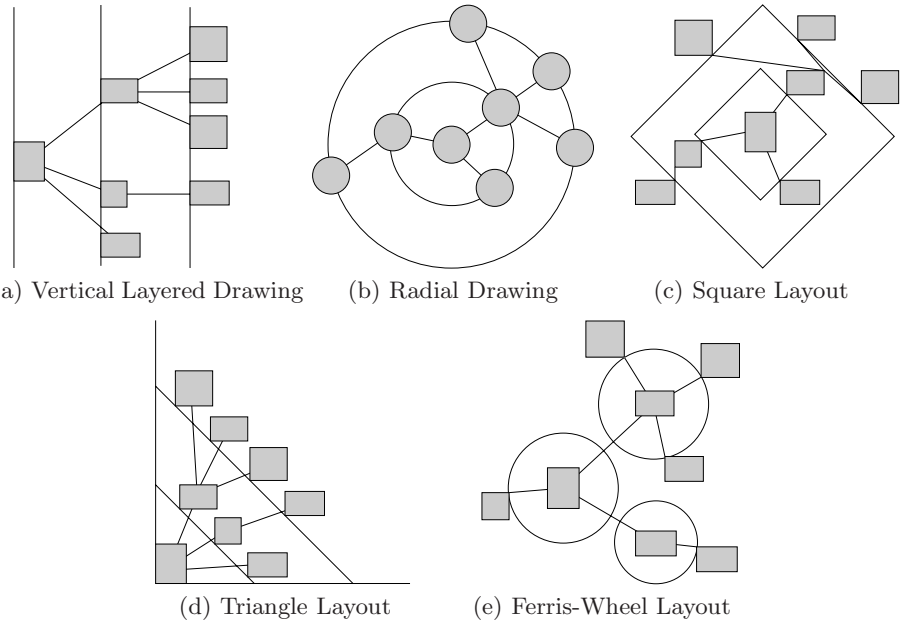
Vertical Layered Drawing (Fig. 1(a)). The layers are vertical lines. It is a very simple approach with good results. The paths in the tree are very easy to follow. The disadvantage is that it is not easy to add not-tree edges to the graph. This approach is used by Experimental RDF Visualizer created in HP labs [8] that avoids non-tree edges by duplicating parts of the graph and transforming it to a tree. Another example is IsaViz [9].

Horizontal Layered Drawing. Is a variant of the vertical layered drawing. It is rarely used because unlike vertical drawing, that offers plenty space for node and edge labels, long node labels make this layout impractical.

Radial Drawing (Fig. 1(b)). The nodes are placed on concentric circles with increasing diameters. The root of the tree is placed in the center. The nodes are usually displayed as circles but as radial drawing is an extremely common technique, there are plenty of variants. Examples of uses of this technique include gnutellavision [10] and GViz [11].

Square Layout (Fig. 1(c)). Square layout is a variant of the radial layout that uses concentric squares instead of circles. It is better suited for drawing rectangular nodes [6].

Triangle Layout (Fig. 1(d)). Triangle layout was introduced as a modification of square layout that uses only the first quadrant of the plane (with coordinate origin in the center of the squares). It is further described in the following parts of this paper.



(a) Vertical Layered Drawing (b) Radial Drawing (c) Square Layout

(d) Triangle Layout (e) Ferris-Wheel Layout

Fig. 1. Layout algorithms

There are more approaches to drawing trees than just layered drawing, including:

Ferris-Wheel Layout (Fig. 1(e)). The Ferris-Wheel layout is inspired by the radial layout but only leaves that are direct neighbors of a node are displayed on a circle around the node. Other nodes are positioned in the drawing space without any sophisticated layout algorithm and positioning them to a 'good' position is left to the user. To handle nodes with high degree, the user is given the option to zoom in on one of the circles (called *wheels*) and gradually explore the nodes by rotating the wheel. This approach is used in PGV [12].

Spring Embedding. Spring Embedding does not specify an exact algorithm for positioning of the nodes. The nodes are connected by *springs* that either pull them together or push them apart. Then the effect of the springs is simulated until a stable position is reached. In the basic version, the connected nodes are connected by springs that pull them together and unconnected nodes with ones that push them apart. By changing power or direction of the springs, layouts with more complex characteristics can be achieved. This approach is used for instance in RDF Gravity [13].

Although many different techniques can be used for the visualization, it is difficult to find a precise way of evaluating them. We have set up several criteria to compare different layout techniques. Some of these criteria are requirements imposed on the layout algorithms by the nature of the RDF data while other criteria were set up to improve user-friendliness of the resulting application. Note

that numbers in parentheses after the criteria definitions correspond to numbers of columns in the Table 1.

Data-imposed criteria. Based on the experience with real RDF data we can assume that the data will contain nodes with high degree. Even such nodes should be displayable without making the visualization unreadable to the user (1). For the same reason the area that can be used to draw children of a node should not be too limited (2). Although it may not always be the case, there is a significant chance that number of nodes on each level will be much larger than on the previous one. Thus the size of the layers increase gradually (3).

User-imposed criteria. The visualization should be *well-arranged*. But there is no general understanding of what that means [14]. We have picked several criteria we believe are important when working with RDF data.

The user should be able to easily locate ancestor and descendants of a node (4). If the user follows a certain path through the tree, then the whole path should at least roughly maintain the same direction (5). Last but not least, the area required to draw the graph should not be too large (6).

Table 1. Comparison of different layout techniques

	1	2	3	4	5	6
Radial Layout	☹	part of annulus wedge	☺	☺	C	A
Vertical Layered Drawing	B	whole layer	☹	☺	to the right	☺
Horizontal Layered Drawing	B	whole layer	☹	☺	downwards	☹
Square Layout	☺	limited	☺	+/–	☹	☺
Triangle Layout	☺	whole layer	☺	☺	C	☺
Ferris-Wheel Layout	☺	not limited	0	☺	☹	☹
Spring Embedding	B	not limited	0	☹	☹	☺

A: The radial layout is best suited for drawing circular nodes. With rectangular nodes the available area can be used inefficiently if the nodes are placed onto the layer in a wrong order. If incremental navigation is used, the correct order cannot be maintained without reordering the nodes.

B: The *node merging* – can be used to handle nodes with high degree.

C: Although the path does not follow a direct route from the center, it generally follows a certain direction without significant deflections.

We have evaluated the listed drawing techniques according to the selected criteria. The results are summarized in the Table 1. The presented results are either claimed by the authors of the individual algorithms or can be easily deduced by examining the algorithms. Although this is certainly not a definitive comparison of existing tree drawing techniques, the results show that the idea of triangle

layout is worth exploring. We have created an experimental implementation. There is currently no other implementation we are aware of.

The next part of the text gives a more detailed description of the triangle layout algorithm and it's properties while Section 3 is focused on the implementation.

2.2 Triangle Layout Algorithm

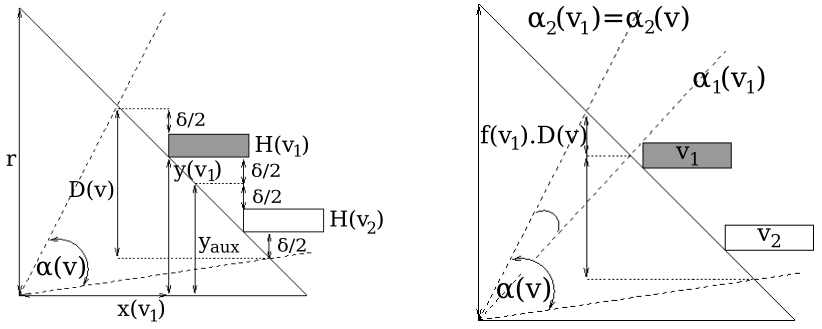
The purpose of the algorithm is to determine positions of the part of the graph that is visible at the moment. The edges of the graph that the user used to reach the visible nodes form a navigation tree T with root r_T. The children of node v are nodes that were reached by exploring edges connecting them to the node v. The order of the children is the same as the order in which they were reached. All nodes with the same distance from the root form a *layer*. A node with distance i from the root is placed in layer l_i (by $L(h)$ we denote nodes on the level h of the tree and $L(0) = \{r_T\}$). Layers are represented as lines connecting $[r_i, 0]$ and $[0, r_i]$, where the value r_i is called *radius of layer* l_i.

The nodes are drawn as rectangles $\Gamma(v)$ that are $H(v)$ pixels high and $W(v)$ pixels wide. They are labeled by URI or literal value of the node they represent and also display a list of edges that start or end in the node (for further details see Subsection 3.1). The rectangle $\Gamma(v)$ representing node $v \in L(i)$ is placed from the outside of the line representing l_i (we place the lower left corner of the vertex onto the line). The corner of the rectangle $\Gamma(v)$ that lies on the layer is denoted $\gamma_0(v)$ in the following text, while the opposite corner is denoted $\gamma_1(v)$. The radius r_i of each level is computed so that $r_{i+1} > r_i$ and to make sure there is enough space to place all nodes that belong to the layer l_i.This is influenced by the fact that we place descendants of node v into a so called *angle of influence* of the node v. The angle of influence is actually defined by two angles that define lower and upper boundary where all descendants (even indirect ones) must fit. This way each path in the tree is given a certain direction to follow, which was one of the user-imposed criteria defined in the previous section. Having $\alpha_1(v), \alpha_2(v) \in \langle 0, 90 \rangle$ and radius r the *vertical range* (height of the available space in pixels) available to node v is

$$D(v) = r.\left(\frac{sin\ \alpha_1(v)}{sin\ \alpha_1(v) + cos\ \alpha_1(v)} - \frac{sin\ \alpha_2(v)}{sin\ \alpha_2(v) + cos\ \alpha_2(v)}\right)$$

We fit the successors of v into this vertical range. Let $v_1 \ldots v_k$ be the children of the vertex v and let v_i have a size of $H(v_i) \times W(v_i)$. If the minimal distance between vertices is δ, then the minimal required vertical space for the children of v is $\Sigma_{i=1}^{k}(H(v_i) + \delta)$. Hence the inequality $D(v) > \Sigma_{i=1}^{k}(H(v_i) + \delta)$ should hold.

The layout algorithm first displays the root on the coordinate origin (i.e. $r_0 = 0$). For each depth h of the tree (beginning with $h = 1$) the algorithm works as follows (see also Fig. 2):



(a) The vertical range (b) Vertical range distribution

Fig. 2. Layout algorithm – explanation

Let r_{cont} be such radius, that triangle $[r_{cont}, 0], [0, 0], [0, r_{cont}]$ completely contains all vertices in layers $l_1 \dots l_{h-1}$. For each vertex $v \in L(h-1)$ the angle of influence has already been computed. Let $v_1 \dots v_k$ be the children of v and $H(v_1) \dots H(v_k)$ their heights. From the inequality $D(v) > \Sigma_{i=1}^{k}(H(v_i) + \delta)$ we compute the minimal required radius r_{min} for the children of v. Let r be the maximum of the minimal required radii and the radius r_{cont}. The vertices from $L(h)$ will be placed on layer with radius r. Radius r of the square and the angle of influence of vertex v determine the vertical range $D(v)$ for the sub-tree rooted in v. The distance $\delta(v)$ between children of v has to be recomputed from the inequality $D(v) > \Sigma_{i=1}^{k}(H(v_i) + \delta(v))$. Now, having global parameter r – radius of the layer and for each vertex $v \in L(h-1)$ the parameter $\delta(v)$, we can compute the display coordinates of children of v and their angles of influence. More formally, for each v_i with height $H(v_i)$ we determine the angle of influence of v_i and the coordinates of $\gamma_0(v_i)$.

The angle of influence of the node v is divided among the children of v according to a function $f: V \to \langle 0, 1\rangle$ where $\Sigma_{i=1}^{k} f(v_i) = 1$.

```
LAYOUT ALGORITHM(T)
 1  γ0(rT) ← [0,0]   //Place the root vertex rT to the coordinates origin
 2  α1(rT) ← 0, α2(rT) ← 90
 3  for each h in {1,2,...}
 4     do
 5        for each v in L(h−1)
 6           do COUNT(rmin(v))
 7        r ← max{rcont, max{rmin(v) | v ∈ L(h−1)}}
 8
 9        for each v in L(h−1)
10           do COUNT(δ(v))
11              D(v) ← r · ( sin α2(v) / (sin α2(v) + cos α2(v)) − sin α1(v) / (sin α1(v) + cos α1(v)) )
12        for each v in L(h−1)
13           do α1(v0) ← α2(v), γ ← α2(v)
14              for i = 1 to k
```

15 **do**
16 $y_{aux} \leftarrow r \cdot \frac{sin\ \gamma}{sin\ \gamma + cos\ \gamma} - H(v_i) - \delta(v)$
17 $\gamma \leftarrow arctg\ \frac{y_{aux}}{r - y_{aux}}$
18 $y(v_i) \leftarrow y_{aux} + \frac{\delta(v)}{2}$
19 $x(v_i) \leftarrow r - y(v_i)$
20
21 $\alpha_2(v_i) \leftarrow \alpha_1(v_{i-1})$
22 $y_{aux} \leftarrow r \cdot \frac{sin\ \alpha_2(v_i)}{sin\ \alpha_2(v_i) + cos\ \alpha_2(v_i)} - f(v_i) \cdot D(v)$
23 $\alpha_1(v_i) \leftarrow arctg\ \frac{y_{aux}}{r - y_{aux}}$

2.3 Vertical Range Distribution

The angle of influence of a node is divided among its children according to the function f. Let v be a node and $u_1 \dots u_k$ children of v. The only constraint for the function f imposed by the algorithm is that $\Sigma_{i=1}^{k} f(u_i) = 1$. The choice of the function greatly affects the behavior of the visualization algorithm. In [6] we proposed the following definition of f.

$$f(u_i) = \frac{H(u_i) + \delta}{\sum_{j=1}^{k}(H(u_j) + \delta)}$$

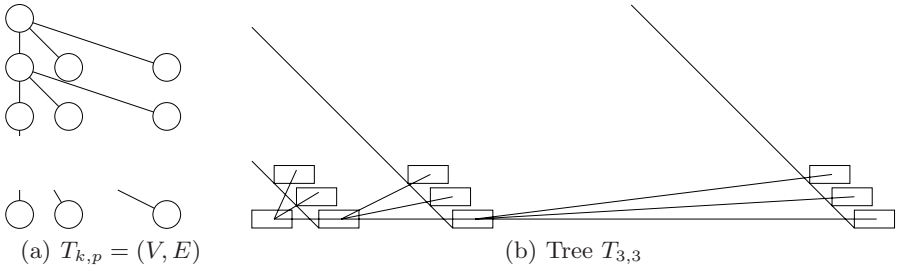
In the following text we use $r_i^{req}(v)$ to denote the minimal radius of layer l_i such that all children of node $v \in L(i-1)$ fit into the angle of influence of the node v. We also use r_i^{req} for $max\{r_i^{req}(v) \mid v \in L(i-1)\}$.

Consider a tree (see Figure 3) $T_{k,p} = (V, E)$ where all nodes are of the same size (H and W) and

$$V = \{v_{0,1}\} \cup \{v_{i,j} \mid i \in \{1 \dots p\} \ \wedge \ j \in \{1 \dots k\}\}$$
$$E = \{(v_{i,1}, v_{i+1,j}) \mid i \in \{0 \dots p-1\} \ \wedge \ j \in \{1 \dots k\}\}$$

On every level of the tree, there is a *critical node* v such that $r_{i+1}^{req}(v) = r_{i+1}^{req}$. Clearly $v_{0,1} \dots v_{p-1,1}$ are critical nodes. We denote $v_{i,1}$ as v_i in the following text.

For a critical node v_i the angle of influence covers $\left(\prod_{j=0}^{i-1} k\right)^{-1}$ of the total vertical range of level l_{i+1}. We need to place k children of v_i into this fraction of the vertical range. Thus r_{i+1}^{req} of the level l_{i+1} is $r_{i+1}^{req} = r_{i+1}^{req}(v_i) = H. \prod_{j=0}^{i} k$.



(a) $T_{k,p} = (V, E)$ (b) Tree $T_{3,3}$

Fig. 3. Example of a tree that requires large area

The number of nodes in the tree $T_{k,p}$ is $N = k.p + 1$, so the radius of l_p is

$$r_p \geq r_p^{req} = H \cdot k^p = H \cdot \left(\frac{N-1}{p}\right)^p$$

So the area required to draw the graph grows exponentially with the number of nodes. This is not a good result from the theoretical point of view and it was also confirmed by the implementation of the algorithm using real-world data.

A better choice seems to be such function f where the value of $f(u_i)$ is the number of nodes of $T(u_i)$ (the tree rooted in u_i) divided by the number of nodes of $T(v)$. In the following text we will prove, that this function produces better drawings of the tree. The number of the nodes of $T(v)$ is denoted $N(v)$ while N denotes the number of nodes in the whole tree. First, we compute $H = max\{H(v) + \delta \mid v \in V\}$ and $W = max\{W(v) + \delta \mid v \in V\}$ and use them as the heights and widths of all nodes in the graph and use $\delta = 0$.

Lemma 1. *Every node v is assigned at least $\frac{N(v)}{N}$ of the vertical range available to the whole layer that the v is positioned on.*

Proof. For the root r_T the statement holds ($\frac{N(v)}{N} = 1$).

Let v be a child of r_T. Then v is assigned $\frac{N(v)}{N-1}$ of the vertical range and $\frac{N(v)}{N-1} > \frac{N(v)}{N}$ so the statement holds.

Let v be a child of u. We already know, that u was assigned at least $\frac{N(u)}{N}$ of the vertical range. This vertical range is divided among the children of u. The node v is assigned $\frac{N(v)}{N(u)-1}$ of the range of u, which totals to $\frac{N(u)}{N}\frac{N(v)}{N(u)-1} = \frac{N(u)}{N(u)-1}\frac{N(v)}{N}$. Since $\frac{N(u)}{N(u)-1}\frac{N(v)}{N} > \frac{N(v)}{N}$ the statement holds. □

Lemma 2. *For every node $v \in L(i-1)$ the required radius $r_i^{req}(v)$ is at most $N \cdot H$.*

Proof. The node v is assigned at least $\frac{N(v)}{N}$ of the vertical range. The range has to be divided among the children of v which means at most $N(v) - 1$ nodes. Height of each child is H so the total height of the children of v is at most $H \cdot (N(v) - 1)$. The $\frac{N(v)}{N}$ fraction of the whole vertical range has to cover the height of the children and since the total vertical range is equal to the radius r_i the value of r_i must be big enough for $r_i\frac{N(v)}{N} \geq H(N(v) - 1)$ to hold. This is equivalent to $r_i \geq H\frac{N(v)-1}{N(v)}N$. The value $r_i = N \cdot H$ fulfills this condition. The condition $r_i \geq max\{r_i^{req}(v) \mid v \in L(i)\}$ implies $r_i^{req}(v) \leq r_i = H \cdot N$. □

For every layer l_i of the tree, the radius r_i^{req} required to fit all children is lesser than $N \cdot H$. The actual radius of layer l_i is one of the following

- r_i^{req} if $r_{i-1} + (H + W) < r_i^{req}$
- $(H + W) \cdot i$ if the path to the root contains no layer j where $r_j^{req} = r_j$.
- $r_j^{req} + (i - j - 1) \cdot (H + W)$ where l_j is the first layer on the path to the root where $r_j^{req} = r_j$.

The maximal number of layers is $N-1$. For the last layer l_p, the r_p is one of the values:

- $r_p^{req} \leq N \cdot H$ (inequation holds due to Lemma 2)
- $(H+W) \cdot p \leq (H+W) \cdot N$ since $p \leq N-1$.
- $r_j^{req} + (p-j-1) \cdot (H+W) \leq r_1^{req} + (p-2) \cdot (H+W) \leq N \cdot H + N \cdot (H+W)$ (Lemma 2 and $p \leq N-1$).

Thus $r_p \leq N \cdot H + N \cdot (H+W) = N(2 \cdot H + W)$. The area required to draw the graph grows quadratically with the number of nodes but also with the value of H and W. Since for rooted trees the layered drawings have quadratic area requirement [15], the area is optimal. The widths of the nodes are limited by the length of the longest label in the data. The heights are limited by the highest node degree present in the data. Although both of these numbers could be potentially very large (causing H and W to be large), for practical reasons they can be limited by much lower threshold (only first part of the labels and some of the edges are displayed). The user may still be given another way of accessing the complete information. This approach is used in our implementation.

3 Implementation

We have implemented the proposed algorithm using the Semantic Web infrastructure developed at the Faculty of Mathematics and Physics of the Charles University in Prague [4,1,2].

The layout algorithm is implemented independent of the data-source and the user interface. At the moment, there is only a SDL-based user interface. This interface displays the drawing to the user and enables him or her to scroll through the whole drawing (it may not fit on the screen) and expand edges by clicking their label in a merged node. For implementation reasons, the drawing is turned upside down, so the origin of the coordinate system is in the upper left corner and the y-axis grows downwards.

3.1 Node Merging

We use our novel technique called *node merging* to help the user navigate the graph. Vertex does not contain only its label but also list of incoming and outgoing edges. This allows us to present the neighbors of the vertex to the user without using too much space. Important advantage of this approach is the fact that the user picks only the neighbors he or she is interested in and the view is then extended only by these vertices. This way we eliminate problem that a RDF node can have thousands (or even hundreds of thousands) of neighbors. Without node merging we would either have to display all of the neighbors which would hardly create a well-arranged and readable drawing of graph or the algorithm would have to pick only a few of the neighbors to display. If node merging is used

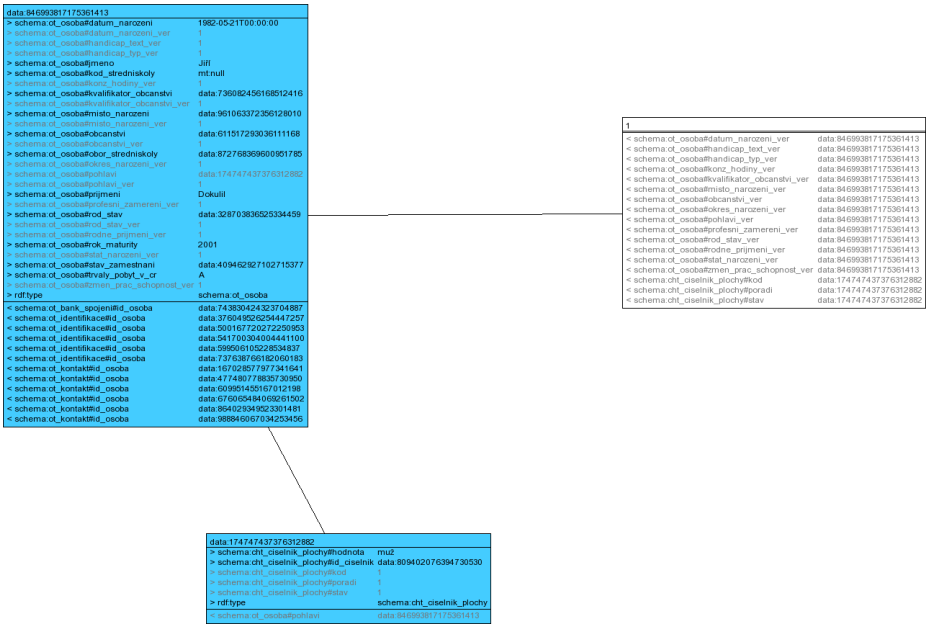


Fig. 4. Example of a two-layered tree

and the number of neighbors is small, the neighbors can be displayed directly in the vertex. If the number is higher, the list of neighbors is displayed in a separate window with the option to filter the displayed entries, which allows handling of even nodes with large number of neighbors.

Node merging is also useful for displaying certain special type of nodes. RDF data usually contain nodes representing certain object with outgoing edges representing its properties, e.g. a person together with his or her name, date of birth, etc. Merged node for the person will contain the name and other information directly so the user can see them without expanding the neighbors. Furthermore a lot of drawing space is conserved since the user will probably be interested in these values and would expand all of the neighbors which may mean adding tens of vertices.

3.2 Animation

When the users expands an edge so that a new node is displayed a drawing of the new tree has to be computed and displayed. To improve the user's experience the transition between the old drawing and the new drawing is animated in real-time. This not only 'looks nice' but more importantly it helps the user maintain connection between objects in the old and the new drawing. Using animation between time-slices to show how nodes and edges are moved to the new positions may also assist in preserving the mental map over time [16].

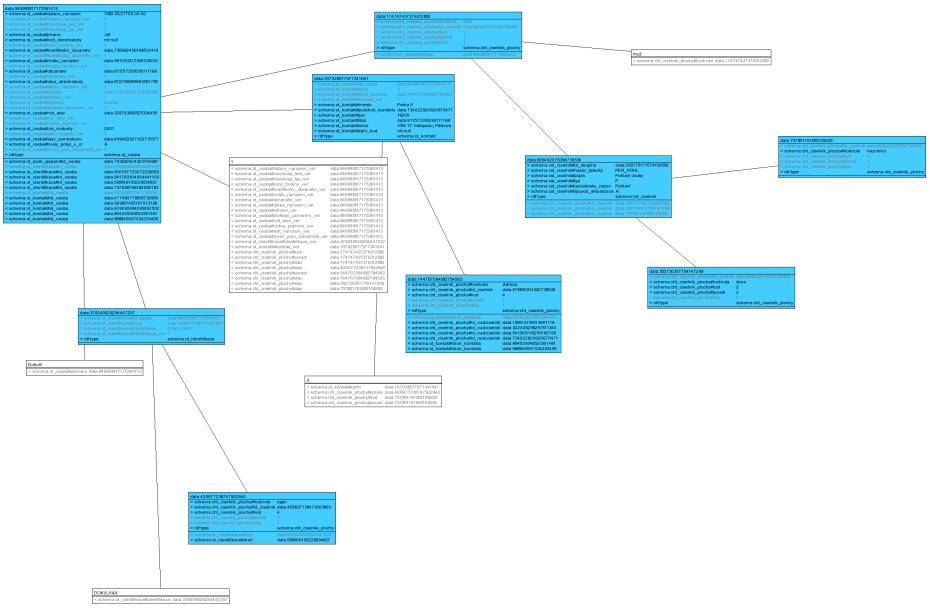


Fig. 5. Example of a large tree

The animation is a simple linear transition of rectangles that represent nodes and lines that represent edges.

3.3 Examples

We have tested the application using the data described in [17]. Figures 4 and 5 are screenshots of some of the visualizations produced by the system.

Darker nodes represent URIs while white nodes are literals. In the list of incoming and outgoing edges, the black can be clicked and new node is displayed, the gray ones represent edges that are already expanded.

4 Conclusion and Future Work

We have designed and implemented a visualization tool to supplement the semantic web infrastructure. The visualization is capable of handling even very large data. We believe it will aid in development of applications that use the infrastructure.

The visualizer could be extended to generate queries based on the displayed data (e.g. by making a query pattern that mirrors the currently displayed graph but literals are converted to query parameters) creating a kind of *query by example* system.

There is still room for improvement in the visualizer itself, especially handling of non-tree edges. Although it is not a significant issue the number of intersections between edges and nodes can be further reduced or even eliminated.

References

1. Dokulil, J., Tykal, J., Yaghob, J., Zavoral, F.: Semantic Web Repository And Interfaces. In: UBICOMM 2007 (includes SEMAPRO 2007), pp. 223–228. IEEE, Los Alamitos, California (2007)
2. Dokulil, J., Tykal, J., Yaghob, J., Zavoral, F.: Semantic Web Infrastructure. In: First IEEE International Conference on Semantic Computing, Los Alamitos, California, pp. 209–215 (2007)
3. Carroll, J.J., Klyne, G.: Resource description framework: Concepts and abstract syntax. W3C Recommendation (2004)
4. Yaghob, J., Zavoral, F.: Semantic web infrastructure using datapile. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Itelligent Agent Technology, pp. 630–633. IEEE, Los Alamitos (2006)
5. Herman, I., Melançon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. IEEE Trans. Vis. Comput. Graph 6(1), 24–43 (2000)
6. Dokulil, J., Katreniaková, J.: Visualization of large schemaless RDF data. In: UBICOMM 2007 (includes SEMAPRO 2007), pp. 243–248. IEEE, Los Alamitos, California (2007)
7. Dokulil, J., Katreniaková, J.: Vizualizácia RDF dát pomocou techniky zlučovania vrcholov. In: Proc. of ITAT 2007: Information Technologies-Applications and Theory, Seňa, Slovakia, PONT, pp. 23–28 (2007)
8. Sayers, C.: Node-centric rdf graph visualization. Technical Report HPL-2004-60, HP Laboratories Palo Alto (April 2004)
9. Pietriga, E.: IsaViz: A Visual Authoring Tool for RDF, http://www.w3.org/2001/11/Isaviz/
10. Yee, K.P., Fisher, D., Dhamija, R., Hearst, M.A.: Animated exploration of dynamic graphs with radial layout. In: INFOVIS, pp. 43–50 (2001)
11. Wood, J., Brodlie, K., Walton, J.: gViz - Visualization Middleware for e-Science. In: VIS 2003. Proceedings of the 14th IEEE Visualization 2003, p. 82. IEEE Computer Society, Washington, DC, USA (2003)
12. Deligiannidis, L., Kochut, K.J., Sheth, A.P.: User-Centered Incremental RDF Data Exploration and Visualization. In: ESWC 2007 (submitted, 2006)
13. Goyal, S., Westenthaler, R.: RDF Gravity (RDF Graph Visualization Tool), http://semweb.salzburgresearch.at/apps/rdf-gravity/user_doc.html
14. Huang, W., Eades, P.: How people read graphs. In: Hong, S.H. (ed.) APVIS. CRPIT, vol. 45, pp. 51–58. Australian Computer Society, Australia (2005)
15. Reingold, E., Tilford, J.: Tidier Drawings of Trees. IEEE Transactions on Software Engineering SE-7, 223–228 (1981)
16. Purchase, H.C., Hoggan, E., Görg, C.: How Important Is the "Mental Map"? – An Empirical Investigation of a Dynamic Graph Layout Algorithm. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 184–195. Springer, Heidelberg (2007)
17. Dokulil, J.: Transforming Data from DataPile Structure into RDF. In: Proceedings of the Dateso 2006 Workshop, Desna, Czech Republic, pp. 54–62 (2006)

Creation, Population and Preprocessing of Experimental Data Sets for Evaluation of Applications for the Semantic Web

György Frivolt, Ján Suchal, Richard Veselý,
Peter Vojtek, Oto Vozár, and Mária Bieliková

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia
Name.Surname@fiit.stuba.sk

Abstract. In this paper we describe the process of experimental ontology data set creation. Such a semantically enhanced data set is needed in experimental evaluation of applications for the Semantic Web. Our research focuses on various levels of the process of data set creation – data acquisition using wrappers, data preprocessing on the ontology instance level and adjustment of the ontology according to the nature of the evaluation step. Web application aimed at clustering of ontology instances is utilized in the process of experimental evaluation, serving both as an example of an application and visual presentation of the experimental data set to the user.

1 Introduction

Exponentially growing volume of information on the Web forces designers and developers to solve navigation and search problems with novel approaches. Faceted browsing, clustering and graph visualizations are only a few possibilities which can be – or even are – currently used. Nevertheless, in order to evaluate how any of these approaches improve navigation or searching, experimental data sets are always needed. Such data sets can be created either by generating artificial data or – as it is in our case – by acquiring data from existing real data sources. While using data from real data sources seem to be an attractive solution, a creation of such experimental data set comes with problems of its own.

We describe the process of creating an experimental evaluation ontology from existing data sources that represents a generalization of the process that we applied in the domain of scientific publications[1]. This process is influenced by the fact that the ontology is used as an experimental evaluation data set for clustering in an application for the Semantic Web. The major contribution of this work is design and experimental evaluation of a framework dedicated to data acquisition, ontology creation, data preprocessing and clustering.

[1] Project MAPEKUS: http://mapekus.fiit.stuba.sk/

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 684–695, 2008.

1.1 Process of Data Set Development

Our approach exploits real data from existing resources as a basis for data set creation. In the process of creating such a data set firstly, suitable data sources, e.g., from the Web, are identified and data from these sources are retrieved (Section 2). Unfortunately, data acquired from various sources rarely share a perfectly common data model definition (ontology) thus a unified ontology is usually created and mappings between other data models are defined (Section 3).

There exist several methodologies of an ontology building process.

In [1] following reuse processes are combined together: *fusion/merge* and *composition/integration*. Using *fusion/merge* approach, the ontology is built by unifying knowledge from two or more different ontologies. In the *composition/integration* approach each reused ontology is a module of the resulting ontology.

Having the unified data model filled with data from all selected sources a cleaning process needs to be started in order to remove, duplications, inconsistencies or even completely wrong data (Section 3.2). Finally, such cleaned data set can be used to evaluate the semantic web-based application. In this paper, we discuss the usage of a data set to evaluate a semantic web application in the domain of scientific publications which exploits graph clustering methods based on graphs extracted from data set ontology (Section 4).

1.2 Domain for Experimentation

We have used proposed approach for an experimental data set development in our research project in the domain of scientific publications. In this project three major scientific publication portals (ACM, `www.acm.org/`, DBLP, `www.informatik.uni-trier.de/~ley/db/` and Springer, `www.springer.com/`) have been selected as the relevant sources for a large common scientific publication database. Acquisition of metadata from these sources resulted into a large amount of instances of authors, publications, journals, etc. Table 1 shows basic statistics indicating the scale of the acquired data sets.

This database serves as an ontology-based meta-data repository in our publication portal which aims at improving the navigation in such large information spaces by utilizing faceted and visual navigation by exploiting the faceted

Table 1. Instance counts of data sets acquired from ACM, DBLP and Springer

	Data Set		
Instance	ACM	DBLP	Springer
Author	126 589	69 996	57 504
Organization	17 161	—	6 232
Publication	48 854	47 854	35 442
Keyword	49 182	—	—
Reference	454 997	—	—

browser and visual navigation in clusters. While faceted browser operates directly on the base ontology, clustering is done on graphs extracted from this ontology. Figure 1 shows the overall process of data acquisition via wrapping, ontology integration, data cleaning, and finally graph extraction and clustering using as an example its special instance for our research project domain.

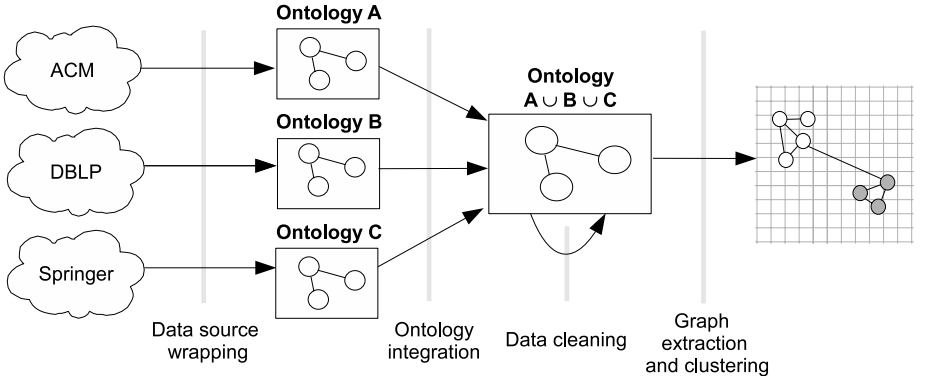


Fig. 1. Overview of the process of data set creation from three various data sources

2 Data Acquisition

The purpose of the data acquisition stage is to create and populate the experimental data set of the respective domain with domain specific data. For evaluation purposes it is necessary to acquire a large amount of data, ideally from several different data sources, in order to create a statistically relevant sample.

2.1 Acquisition Methods

We considered the use of different data acquisition methods ranging from manual approaches through assisted or semi-automatic approaches with only minimum user intervention required to set the specific initialization parameters to fully automatic approaches.

Extraction via web scraping. We employ two types of *web scraping* as suitable means for automatic extraction of huge data sets targeted at predetermined web sources, which contain suitable domain data:

- In case of *wrapper induction* the developer specifies positive and negative examples and the system produces extraction rules [2].
- *Special-purpose automated data gathering* is hand-written and thus custom tailored for a specific purpose – most commonly for the scraping of a particular web site.

Manual data entry. While tedious and not overly effective it is also possible to acquire data manually by data entry workers. This method was considered supplementary and was feasible for small sets of data only, albeit of a high quality. Manual data entry was used in our project scarcely and only when automatic acquisition was not possible.

2.2 Wrapper Induction from User Examples

The wrapper inducer learns the wrapper from positive examples gained from the user. The problem of wrapper induction, the verification and comparison of wrapping approaches was introduced by Kushmerick [3]. The wrappers are induced from examples gained from the user. The process of learning is outlined as Algorithm 1.

Algorithm 1. Wrapper induction learning process

1: open the web page you wish to wrap
2: **repeat**
3: select the example which is an instance of the pattern
4: the wrapper inducer shows the generalization of the examples
5: **until** not all instances of the pattern are listed after the generalization

We call document information pieces we wish to wrap *document patterns*. A pattern is a coherent part of the document. Patterns are filtered out by filters learnt during the training process. The form and representation of the filter is the matter of the learning method. Patterns are often repeating structures, such as items on web pages listing publications of an author or institution. The patterns can be defined in recursive manner.

We describe a document as a set of elements called subdocuments. The subdocuments have attributes. We distinguish several possible *document representations*. The document representations can differ in a) the way of partitioning the documents into subdocuments, b) the attributes describing the subdocuments and c) the relation among the subdocuments. We considered two types of subdocuments so far in our project:

- *XML representation* – the subdocument is represented as W3C Document Object Model (DOM, `www.w3.org/DOM/`) structure. The elements of the documents are addressable by XPath expression. We operate on the XPath expression of the subdocument.
- *Attributed elements* – the subdocuments are also elements of the DOM-tree. The attributes of the subdocuments are type (name) of the element, style class, depth of the element in the DOM-tree and an index (if the element is included in a table).

Recently wrapping from visual web page representation is being researched [4]. Visual representation concerns the visual features visible by the user browsing

a web page. Relying on the HTML structure of the document does not reflect always the web pages' visual features as creators of web pages often do not want the page to be wrapped. Visual representation aims to tackle such situations.

The wrapper inducer is trained on the user examples (subdocuments). The result of the training is a generalization of the examples. We set the following conditions on the generalization: a) each positive example must be in the resulting selection; b) no negative example can be in the resulting selection.

We proposed and implemented two learning strategies for generalization of the document pattern.

Simple XPath learning strategy. The strategy generalizes only from positive examples and operates on the XML document representation. The process of learning starts with an empty filter. The examples are XPath expressions. The algorithm of the learning process is outlined as Algorithm 2.

Algorithm 2. Process of generalization of the XPath expressions in simple XPath learning strategy

```
1: Generalization ← receive the first example from the user
2: while the training process is not terminated do
3:     NewExample ← a new example from the user
4:     update Generalization with NewExample
5: end while
```

Updating the generalization is realized by one of the following actions: a) *index removal* – both XPath expressions are compared element by element from the left side of the expressions. If any index on this path differs this index is removed from the generalization; b) *XPath truncation* – if any tag in the XPath expressions differ, the tag and the path after the tag is truncated.

Attribute selection learning strategy. We apply machine learning methods on the attributes (listed in description of the document representation) of the subdocument. The strategy is able to learn from positive and negative examples. The "positivity" of an example is used as the *next* attribute of the examples. If any of the attribute is missing, its value is set to special value *!missing* and is used in the process of classification [2].

3 Data Preprocessing

Data preprocessing includes methods and procedures which transform the data obtained in the process of data acquisition to a form which is more appropriate for data processing and experimentation.

3.1 Data Integration

In the first phase of data preprocessing it is necessary to integrate data from different sources and thus different models into the one. It includes:

- *Unified data model definition* – data model should be strong enough to represent all instances and their relations from all input sources. For this purpose the domain ontology for publications metadata was created based on generalizations of relevant parts of source data models.
- *Data mapping definition* – for every data source a definition of data mapping between source data model and unified data model is required. In our case, this transformation is executed already in wrappers.

3.2 Data Cleaning

This part of the process is aimed at cleaning of inconsistencies, which are present in data because of:

- *Inconsistencies in the source data model* – inconsistencies and other flaws like data duplicities, incomplete or wrong data can be transferred from source model.
- *Inconsistencies due the source integration* – because data are integrated from many sources, duplicities of instances for example authors or publications may occur.
- *Inconsistencies created in the wrapping process* – there is a chance that duplicates are created during the wrapping process, for example not all links to the authors can be followed and checked due to high increase of the process duration.

Single-pass instance cleaning. This phase is designed to correct data flaws in scope of one instance like:

- correcting the format of some data types, e.g., names, which should start with capital letter;
- separating data fields, e.g., separating first names and surname;
- filtering of instances with insufficient data to work with them;
- filtering of data fields, e.g., conjunctions from key terms.

The instance cleaning is realized as a set of filters, one for each particular task. This solution is based on pipes and filters architecture. Whole process can be realized by one pass through all instances and therefore has linear time complexity. It is effective to realize it directly after acquisition of each instance using a wrapper yet before storing it because the processor is only lightly loaded while downloading data from sources.

3.3 Duplicates Identification

The aim of this phase is to identify instances that are describing identical entity (e.g., author or publication). As the acquisition process is based on the semantics of the domain and extracts data together with their meaning to common domain ontology no approaches for comparing concepts and discovering a mapping between them [5] are needed.

We combine two methods, comparing data itself (in terms of ontology data type properties) and working with relations between data (object type properties), which was already described in [6].

The overall similarity of the instances is computed from similarity of their properties that are not empty. For each property of an instance we use weighting method with positive and negative weight. This is more general than simple weighting, which in many cases is not sufficient. For example, consider the country property of two authors – if the similarity is low it means that the authors are not likely the same person (need for strong weight), but if it is high, it does not mean, the authors likely are the same person (need for low weight). The parameters are three values, positive weight p, negative weight n, and threshold t, which determines where to use positive weight and where to use negative. Overall similarity $S \in <0,1>$ is calculated as:

$$S = \frac{\sum_{i=1}^{n} F_i\left(s_i\right) + n_i}{\sum_{i=1}^{n} p_i - n_i} \tag{1}$$

where n is number of properties, s_i represents the similarity of i-th properties ($s_i \in <0,1>$), p_i stands for positive weight, n_i for negative (values between 0 and 100) and F_i represents step function, which is calculated as

$$F_i(x) = \begin{cases} p_i x & \text{if } x \geq t_i \\ -n_i\left(1-x\right) & \text{if } x < t_i \end{cases} \tag{2}$$

where i is the index of property, p_i is positive weight, n_i negative and t_i stands for threshold of i-th property (value between 0 and 1).

To decide whether the instances are identical a threshold is applied. If the similarity is above its value the instances are evaluated as identical.

Every instance is compared to every other instance of its class, which leads to quadratic asymptotic time complexity. To shorten the duration of comparing process we use a simple clustering method for authors and divide them into groups by the first letter of their surnames.

Comparison of data type properties. This method is based on comparison of corresponding data type properties of two instances of the same class. For the data comparison there are used several string similarity metrics like QGrams, Monge-Elkan distance, Levenshtein distance and many other[2]. Different method can be specified for each property including so called *composite metric*, which is weighted combination of several metrics (for example QGrams with weight of 0.6 and Monge-Elkan, weight 0.4), where weights are set according to the results of the experiments. We also use special metrics for some properties, e.g.,

[2] Chapman, S.: SimMetrics, `www.dcs.shef.ac.uk/~sam/stringmetrics.html`, Cohen, W.: Record Linkage Tutorial: Distance Metrics for Text, `www.cs.cmu.edu/~wcohen/Matching-2.ppt`

names where we consider abbreviations of their parts. Some properties needs to be compared only for identity, e.g., ISBN of the books.

Comparison of object type properties. The principle of this method is to compare the relations to neighboring instances in the ontology. With increasing number of mutual instances the probability that compared instances are the same grows.

For each object property (e.g., authors of two books) we compare their datatype properties to determine how close to each other they are. Each match is included in computing the overall similarity. Thus if two books have three mutual authors, all three partial similarities are counted. The same approach also applies on the authors that are different.

3.4 Duplicates Resolving

We identified several possibilities when two instances were identified as identical:

- mark instances as identical via special object property,
- manually resolve (this possibility is appropriate, when small number of duplicates was found),
- delete the instance with less information,
- join instances, take the data type properties with higher length, join non-functional object properties.

3.5 References Disambiguation

Aim of this phase is to identify and disambiguate identical references between publications [7]. Parts of references can be recognized using regular expressions and set of experimentally discovered rules. The process of identity identification between two references consists of three steps:

1. *Comparison of titles* – the title similarity is identified using the Levenshtein distance. If the ratio of this value and the length of both strings greater than 5% (value set as a result of experiments with publications metadata) references are considered not to be identical. Otherwise the process continues with the next step.
2. *Comparison of years* – if the years in references are different, references are not identical. If they are equal proceed to step three.
3. *Comparison of authors* – if at least one author is mutual for both references, they are considered to be identical.

4 Data Processing

After the proceprocessing is done, processing and presentation are applied. In our case user navigation in the metadata enhanced data set is employed, e.g. faceted navigation [8] and visual navigation in clusters of data instances. We describe as an example of processing the instance clustering.

4.1 Extracting Graphs from Ontologies

While data instances usually incorporate complex representation of a state-space, clustering all the instances bound together with all types of properties is computationally demanding. In such cases, it is worth utilizing a graph extraction from the ontology.

We represent ontology by the Web Ontology Language (OWL, www.w3.org/TR/owl-features/). The OWL representation consists of triplets (RDF based subject-predicate-object construction [9]), which serve as the basis for graph representation of ontology.

Example in Figure 2 shows a scientific publication ontology which consists of instances which belong to classes "Author", "Paper" and "Keyword" and properties between instances are "hasKeyword", "isWrittenBy" and "references", and extraction of all vertices of class "Author" and edges defined by the property "isWrittenBy".

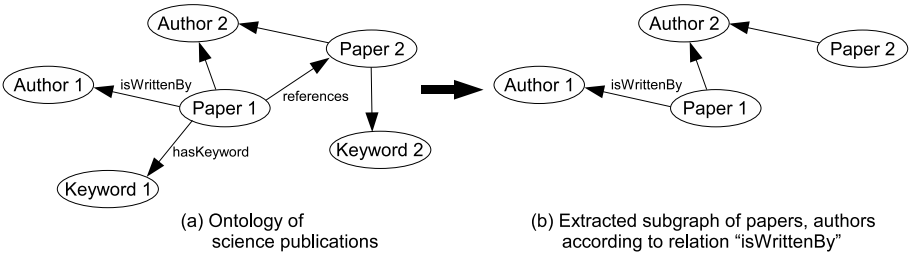


Fig. 2. Extraction of graph of papers and authors from ontology of publications

4.2 Clustering

The process of clustering utilizes the unsupervised learning schema [10], which is advantageous when no exact taxonomy of data set exists. The disadvantage of involvement of clustering is that created clusters usually do not have attached semantic labels, hence resolving why particular data instances were grouped together can be hard in some cases. An example of graph-based clustering is depicted in Figure 3.

Composing hierarchy of clusters. We represent the information in a space of hierarchical clusters. While many clustering methods were invented over the time [11], almost any clustering method can be simply used when it is properly implemented according to the JUNG library (Java Universal Network/Graph Framework, jung.sourceforge.net/), used during the clustering. Hierarchical composition of clusters is implicitly used. Clustering of the input graph is following:

1. Input graph is loaded from relational database and converted to a graph representation using the JUNG library.

2. Clustering parameters are determined, e.g., which clustering method is used, number of cluster layers in the hierarchy, number of vertices in cluster (if the selected clustering method is designed to be parameterized in such way).
3. Layers of the cluster hierarchy are created sequentially from bottom to top.
4. Whole hierarchy of clusters is stored in relational database. With the aim to allow any operations with the clusters, also inverted hierarchy is generated and stored.

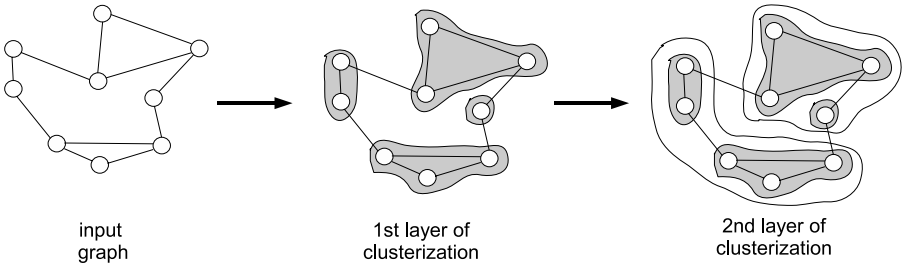


Fig. 3. Example of two layered graph clustering

5 Evaluation – Duplicate Identification

To measure the effectiveness of duplicity identification process data from DBLP together with artificially created duplicates were used. The instances were randomly selected and the duplication of one instance included:*(i)* Mutation of randomly selected datatype properties with one of these operations: words exchange, letter exchange, word deletion, letter deletion, letter duplication; *(ii)* For each object property there is a chance to be deleted or its reference to be duplicated and its datatype properties mutated like in previous step.

Ten measurements were made for each data set of 1 000, 5 000, 10 000 and 20 000 publications, with 100 injected duplicities. Surname clustering for authors, Levenstein metric for publication titles, special name metric for author given and family names and exact string comparison for ISBN were used. These metrics were weighted for each property to achieve best results. Similarity threshold of 0.8 was used (each pair of instances, that achieved score of 0.8 and greater on the scale from 0 to 1 were considered to be duplicity).

In Table 2 there are columns with number of identified duplicities (column *correct*), wrong identified duplicities so called false positives (column *wrong*) and unidentified duplicities (column *missing*). It contains also statistical measures such as precision, recall and F1 measure. Each cell contains average value and standard deviation.

Only instances that could be identified after clustering are considered in these measures, because clustering is not a part of identification method. Without clustering, the results should be same for all 100 duplicates, but the whole process would be much more time consuming.

Precision is almost constant in each data set. There is a linear growth in number of false positives, but number of compared instances grows also linear and number of comparisons grows quadratic.

Table 2. Results of duplicate identification experiment on various DBLP sample sizes

	Duplicity identification					
Sample size	*Correct*	*Wrong*	*Missing*	*Precision*	*Recall*	*F1*
1 000	53.2 ± 4.23	5.7 ± 7.52	14.1 ± 4.37	0.79 ± 0.06	0.90 ± 0.08	0.84 ± 0.04
5 000	70.6 ± 5.93	6.6 ± 5.95	18.8 ± 7.05	0.79 ± 0.07	0.92 ± 0.05	0.85 ± 0.05
10 000	74.5 ± 6.07	20.7 ± 5.20	17.7 ± 4.19	0.81 ± 0.05	0.78 ± 0.04	0.80 ± 0.04
20 000	81.3 ± 4.38	24.7 ± 5.58	13.9 ± 2.98	0.85 ± 0.03	0.77 ± 0.02	0.81 ± 0.02

Number of duplicates in real data was also measured. Data from each source and data combined from all sources were used (Table 3). Found duplicates were checked manually and over 90% of them were correct.

Table 3. Result of duplicate identification experiment on real data

Sample size	*ACM*	*DBLP*	*Springer*	*Combined*
1 000	44	5	10	24
2 000	96	7	24	36
3 000	133	25	41	48
4 000	154	26	55	62
5 000	175	29	68	78

Some interesting facts have been discovered in the domain of publications:

- Many duplicities of the publications are present only because of capital letters.
- Chinese names are hard to distinguish, because they are very short.
- There is a problem with various editions and re-editions of books, their title differs only in one number or short word.
- The family members, e.g. brothers writing the same books and having mutual collaborators are hard to distinguish.

6 Conclusions

The data integration process is important when evaluating applications for the Semantic Web on real data sets from various sources. However, only involvement of powerful data acquisition and preprocessing methods can ensure the quality of the resulting data set.

In our research, we focused on integration of the data from various large scale data sources in the domain of scientific publications, from which relevant data was acquired using two different *web scraping* approaches and stored in the ontological repository for further processing and support of navigation.

During the evaluation we also encountered several bottlenecks that currently seriously limit experimenting and widespread deployment of applications for the Semantic Web, mainly the general immaturity of ontological repositories in terms of their processing speed of ontological queries. Having hundred thousands of instances in the ontology, preprocessing realized along with data gathering from their source proved as well-suited approach.

Several of proposed methods for data preprocessing presented in this paper are also suitable for merging various sources of meta-data in particular domain order to provide effective visualization and navigation in data.

Acknowledgment. This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT-20-007104 and the Scientific Grant Agency of Slovak Republic, grant No. VG1/3102/06.

References

1. Pinto, H.S., Peralta, D.: Combining Ontology Engineering Subprocesses to Build a Time Ontology. In: K-CAP 2003, pp. 88–95. ACM Press, New York (2003)
2. Čerešňa, M.: Interactive Learning of HTML Wrappers Using Attribute Classification. In: Proc. of the First Int. Workshop on Representation and Analysis of Web Space, Prague, Czech Republic, pp. 137–142 (2005)
3. Kushmerick, N.: Wrapper Induction: Efficiency and expressiveness. Artificial Intelligence 118(1), 15–68 (2000)
4. Simon, K., Lausen, G.: ViPER: Augmenting Automatic Information Extraction with Visual Perceptions. In: CIKM 2005, pp. 381–388. ACM Press, New York (2005)
5. Weinstein, P.C., Birmingham, W.P.: Comparing Concepts in Differentiated Ontologies. In: KAW 1999 (1999)
6. Andrejko, A., Barla, M., Tvarožek, M.: Comparing Ontological Concepts to Evaulate Similarity. In: Návrat, P., et al. (eds.) Tools For Acquisition, Organisation and Presenting of Information and Knowledge, STU, pp. 71–78 (2006)
7. Rado, L.: Sharing of Research Results on Portal based on Semantic Web. Master's thesis project report, Bieliková, M. (supervisor), Slovak University of Technology in Bratislava (2007)
8. Tvarožek, M., Bieliková, M.: Adaptive Faceted Browser for Navigation in Open Information Spaces. In: WWW 2007, pp. 1311–1312. ACM Press, New York (2007)
9. Beckett, D.: Redland RDF Storage and Retrieval. In: SWAD-Europe Workshop on Semantic Web Storage and Retrieval (2004)
10. Hinton, G., Sejnowski, T.J.: Unsupervised Learning and Map Formation: Foundations of Neural Computation. MIT Press, Cambridge (1999)
11. Frivolt, G., Pok, O.: Comparison of Graph Clustering Approaches. In: Bieliková, M. (ed.) IIT.SRC 2006, Bratislava, Slovakia, pp. 168–175 (2006)

Algorithm for Intelligent Prediction of Requests in Business Systems

Piotr Kalita, Igor Podolak, Adam Roman, and Bartosz Bierkowski

Institute of Computer Science, Jagiellonian University, Krakow, Poland
{kalita,roman,bierkows}@ii.uj.edu.pl,uipodola@theta.uoks.uj.edu.pl

Abstract. We present an algorithm for intelligent prediction of user requests in a system based on the services hosted by independent providers. Data extracted from requests is organized in a dynamically changing graph representing dependencies between operations and input arguments as well as between groups of arguments mutually coexisting in requests. The purpose of the system is to suggest the possible set of future requests basing on the last submitted request and the state of the graph. Additionally the response time may be shortened owing to the background executing and caching of the requests most likely to be asked. The knowledge extracted from the graph analysis reveals the mechanisms that govern the sequences of invoked requests. Such knowledge can help in semi-automatic generation of business processes. The algorithm is a part of ASK-IT (Ambient Intelligence System of Agents for Knowledge-based and Integrated Services for Mobility Impaired users) EU project[1].

1 Introduction

Systems that predict the user requests are widely investigated [1,3,4,5,7,8,9]. Their purposes are twofold: firstly recommending the user some action that he is likely to submit as an option and in consequence dynamic customization of his environment and secondly so called *prefetching* - decreasing the request latency by obtaining in advance the response to the request likely to be issued or *intelligent caching* - caching the responses to most popular requests. Typical areas of applications are web usage mining [3] or prediction of web service calls [8].

Typically the prediction algorithms are based on Markov chain approach. There are many variants of Markovian algorithms: point–based (where the submitted request is assumed to depend only on the previous one), path–based (where the predicted query is based on several preceding ones) [9], cascading Markov models (being the combination of former two) [1], classification pruning algorithms [7], algorithms that use generalized sequences of requests [3], Markov models with aging [4]. Markov approach is highly memory consuming and has

[1] This research was partially funded by the European Union 6th Framework project ASK-IT (IST-2003-511298).

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 696–707, 2008.

large computational complexity, untolerable if the requests are invocations of some operations with changing arguments. Also, it is not scalable: typically the request prediction is based on the analysis of the server log and not the real time request processing. Another approach is through frequent sets and association rules [5]. All requests are searched and the subsets where the probability density of the individual values of requests is relatively high are found. Then, using for example the *Apriori* algorithm (see [6] Chapt. 8.2), frequent sets (X, Y) are partitioned and translated into rules of type $X \Rightarrow Y$. Still this methodology consists of inspecting the server log files.

The objective of this article is to propose the new prediction algorithm that is faster and less memory consuming than above ones. Implementation of proposed algorithm is the part of the system implemented in ASK-IT [2] EU project. The general goal of the ASK-IT system is to help mobility impaired users while moving and traveling. Each user is supposed to have a PDA device (*e.g.* a palmtop, mobile phone) through which he submits queries about the route to another place, places of interest, help calls, etc. The reply is provided by a system of services. Since the number of users in the system is high and their requests are similar, the system can be optimized with the help of proposed algorithm by *intelligent caching* and storing the information for a number of users with similar objectives. At the same time the request prediction system helps mobility impaired user with taking up decisions by suggesting the most probable request basing on historical data stored in the graph structure. It is assumed that the request is an invocation of the operation (method) with some arguments. Used prediction algorithm is *graph based* that highly reduces the complexity since the graph operations are performed locally and furthermore independent parts of the graph can be accessed in parallel. Therefore the real time prediction and updating of the structures is possible. This is crucial in the ASK-IT system as it is required not to store any user related data on the server side (in particular the history of the requests). Also aging is taken into account in proposed algorithms.

2 System Architecture

The architecture of the system that takes advantage of the proposed request prediction algorithm, as it is going to be implemented in ASK-IT is depicted in the Fig.1. The client system (1), which is possibly a PDA mobile device issues a service request to the Management Module (2) that is also responsible for *matchmaking*, i.e. associating one of the service providers with the issued request. Request Prediction System RPS (3), and service providers (4) are available to the Management Module as web services. Once the management module receives a client request, it issues it to the provider that delivers the reply and, concurrently, its filtered copy to RPS that delivers the predicted further requests. Independently it is possible that, in idle time, RPS submits the most popular requests to the providers and *pre-fetches* the replies to the cache memory. Then, if the set of suggested requests contains the ones with cached answers, RPS sends the answers to the user together with the queries.

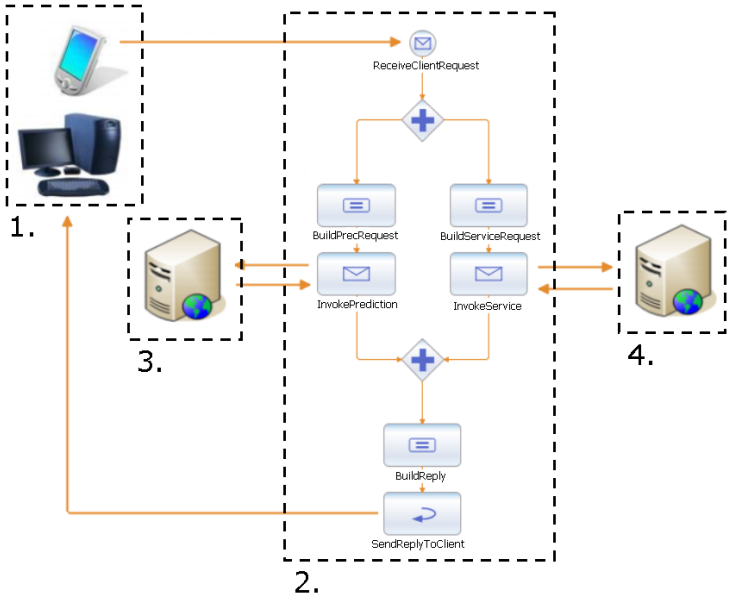


Fig. 1. Overview of service based system that includes the prediction of requests. 1. Client machine (PDA). 2. Management Module that associates the client request with the service provider and issues the request to the provider and the prediction system. 3. Request Prediction System (RPS). 4. Service provider.

The message that is sent to RPS consists of the name of the previous operation issued by the user, and the name of the current operation (method) together with the values of its parameters that together constitute the instance of a method call. The operation and parameters names are the part of the set of namespaces which constitutes the system ontology. The answer that is issued by RPS is constituted by the number of operations and parameters. Such selection is combined with the actual reply from the service provider and sent back to client as a suggestion for future possible requests. If the answer to some of the suggested requests is cached in RPS, then the user receives it instantly together with the request suggestion. Reply sent back by RPS is constructed basing on the graph data structure stored therein - Request Prediction Graph (RPG) and the user request. RPG is also updated basing on the request data. In its data structures RPS remembers the logical dependence between two consecutive operations, since the name of the previous request is included in the RPS call. We remark that no personal user data is remembered by RPS.

Web service architecture is recommended for RPS: then the input (request and previous request operation) and output (collection of suggested requests) are represented by SOAP messages described by Schema contained in WSDL interface of RPS. The Schema depends on the Schemas of the service providers of the system.

We remark that in presented model predicted requests depend on the request only and not on the answer to this request. RPS predicts next user queries basing on:

- frequency of associations of argument values with the name of operation,
- frequency of mutual coexistence of argument values in the request,
- frequency of the succeeding pairs of operations.

The first two of above frequencies are stored as weights of vertices in RPG and the third one, represents the automatic detection of logical dependence of between two consecutive operations, is remembered in a *matrix*.

3 RPG Graph

Request Prediction Graph (RPG) is a 6-tuple $(G, cons, popm, parw, kind, ord)$, where:

1. $G = (MET \cup PAR, WM \cup WP)$, $MET \cap PAR = \emptyset$, $WM \cap WP = \emptyset$;
2. $WM \subseteq MET \times PAR$, $WP \subseteq \binom{PAR}{2}$;
3. $cons: \ N^2 \to \mathbb{Q}$, $popm: \ MET \to \mathbb{Q}$, $parw: \ PAR \to \mathbb{Q}$;
4. $kind: \ MET \to N$;
5. $ord: \ MET \times PAR \to \mathbb{N}$;
6. $\forall m \in MET \ \{ord(m,p): \ (m,p) \in WM\} = \{1, 2, ..., deg(m)\}$.

N is the set of all methods (or, more general, operations) available in the system. Graph G consists of two sets of vertices (two layers), MET and PAR. Each vertex from MET represents an instance of a method from N. MET consists of methods called in the system. Each vertex from PAR represents an argument (parameter). Function $kind$ assigns to each instance $M \in MET$ a method (from N) represented in G by vertex M. WM and WP are the edges: edges in WM, between method and parameter, denote that parameter was used in a method call as an argument, while edges in WP, between pairs of parameters denote that they mutually appeared in a call.

There are three different types of weights between the elements of G:

- each parameter $p \in PAR$ has its weight $parw(p)$,
- each $M \in MET$ has it's weight $popm(M)$, representing the 'popularity' of a method,
- each pair $(M_1, M_2) \in MET^2$ has it's weight $cons(kind(M_1), kind(M_2))$, which represents the connection between two methods invoked consecutively. This value is also defined for $kind(M_1) = kind(M_2)$. Function $cons$ can be represented as a square matrix of size $N \times N$, called Method Consequence Matrix.

Invocation of a method $X(p_1, p_2, ..., p_k)$ with k parameters is represented in G by the subgraph $G_X = (\{M, p_1, ..., p_k\}, E_P \cup E_M)$ such that:

1. $M \in MET, \forall 1 \leq i \leq k \quad p_i \in PAR$;
2. $E_P \subset WP$, $E_M \subset WM$;
3. $E_P = \{\{p_i, p_j\} : \ 1 \leq i, j \leq k, \ i \neq j\}$;
4. $E_M = \{(M, p_i) : \ 1 \leq i \leq k\}$, $ord(M, p_i) = i$.

In other words, all elements from $\{p_1, ..., p_k\}$ form a k-clique and each of them is connected with M. Function *ord* remembers the order of the parameters.

4 Algorithms

We introduce two operations on RPG. The first one is connected with a graph update. Graph is updated whenever a request is invoked by the client. The second operation is used for actual prediction of the user's future request. This prediction, in the simplest version, is based on the last request submitted by the same user.

UpdateGraph (listing 1) presents the algorithm which performs the first operation. Whenever a new request $R = M(p_1, ..., p_k)$ is submitted (where M is a method, p_i its arguments), we add to PAR all the vertexes representing arguments p_i, which does not belong to PAR yet. Also, if the request has not been issued before we add to MET the vertex R representing the instance of method M. Then the edges are added between all pairs of p_i's and each of p_i's and R. The subgraph G_R that contains p_i's and R with associated edges corresponds to $M(p_1, ..., p_k)$. Next three weights are increased we increase (by 1): 1) weights for each argument p_i, 2) weight for R and 3) the (L, M)-entry in Method Consequence Matrix, where L is the method invoked by the same user just before invoking R. If at least one of the weights exceeds a given threshold, all the weights of this type in RPG are to be divided by 2 (thus the aging is introduced).

PredictRequest (listing 2) presents the algorithm for request prediction. The idea is as follows: suppose that user issued in the request the operation L (arguments are irrelevant) and then another request, represented by $M(p_1, ..., p_k)$. The algorithm takes the 1-neighborhood N in PAR layer of the set $P = \{p_1, ..., p_k\} \subset$ PAR, that is, the set of all vertexes $v \in$ PAR such that the distance between v and at least one of p_i is at most 1. 1-neighborhood is required in order to produce requests that are highly associated with the current one. Larger neighborhoods could lead to results loosely connected with the original request. Then, $M^* \subset$ MET is chosen in a following manner: we choose all method instances with sets of parameters totally included in N. We remove M from M^*. Next, for each method from M^* we compute it's rank regarding to the corresponding weights in RPG. For each $m \in M^*$, representing method $m(p_1, ..., p_k)$, we take the mean of the weights of edges forming the clique $(p_1, ..., p_k)$, multiply it by the 'popularity' of method m and, finally, multiply the result by the (M, m)-entry of Method Consequence Matrix. Methods with highest ranks are suggested to the user. Ranking function is simplistic in order to simplify the process of prediction that is complicated already. Of course other heuristic ranking formulas are possible and the best formula is going to be chosen basing on the system performance.

The construction of this algorithm comes from the nature of ASK-IT system, for which it is designed and implemented. In this system the sets of parameters in consecutive methods, invoked by user, have usually nonempty intersection. For example, if user wants first to find route from A to B, next he(she) usually wants to find some points of interest in B or the route from B to C. The parameter 'B' is present in consecutive requests. Therefore the algorithm is designed in a 'parametric-oriented' manner that motivates the use of 1-neighborhood.

5 Example

In this section we provide a very simple example that elucidates the algorithm work. Let us suppose that the tourist recently asked the system for the route from Wieliczka to Cracow. Next he(she) wants to get from Cracow to Auschwitz and submits a request to the system on the preferred route. On this request RPS updates the *RPG* and predicts a new request.

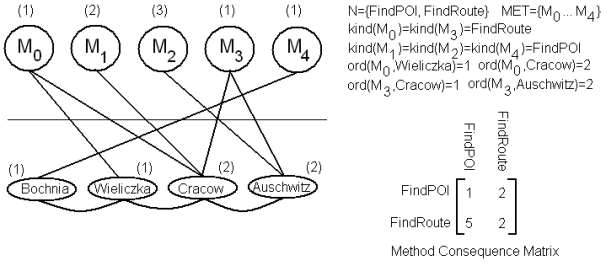


Fig. 2. Initial RPG graph. Numbers in brackets over vertexes representing instances of the methods are the *popm* weights, indicating 'popularity' of individual instances. Numbers below arguments represent their weights, *parw*.

Fig.2 presents the graph before the request FindRoute(Cracow,Auschwitz) and Fig.3 shows the graph modification after the user's request.

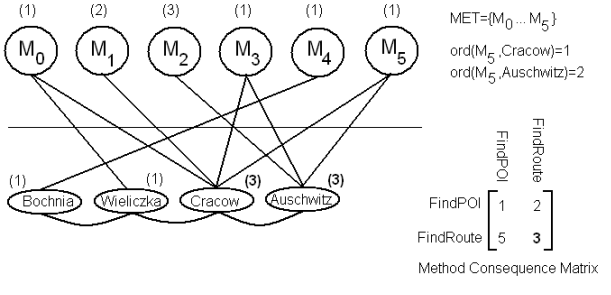


Fig. 3. Graph modified by request M_5

Algorithm 1. UpdateGraph($R = (M, P), L$)

Input: Current request R (method M and parameters list P) and a previous request L submitted by the same user

begin
foreach $p \in P$ **do** // update the set of parameters
    if $p \notin PAR$ **then** $PAR \leftarrow PAR \cup \{p\}$;
    foreach $p, q \in P : p \neq q$ **do**
        if $\{p, q\} \neq WP$ **then** $WP \leftarrow WP \cup \{p, q\}$; //make a clique P
$t \leftarrow 0$;
foreach $p \in P$ **do** // update weights in the parameters layer
    $parw(p) \leftarrow parw(p) + 1$;
    if $parw(p) >$ threshold_parw **then** $t \leftarrow 1$;
if $t = 1$ **then** ModifyParW();
$Y \leftarrow P$;
$y \leftarrow$ take_from(Y); //take first element from the list and remove it
// X_1 is the set of methods for which y is the first parameter
$X_1 \leftarrow \{m \in MET : \ kind(m) = M \wedge (m, y) \in WM \wedge ord(m, y) = 1\}$;
$i \leftarrow 1$;
if $Y \neq \emptyset$ **then**
    repeat
        $y \leftarrow$ take_from(Y);
        $i \leftarrow i + 1$;
        // X_i is the set of methods for which y is the i-th parameter
        $X_i \leftarrow X_{i-1} \cap \{m \in MET : kind(m) = M \wedge (m, y) \in WM \wedge ord(m, y) = i\}$;
    until $X_i = \emptyset \ \vee \ Y = \emptyset$;
// X_i is the set of all methods with parameters set P and with the same order as in method M. If $X_i = \emptyset$ then add a vertex representing instance of M.
if $X_i = \emptyset$ **then** // notice that $|X_i| = 0$ or $|X_i| = 1$
    $MET \leftarrow MET \cup$ instance(M); // instance(M) is a vertex representing M
    foreach $p \in P$ **do**
        $WM \leftarrow WM \cup$ (instance(M), p);
    $popm$(instance(M)) $\leftarrow 1$; // initialize the weight of a new method
    $cons(L, M) \leftarrow cons(L, M) + 1$;
    if $cons(L, M) >$ threshold_cons **then** ModifyCons();
else // M is already present in MET, so update weights only
    $e \leftarrow$ take_from(X_i);
    $popm(e) \leftarrow popm(e) + 1$;
    if $popm(e) >$ threshold_popm **then** ModifyPopM();
    $cons(kind(e), M) \leftarrow cons(kind(e), M) + 1$;
    if $cons(kind(e), M) >$ threshold_cons **then** ModifyCons();
end.

Finally, in Fig. 4 prediction result is presented:

1. The set of parameters of the current request is $P =$ {Cracow, Auschwitz};
2. Vertex M_5 is added to MET, because no user has submitted such request before. The $popm$ value for M_5 is set to 1;

Algorithm 2. PREDICTREQUEST($R = (M, P)$)

1: Input: Request R (method M and parameters list P)
2:
3: **begin**
4: // in lines 5.-12. we find the vertex representing the instance of M (there can be at most one such vertex)
5: $Y \leftarrow P$;
6: $y \leftarrow$ take_from(Y);
7: $X_1 \leftarrow \{m \in MET: \ kind(m) = M \wedge (m, y) \in WM \wedge ord(m, y) = 1\}$;
8: $i \leftarrow 2$;
9: **while** $Y \neq \emptyset \wedge X_{i-1} \neq \emptyset$ **do**
10: $y \leftarrow$ take_from(Y);
11: $X_i \leftarrow X_{i-1} \cap \{m \in MET: \ kind(m) = M \wedge (m, y) \in WM \wedge ord(m, y) = i\}$;
12: $i \leftarrow i + 1$;
13: $Z \leftarrow X_{i-1}$; // notice that $|Z| = 0$ or $|Z| = 1$;
14: //if $|Z| = 0$ then M or some of it's parameters were removed from the graph
15: **if** $Z = \emptyset$ **return** 0; // we predict nothing
16: **else**
17: $Neigh \leftarrow P$; // Neigh will represent P and its 1-neighborhood
18: $P_R \leftarrow \emptyset$; // P_R will be the set of methods whose parameters sets are totally included in Neigh (lines 19.-23.)
19: **foreach** $p \in P$ **do** $Neigh \leftarrow Neigh \cup \{r \in PAR: \ \{p, r\} \in WP\}$;
20: **foreach** $n \in Neigh$ **do**
21: $T \leftarrow \{m \in MET: \ (m, n) \in WM\}$;
22: **foreach** $t \in T$ **do**
23: **if** $\{p \in PAR: \ (t, p) \in WM\} \subset Neigh$ **then** $P_R \leftarrow P_R \cup \{t\}$;
24: $P_R \leftarrow P_R \backslash$take_from($Z$); // exclude current request from the set of predictions
25: **foreach** $q \in P_R$ **do** // compute ranks of all request from P_R
26: $v \leftarrow 0$;
27: **foreach** $p \in P : (q, p) \in WM$ **do**
28: $v \leftarrow v + parw(p)$;
29: $v \leftarrow v/|P|$; //v = mean weight in clique of parameters
30: $rank(q) \leftarrow cons(M, kind(q)) \cdot popm(q) \cdot v$;
31: SORTANDCUT(P_R); //sort requests regarding their ranks and leave only
32: //a fixed number of requests with the highest rank
33: **end.**

Algorithm 3. MODIFYPARW()

//used for parameters layer weights adaptation

1: **begin**
2: **foreach** $p \in PAR$ **do**
3: $parw(p) \leftarrow parw(p)/2$;
4: **if** $parw(p) <$ min_threshold_parw **then** $parw(p) \leftarrow$ min_threshold_parw;
5: **end.**

Algorithm 4. ModifyCons()

```
//used for Method Consequence Matrix weights adaptation
1: begin
2: foreach m ∈ N do
3:     foreach n ∈ N do
4:         cons(m,n) ← cons(m,n)/2;
5:         if cons(m,n) < min_thresh_cons then cons(m,n) ← min_thresh_cons;
6: end.
```

Algorithm 5. ModifyPopM()

```
//used for request 'popularity' weights adaptation
1: begin
2: foreach m ∈ MET do
3:     popm(m) ← popm(m)/2;
4:     if popm(m) < min_threshold_popm then
5:         P ← {p ∈ PAR : (m,p) ∈ WM};
6:         foreach p ∈ P do
7:             WM ← WM \ (M,p);
8:             if {t ∈ M : (t,p) ∈ WM} = ∅ then
9:                 foreach q : {q,p} ∈ WP do
10:                    WP ← WP \ {q,p};
11:                PAR ← PAR \ {p};
12:        MET ← MET \ {m};
13: end.
```

3. The neighborhood of P is found and added to P; now P = {Cracow, Auschwitz, Wieliczka};
4. For each $p \in P$ we find the set M^* of methods with arguments included in P. We have $M^* = \{M_0, M_1, M_2, M_3\}$. Notice that $M_4 \notin M^*$, because one of it's parameters, Bochnia, does not belong to P ={Cracow,Auschwitz,Wieliczka}. $M_5 \notin M^*$ because M_5 is the current request and there is no sense in including it in the prediction set.
5. Algorithm computes the ranks of all $m \in M^*$. We have $rank(M_0) = 3 \cdot 1 \cdot 3 = 9$, $rank(M_1) = 5 \cdot 2 \cdot 3 = 30$, $rank(M_2) = 5 \cdot 3 \cdot 3 = 45$, $rank(M_3) = 3 \cdot 1 \cdot 3 = 9$.

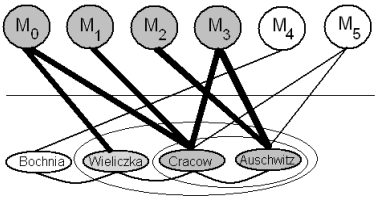


Fig. 4. Prediction of a future request

Therefore the most probable request found by the algorithm is FindPOI (Auschwitz). The next one is FindPOI(Cracow).

6 Algorithm Optimization and Tests

Implementation remarks. In order to provide scalability, the system should be able to switch into the mode of queueing graph updates and performing them off-line in idle time. It is possible to store the requests of individual users in one global queue and then update the structure when the system is free, since the system evolves at much slower time scale then the one at which the requests are issued.

Since graph algorithms are used, the implementation is straightforward with on-line structure update. The system can provide concurrent access to distinct parts of the graph that allows for higher efficiency.

Simulations. In order to check the predictive effectiveness of the algorithm a simple experiment was performed. It was assumed that the set of methods was given by $N = \{M_1, M_2, M_3, M_4\}$ and the set of parameters by PAR $= \{p_1, p_2, p_3, p_4\}$. Furthermore the set MET of all requests (method instances) that can be issued were consists of 5 elements: $I_1 = M_1(p_1, p_2), I_2 = M_2(p_1, p_3), I_3 = M_3(p_2), I_4 = M_4(p_2, p_3, p_4), I_5 = M_1(p_1, p_4)$. The RPG graph (without weights) corresponding to those instances is given in the Fig. 5.

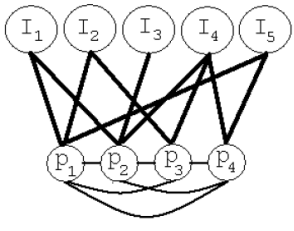


Fig. 5. The RPG graph (without weights) used in the experiment

The arbitrary Markov matrix $M(i,j)_{i,j=1,\ldots,5}$ over the set of states $\{I_1, \ldots, I_5\}$ was selected and according to this matrix the chain that simulates the sequence of user requests was built. According to this chain the request prediction system was taught, using the UpdateGraph algorithm. Then for each of 5 states the PredictRequest algorithm was issued. As the result, for each instance the ranking of the succeeding calls was produced. Two criteria were used to evaluate the algorithm performance. First, the instance that has the highest rank (i.e. is predicted to be most likely to be issued) was compared with the highest entry in the corresponding row of the Markov matrix. The performance measure was the number of instances for which the most probable request is predicted. Second measure was the Euclidean norm $S = \sqrt{\sum_{i=1}^{5} \sum_{j=1}^{5} (M(i,j) - M'(i,j))^2}$, where

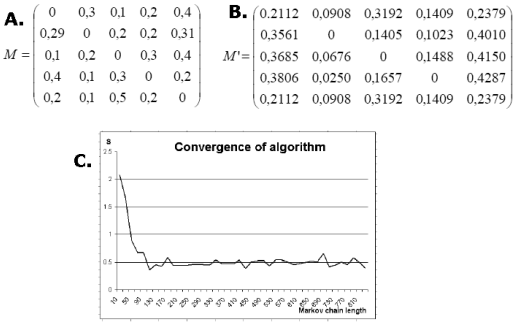


Fig. 6. A. Markov matrix used in the tests. B. Matrix of weighed instance ranks obtained after learning with the chain of length 1000. C. Dependence of the S measure on the length of the Markov chain used to learn.

$M'(i, j)$ is the matrix that as the rows has vectors of ranks of all succeeding predicted requests normalized to 1.

The matrix M used in the test is depicted in Fig.6A and M' obtained by learning with the chain of length $n = 1000$ is presented in Fig.6B. In 3 instances out of 5 request predicted as most likely to be issued has the highest probability in Markov matrix. The measure S between M and M' as the function of the length of the chain used by the algorithm to learn is depicted in the Fig.6C. In particular for $n = 830$ we obtain $S = 0.3944$. We also observe that the S measure quickly converges to some optimal efficiency and is not further improved.

7 Concluding Remarks

The article presents the algorithm for predicting user requests in web service environment. Algorithm takes into account the dependencies between operations as well as between arguments. Also aging of the data is taken into account. Data structures and algorithms are designed in such a way that the system is scalable with respect to the number of users. The system is being implemented now within the ASK-IT project [2].

Prediction of requests is typically considered in the framework of web usage where the request is understood as the retieval of a static web page. In such context the resource ranking algorithms, like the HITS algorithm (see [6] Chapt. 8.8) or the web usage analysis algorithms, like the mining path-traversal patterns (see [6] Chapt. 8.9) are efficient. The approach presented in this article strongly takes into account the context of each request, i.e. the values of the arguments used in the current call. Therefore our approach is rather uncomparable to the mentioned ones.

Presented approach predicts several requests that are most likely to be issued. It is therefore possible that the call patterns that are useful, although unfrequent, will be hidden from the user. This issue is partially resolved by aging introduced

in UpdateGraph algorithm through division the weights by 2 while keeping their increasing by 1. This flattens differences between weights.

Some improvements and extensions will be the subject of future research. In particular the further tests are required in order to tune the algorithms and verify their scalability. We plan to run tests that use larger number of methods, in particular methods and arguments from ASK-IT ontology, multiple number of short Markov chains in order to simulate multi user behavior, finally use the real user requests. We also plan a verification system in an open environment, where a user ranks the predicted requests himself. This verification could then be used for the system fine-tuning, allowing to develop the automatic, dynamical change of the parameters, such as thresholds or calculation of weights.

Semantics of parameters is currently represented through the position in the issued request and association with the parameters of other requests that have the same semantic type, however further incorporation of semantics into the model is also planned.

References

1. Ashamala, J.: User modelling on the world wide web. Master's thesis, Monash University, Melbourne (2004)
2. ASK-IT: Ambient Intelligence System of Agents from Knowledge Based and Integrated Services for Mobility Impaired Users, Eurepean Union 6th Framework project, http://www.ask-it.org
3. Gery, M., Haddad, H.: Evaluation of web usage mining approaches for user's next query prediction. In: Proceedings of WIDM 2003, pp. 74–81 (2003)
4. Haffner, E.-G., Roth, U., Engel, T., Meinel, C.: Modeling of time and document aging for request prediction - one step further. In: Proceedings of Symposium on Applied Computing ACM SAC, pp. 984–990 (2000)
5. Hastie, H., Tibshiranie, R., Friedman, J.: The elements of statistical learning. Springer, Heidelberg (2001)
6. Kantardzic, M.: Data Mining, Concepts, Models, Methods and Algorithms. IEEE Computer Society, Wiley-Interscience (2003)
7. Li, I.T., Yang, Q., Wang, K.: Clasification pruning for web request prediction. In: WWW 2001. Proceedings of the 10th World Wide Web Conference (2001)
8. Mancini, E., Villano, U., Mazzocca, N., Rak, M., Torella, R.: Performance-driven developement of a web services application using metapl/hesse. In: PDP 2005. Proceedings of 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing, pp. 12–19 (2005)
9. Su, Z., Yang, Q., Lu, Y., Zhang, H.-J.: Whatnext: A prediction system for web requests using n-gram sequence models. In: WISE 2000. Proceedings of the 1st International Conference on Web Information System and Engineering, pp. 200–207 (2000)

Mining Personal Social Features in the Community of Email Users

Przemysław Kazienko and Katarzyna Musiał

Institute of Applied Informatics, Wrocław University of Technology,
Wyb.Wyspiańskiego 27, 50-370 Wrocław, Poland
{kazienko, katarzyna.musial}@pwr.wroc.pl
http://www.zsi.pwr.wroc.pl/~kazienko

Abstract. The development of structure analysis that constitutes the core part of social network analysis is continuously supported by the rapid expansion of different kinds of social networks available in the Internet. The network analyzed in this paper is built based on the email communication between people. Exploiting the data about this communication some personal social features can be discovered, including personal position that means individual importance within the community. The evaluation of position of an individual is crucial for user ranking and extraction of key network members.

The new method of personal importance analysis is presented in the paper. It takes into account the strength of relationships between network members, its dynamic as well as personal position of the nearest neighbours. The requirements for the commitment function that reflects the strength of the relationship are also specified. In order to validate the proposed method, the dataset containing Enron emails is utilized; first to build the virtual social network and afterwards to assess the position of the network members.

Keywords: email communication, user ranking, social network analysis, personal importance, social features in community.

1 Introduction

The various kinds of e-commerce and e-business solutions that exist in the market encouraged the users to utilize the Internet and available web-based services more willingly in their everyday life. Many customers look for services and goods that have high quality. Thus, not only the information provided by vendors is important for potential customers but also the opinions of other users who have already bought the goods or used the particular service. It is natural that users, to gather other people opinions, communicate with each other via different communication channels, e.g. by exchanging emails, commenting on forums, using instant messengers, etc. This information flow from one individual to another is the basis for the social network of users (*SNU*). This network can be represented as a directed graph, in which nodes are the users and the edges describe the information flow from one user to another. One of the most meaningful and

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 708–719, 2008.

useful issue in social network analysis is the evaluation of the personal importance within the network. Since the social network describes the interactions between people, the problem of assessment the personal importance becomes very complex because humans with their spontaneous and social behavior are hard predictable. However, the effort should be made to evaluate their status because such analysis would help to find users who are the most influential among community members, possess the highest social statement and probably the highest level of trust [12], [21]. These users can be representatives of the entire community. A small group of key persons can initiate new kinds of actions, spread new services or activate other network members [18]. On the other hand, users with the lowest position should be stimulated for greater activity or be treated as the mass, target receivers for the prior prepared services that do not require the high level of involvement. In order to calculate the position of the user, the new measure called personal importance is introduced in the further sections. It enables to estimate how valuable the particular user within the *SNU* is. In contrary to the PageRank algorithm that is designed to assess the importance of the web pages, the presented personal importance measure take into account not only the significance of the direct connections of a person but also the quality of the connection.

2 Related Work

The main concept of a regular social network can be described as a finite set of nodes that are linked with one or more edges [10], [13], [24]. A node of the network is usually defined as an actor, an individual, corporate, collective social unit [24], or customer [26] whereas an edge named also a tie or relationship, as a linkage between a pair of nodes [24]. The range and type of the edge can be extensive [13], [24] and different depending on the type and character of the analyzed actors.

The social networks of users somewhat differ from the regular ones and because of that they yield for new approaches to their definition and analysis. *SNU* is also called an online social network [10], computer-supported social network [25], or web community [11]. Note that there is no one coherent definition of *SNU*. Some researchers claim that a web community can also be a set of web pages relevant to the same, common topic [11]. Adamic and Adar argue that a web page must be related to the physical individual in order to be treated as a node in the online social network. Thus, they analyze the links between users' homepages and form a virtual community based on this data. Additionally, the equivalent social network can also be created from an email communication system [1]. Others declare that computer-supported social network appears when a computer network connects people or organizations [10], [25].

In order to analyze the social networks, the social network analysis (SNA) should be performed. It focuses on understanding the connections among people and the implications of these connections [24]. Thus, the main goal of SNA can be defined as follows: "a methodology for examining the structure among

actors, groups, and organizations and aides in explaining variations in beliefs, behaviours, and outcomes" [14]. SNA provides some measures useful to assess the personal importance within the social network. To the most commonly used belong: centrality, prestige, reachability, and connectivity [13], [24]. There exist many approaches to evaluation of person centrality [9]: degree centrality, closeness centrality, and betweeness centrality. Degree centrality takes into account the number of neighbors that are adjacent from the given person [13]. The closeness centrality pinpoints how close an individual is to all the others within the social network [2]. It tightly depends on the shortest paths from the given user to all other people in the social network. The similar idea was studied for hypertext systems [3]. Finally the betweeness centrality of a member specifies to what extend this member is between other members in the social network [9]. Member a is more important (in-between) if there are many people in the social network that must communicate with a in order to make relationships with other network members [13]. The second feature that characterizes an individual in the social network and enables to identify the most powerful members is prestige. Prestige can be also calculated in various ways, e.g. degree prestige, proximity prestige, and rank prestige. The degree prestige takes into account the number of users that are adjacent to a particular user of the community [24]. Proximity prestige shows how close are all other users within the social community to the given one [24]. The rank prestige [24], is measured based on the status of users in the network and depends not only on geodesic distance and number of relationships, but also on the status of users connected with the user [15].

Another popular measures used for internet analysis is PageRank, which was introduced by Brin and Page to assess the importance of web pages [4], [6], [7]. The PageRank value of a web page takes into consideration PageRanks of all other pages that link to this particular one. Google uses this mechanism to rank the pages in their search engine. The main difference between PageRank and personal importance proposed in this paper is the existence and meaning of commitment function. In PageRank, all links have the same weight and importance whereas personal importance makes the quantitative distinction between the strengths of individual relationships.

3 Mining Personal Importance in the Community

Many different personal social features can be considered in the context of computer communication. However, in this paper, we will focus on the centrality measures, particularly on the personal importance. Before the new method for personal importance measure is presented, the definition of social network of users should be established.

3.1 Social Network of Users

The various kinds of definitions of the social network of users (see Sec. 2) yields for the creation of one consistent approach.

Definition 1. *Social network of users is a tuple SNU=(UID,R), where UID is a finite set of non-anonymous user identities i.e. the digital representation of a person, organizational unit, group of people, or other social entity, that communicate with one another or participate in common activities, e.g. using email system, blogs, instant messengers. R is a finite set of relationships that join pairs of distinct user identities:* $R = \{(uid_i, uid_j) : uid_i \in UID, uid_j \in UID, i \neq j\}$. *Note that relationships are asymmetric, i.e.* $(uid_i, uid_j) \neq (uid_j, uid_i)$. *The set of user identities UID must not contain isolated members – with no relationships and card(UID)>* 1.

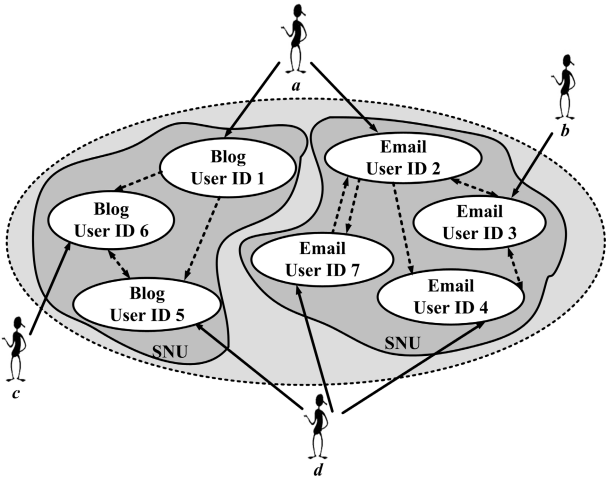


Fig. 1. Two social networks of users

The example of two separate social network of users is presented in Fig. 1. Note that an individual human can simultaneously belong to many social networks in the Internet. Moreover, they can also maintain several Users IDs – see person *d* in Fig. 1. The user identity is a digital representation of the physical social entity. These are objects that can be unambiguously ascribed to one person (individual identity), to a group of people or an organization (group identity). This representation must explicitly identify the social entity (a user, group of users or an organization). This mapping enables to define the connections between social entities based on the relationships between their identities. An individual identity possesses individuals, whereas a group identity corresponds to a group of people, e.g. family that use only one login to the family blog, as well as to an organization, e.g. all employees use one e-mail account to respond customers' requests. Such group identities can by identified by content analysis.

A relationship connects two user identities based on their common activities. Every social entity that is represented by the user identity can be conscious of such relationship or not, depending on the profile of activities. Three kinds of social relationships can be distinguished: Direct relationship – it connects two

user identities with a direct connector. The direct connector is an object that is addressed to the specific type of user identities and related communication, e.g. email addresses (user identities) are connected with messages exchanged among them. Thus, the direct connector can be email communication, phone calls (or VoIP), etc. Quasi–direct relationship – two user identities are aware of the fact that they are in the relationship but they do not maintain the relationship, e.g. people who comment on the same blog. Indirect relationship – the user identity is not aware of the fact that is similar to other user identity. Two user identities are connected by indirect relationship when their profiles are similar, e.g. people who examine and similarly rate the same photos published in the Internet. The examples of *SNU* based on the established definition are: a set of people who date using an online dating system [5], a group of people who are linked to one another by hyperlinks on their homepages [1], the company staff that communicate with one another via email [8], [22], etc.

3.2 Personal Importance Evaluation

Based on the data derived from the source system, we can build a graph that represents the connections between users and then analyze the position of each person within such network. Nodes of the graph represent the users who interact, cooperate or share common activities within the web-based systems while edges correspond to the relationships extracted from the data about their common communication or activities.

Definition 2. *Personal importance function PI(a) of user a respects both the value of personal importance of user's a connections as well as their contribution in activity in relation to a, in the following way:*

$$PI(a) = (1 - \varepsilon) + \varepsilon \cdot (PI(b_1) \cdot C(b_1 \rightarrow a) + ... + PI(b_m) \cdot C(b_m \rightarrow a)) \qquad (1)$$

where: ε *— the constant coefficient from the range* $[0, 1]$*, the same for all* $a \in UID$*. The value of* ε *denotes the openness of personal importance on external influences: how much personal importance is more static (small* ε*) or more influenced by others (greater* ε*);* b_1*,...,*b_m *— acquaintances of a, i.e. users that are in the direct relation to a;* m *— the number of a's acquaintances;* $C(b_1 \rightarrow a)$*,...,*$C(b_m \rightarrow a)$ *— the function that denotes the contribution in activity of* b_1*,...,*b_m *directed to a.*

In general, the greater personal importance one possesses the more valuable this member is for the entire community. It is often the case that we only need to extract the highly important persons, i.e. with the greatest personal importance. Such people surely have the biggest influence on others. As a result, we can focus our activities like advertising or marketing solely on them and we would expect that they would entail their acquaintances. The personal importance of a user is inherited from others but the level of inheritance depends on the activity of the users directed to this person, i.e. intensity of common interaction, cooperation or communication. Thus, the personal importance depends also on the number

and quality of relationships. To calculate the personal importance of the person within the social network the convergent, iterative algorithm is used. This means that there have to be a fixed appropriate stop condition τ.

3.3 Commitment Function in Email Communication

The commitment function $C(b \rightarrow a)$ is a very important element in the process of personal importance assessment, thus it needs to be explained in more detail.

Definition 3. *The commitment funcion $C(b \rightarrow a)$ reflects the strength of the connection from user b to a. In other words, it denotes the part of b's activity that is passed to a. The value of commitment function $C(b \rightarrow a)$ in SNU(UID,R) must satisfy the following set of criteria:*

1. The value of commitment is from the range $[0;1]$*:* $\forall(a, b \in UID)$ $C(b \rightarrow a) \in [0;1]$.

2. The sum of all commitments has to equal 1, separately for each user of the network:

$$\forall(a \in UID) \sum_{b \in UID} C(a \rightarrow b) = 1. \tag{2}$$

3. If there is no relationship from b to a then $C(b \rightarrow a) = 0$.

4. If a member b is not active to anybody and other n members a_i, $i = 1,...,n$ are active to b, then in order to satisfy criterion 3, the sum 1 is distributed equally among all the b's acquaintances a_i, i.e. $\forall(a \in UID)$ $C(b \rightarrow a_i) = 1/n$.

Since the relationships are reflective and with respect to criterion 3, the commitment function to itself equals 0: $\forall(a \in UID)$ $C(a \rightarrow a) = 0$. The example of network of users with values of commitment function assigned to every edge is presented in Fig. 2. According to the above criteria all values of commitment are from the range $[0;1]$ (criterion 1) as well as the sum of all commitments equals 1, separately for each user of the network (criterion 2). Moreover, there is no relationship from b to a so $C(b \rightarrow a) = 0$ (criterion 3). Note also that user c is not active to anybody but b and d are active to c, so according to condition 4, the commitment of c is distributed equally among all c's connections $C(c \rightarrow b) = C(c \rightarrow d) = 1/2$.

The commitment function $C(a \rightarrow b)$ of member a within activity of their acquaintance b can be evaluated as the normalized sum of all contacts, cooperation, and communications from a to b in relation to all activities of a:

$$C(a \rightarrow b) = \frac{A(a \rightarrow b)}{\sum_{j=1}^{m} A(a \rightarrow b_j)} \tag{3}$$

where: $A(a \rightarrow b)$ – the function that denotes the activity of user a directed to user b, e.g. number of emails sent by a to b; m – the number of all users within the *SNU*. In the above formula the time is not considered. The similar approach is utilized by Valverde et al. to calculate the strength of relationships. It is established as the number of emails sent by one person to another person [23].

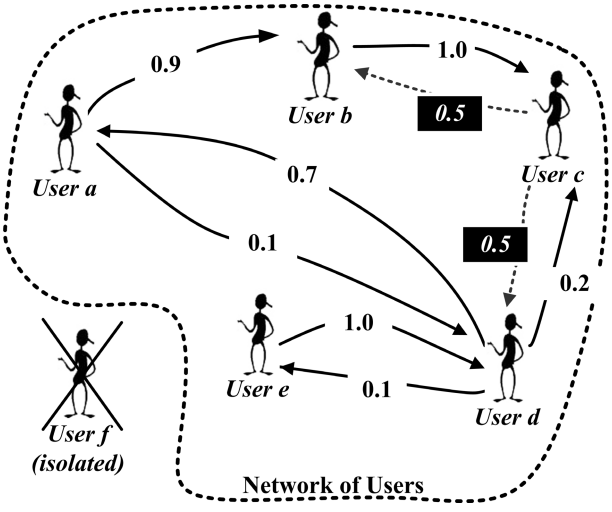


Fig. 2. Example of the social network of users with the assigned commitment values

However, the authors do not respect the general activity of the given individual. In the proposed approach, this general, local activity exists in the form of denominator in formula 3. In another version of commitment function $C(a \rightarrow b)$ all member's activities are considered with respect to their time. The entire time from the first to the last activity of any member is divided into k periods. For instance, a single period can be a month. Activities in each period are considered separately for each individual:

$$C(a \rightarrow b) = \frac{\sum_{i=0}^{k-1} (\lambda)^i A_i(a \rightarrow b)}{\sum_{j=1}^{m} \sum_{i=0}^{k-1} (\lambda)^i A_i(a \rightarrow b_j)} \tag{4}$$

where: i — the index of the period: for the most recent period $i = 0$, for the previous one: $i = 1$,..., for the earliest $i = k - 1$; $A_i(a \rightarrow b)$ — the function that denotes the activity level of user a directed to user b in the ith time period, e.g. number of emails sent by a to b in the ith period; $(\lambda)^i$ -– the exponential function that denotes the weight of the ith time period, $\lambda \in (0; 1]$; k — the number of time periods. The activity of user a is calculated in every time period and after that the appropriate weights are assigned to the particular time periods, using $(\lambda)^i$ factor. The most recent period $(\lambda)^i = (\lambda)^0 = 1$, for the previous one $(\lambda)^i = (\lambda)^1 = (\lambda)$ is not greater than 1, and for the earliest period $(\lambda)^i = (\lambda)^{k-1}$ receives the smallest value. The similar idea was used in the personalized systems to weaken older activities of recent users [16].

One of the activity types is the communication via email or instant messenger. In this case, $A_i(a \rightarrow b)$ is the number of emails that are sent from a to b in the particular period i; and $\sum_{j=1}^{m} A_i(a \rightarrow b_j)$ is the number of all emails sent by a in the ith period. If user a sent many emails to b in comparison to the number of all a's sent emails, then b has greater commitment within activities of a,

i.e. $C(a \rightarrow b)$ will have greater value than other a's neighbors. In consequence personal importance of user b will grow. However, not all of the elements can be calculated in such a simple way. Other types of activities are much more complex, e.g. comments on forums or blogs. Each forum consists of many threads where people can submit their comments. In this case, $A_i(a \rightarrow b)$ is the number of user a's comments in the threads in which b has also commented, in period i, whereas the expression $\sum_{j=1}^{m} A_i(a \rightarrow b_j)$ is the number of comments that have been made by all others on threads where a also commented, in period i.

4 Experiment on Enron Dataset

The experiments that illustrate the idea of personal importance assessment were carried out on the Enron dataset, which consists of the employees' mail boxes. Enron Corporation was the biggest energy company in the USA. It employed around 21,000 people before its bankruptcy at the end of 2001. A number of other researches have been conducted on the Enron email dataset [20], [22]. First, the data has to be cleansed by removal of bad and unification of duplicated email addresses. Additionally, only emails from within the Enron domain were left. Every email with more than one recipient was treated as $1/n$ of a regular email, where n is the number of its recipients. The general statistics related to the processed dataset are presented in Table 1.

Table 1. The statistical information for the Enron dataset

No of emails before cleansing	517,431
Period (after cleansing)	01.1999-07.2002
No. of removed distinct, bad email addresses	3,769
No. of emails after cleansing	411,869
No. of internal emails (sender and recipient from the Enron domain)	311,438
No. of external emails (sender or recipient outside Enron)	120,180
No. of distinct, cleansed email addresses	74,878
No. of isolated users	9,390
No. of distinct, cleansed email addresses from the Enron domain (social network users) without isolated members the set *UID* in *SNU*=(*UID*, *R*)	20,750
No. of network users within *UID* with no activity	15,690 (76%)
Percentage of all possible relationships	5.83%

After data preparation the commitment function is calculated for each pair of members. To evaluate relationship commitment function $C(a \rightarrow b)$ both of the presented formulas – 3 and 4 - were used. Formula 3 was utilized to calculate personal importance without respecting time (*PI*) whereas formula 4 serves to evaluate personal importance with time factor (*PIwTF*). The initial personal importance for all members was established to 1 and the stop condition was as

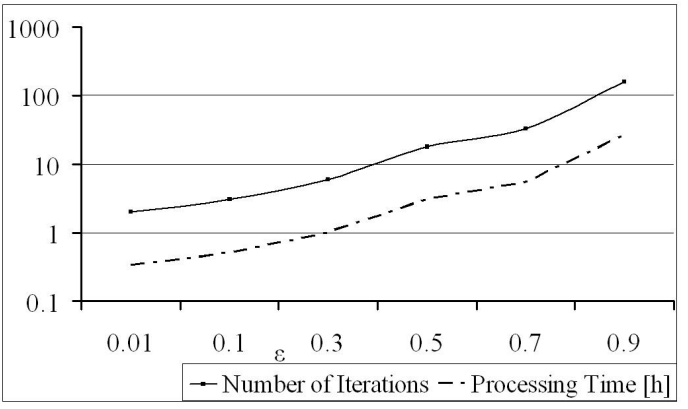


Fig. 3. The number of iterations and processing time in relation to ε

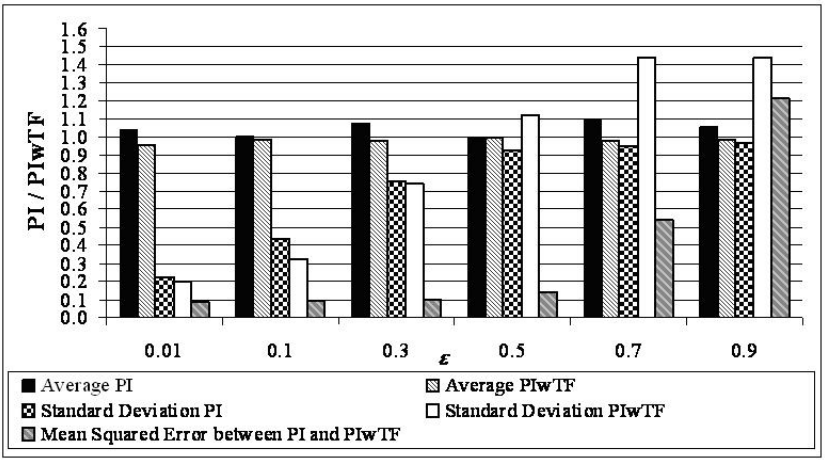


Fig. 4. Average *PI* and *PIwTF*, standard deviation of *PI* and *PIwTF*, mean squared error between *PI* and *PIwTF* calculated for different values of ε

follows $\tau = 0.00001$. The personal importance without and with time coefficient was calculated for six, different values of the ε coefficient, i.e. $\varepsilon = 0.01$, $\varepsilon = 0.1$, $\varepsilon = 0.3$, $\varepsilon = 0.5$, $\varepsilon = 0.7$, $\varepsilon = 0.9$.

The conducted case study revealed that the time necessary to calculate the personal importance for all users tightly depends on the ε value, i.e. the greater ε is the greater processing time is (Fig. 3). The similar influence has the value of ε coefficient on the number of iterations required to fulfill the stop condition. Some additional information about the values of personal importance provides the average personal importance within the *SNU* and the standard deviation of both personal importance values *PI* and *PIwTF* (Fig. 4). The average personal

Table 2. The percentage contribution of members in the Enron social network with $PI \geq 1$ and PI where time factor is not included in relation to ε

ε	0.01	0.1	0.3	0.5	0.7	0.9
$PI \geq 1$	6.973	6.973	6.188	5.494	2.251	0.906
$PI < 1$	93.027	93.027	93.812	94.506	97.749	99.094

Table 3. The percentage contribution of members in the Enron social network with $PIwTF \geq 1$ and $PIwTF$ where time factor is not included in relation to ε

ε	0.01	0.1	0.3	0.5	0.7	0.9
$PIwTF \geq 1$	5.865	4.723	4.443	4.371	4.173	0.906
$PIwTF < 1$	95.135	95.277	95.557	95.629	95.827	99.094

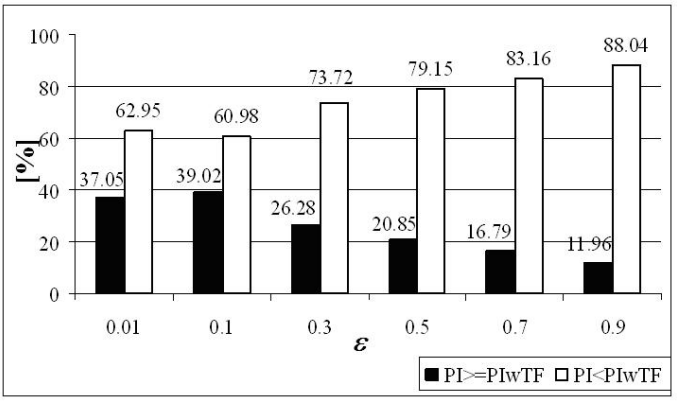


Fig. 5. The percentage contribution of members with $PI \geq PIwTF$ and $PI < PIwTF$ within the Enron social network in relation to ε

importance does not depend on the value of ε. In all cases, it equals around 1 (Fig. 4). It convergence to 1 is formally proved [19]. However, the standard deviation differs depending on the coefficient ε value. The greater ε is, the bigger standard deviation is. It shows that for greater ε the value of the distance between the members' personal importance increases, and this can be noticed for both PI and $PIwTF$. It can be noticed that the value of personal importance PI for over 93% (see also Table 2) and $PIwTF$ for over 95% (see also Table 3) of the community is less than 1 (see also Table 2). It means that only few members exceed the average value that equals 1. This confirms that personal importance can be the good measure to extract the key users in SNU [18]. The comparison of the values of PI and $PIwTF$ (Fig. 5) reveals that more users obtain higher $PIwTF$ position than PI. It means that people who have greater $PIwTF$ were more active in the latest periods. personal importance PI denotes the general position of a user regardless of time. Hence, PI will be the same for a person a

that received n emails from b three years ago and for a user c that also received n emails from b but all in the latest month. Such situation will not appear during calculation of *PIwTF*. In such case the importance of user a will be lower then of the user c, because the weight assigned to the earlier period will be lower than the weight assigned to the latest period.

5 Conclusions

Personal importance of a user in *SNU* reflects the characteristic of the user's neighbourhood. Its value for a given individual respects both personal importance of the nearest acquaintances as well as their attention directed to the considered user. Thus, *PI* measure is an important personal social feature within the community of members who communicate each other. It provides the opportunity to analyze the *SNU* with respect to social behaviours of individuals. This personal importance appears to be a powerful measure, which can be successfully used to select users for project teams [18], find new potential employees, search the consumers for advertising campaigns, recommender systems [17], and finally for use in target marketing [26].

Acknowledgements

This work was partly supported by The Polish Ministry of Science and Higher Education, grant no. N516 037 31/3708.

References

1. Adamic, L.A., Adar, E.: Friends and Neighbors on the Web. Social Networks 25(3), 211–230 (2003)
2. Bavelas, A.: Communication patterns in task – oriented groups. Journal of the Acoustical Society of America 22, 271–282 (1950)
3. Botafogo, R.A., Rivlin, E., Shneiderman, B.: Structural analysis of hypertexts: identifying hierarchies and useful metrics. ACM Transaction on Information Systems 10(2), 142–180 (1992)
4. Berkhin, A.: A Survey on PageRank Computing. Internet Mathematics 2(1), 73–120 (2005)
5. Boyd, D.M.: Friendster and Publicly Articulated Social Networking. In: CHI 2004, pp. 1279–1282. ACM Press, New York (2004)
6. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems 30(1–7), 107–117 (1998)
7. Brinkmeier, M.: PageRank Revisited. ACM Transactions on Internet Technology 6(3), 282–301 (2006)
8. Culotta, A., Bekkerman, R., McCallum, A.: Extracting social networks and contact information from email and the Web. In: CEAS 2004, First Conference on Email and Anti-Spam (2004)
9. Freeman, L.C.: Centrality in social networks: Conceptual clarification. Social Networks 1(3), 215–239 (1979)

10. Garton, L., Haythorntwaite, C., Wellman, B.: Studying Online Social Networks. Journal of Computer-Mediated Communication 3(1) (1997)
11. Gibson, D., Kleinberg, J., Raghavan, P.: Inferring Web communities from link topology. In: 9th ACM Conference on Hypertext and Hypermedia, pp. 225–234 (1998)
12. Golbeck, J., Hendler, J.A.: Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-Based Social Networks. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 116–131. Springer, Heidelberg (2004)
13. Hanneman, R., Riddle, M.: Introduction to social network methods (2006), http://faculty.ucr.edu/~hanneman/nettext/
14. Hatala, J.P.: Social Network Analysis in Human Resources Development: A New Methodology. Human Resource Development Review 5(1), 45–71 (2006)
15. Katz, L.: A new status derived from sociometrics analysis. Psychometrica 18, 39–43 (1953)
16. Kazienko, P., Adamski, M.: AdROSA - Adaptive Personalization of Web Advertising. Information Sciences 11, 2269–2295 (2007)
17. Kazienko, P., Musiał, K.: Recommendation Framework for Online Social Networks. In: AWIC 2006. Studies in Computational Intelligence, vol. 23, pp. 111–120. Springer, Heidelberg (2006)
18. Kazienko, P., Musiał, K.: On Utilizing Social Networks to Discover Representatives of Human Communities. International Journal of Intelligent Information and Database Systems (to appear, 2007)
19. Kazienko, P., Musiał, K.: Social Position of Individuals in Virtual Social Networks (to appear, 2008)
20. Priebey, C.E., Conroy, J.M., Marchette, D.J., Park, Y.: Scan Statistics on Enron Graphs. Computational & Mathematical Organization Theory 11, 229–247 (2005)
21. Rana, O.F., Hinze, A.: Trust and reputation in dynamic scientific communities. IEEE Distributed Systems Online 5(1) (2004)
22. Shetty, J., Adibi, J.: Discovering Important Nodes through Graph Entropy The Case of Enron Email Databases. In: 3rd International Workshop on Link Discovery, pp. 74–81. ACM Press, New York (2005)
23. Valverde, S., Theraulaz, G., Gautrais, J., Fourcassie, V., Sole, R.V.: Self-organization patterns in wasp and open source communities. IEEE Intelligent Systems 21(2), 36–40 (2006)
24. Wasserman, S., Faust, K.: Social network analysis: Methods and applications. Cambridge University Press, New York (1994)
25. Wellman, B., Salaff, J.: Computer Networks as Social Networks: Collaborative Work, Telework, and Virtual Community. Annual Review of Sociology 22, 213–238 (1996)
26. Yang, W.S., Dia, J.B., Cheng, H.C., Lin, H.T.: Mining Social Networks for Targeted Advertising. In: HICSS 2006, Track 6, p. 137a. IEEE Computer Society, Los Alamitos (2006)

Proofs of Communication and Its Application for Fighting Spam*

Marek Klonowski** and Tomasz Strumiński

Institute of Mathematics and Computer Science,
Wrocław University of Technology,
ul. Wybrzeże Wyspiańskiego 27
50-370 Wrocław, Poland
marek.klonowski@pwr.wroc.pl, tomasz.struminski@pwr.wroc.pl

Abstract. In this paper we present a *communicational proof-of-work* – a new tool that can be used, among others, for filtering messages and limiting spam. Our idea can be regarded as an analogue of regular proofs-of-work introduced by Dwork and Naor. The idea presented in our paper is as follows: the prover has to provide a convincing evidence that he had communicated with other, randomly chosen entity. This approach is essentially different from previous proofs-of-work, because fulfilling this requirement does not depend on resources owned by the prover (e.g. sender) only. Thanks to this, even if the adversary (e.g. spammer) has an access to much more efficient computers, he does not have any important advantage over regular, honest users of the system. We also demonstrate some other applications of the presented idea as well as some extensions based on its combination with regular proofs-of-work. Together with algorithms we also briefly describe a proof of our concept i.e. working implementation. We present some experimental data and statistics obtained during tests of our application.

1 Introduction

According to the latest statistics, as much as 85 to 95% of all e-mail messages are spam nowadays [5]. The main reason for this situation is very low sender-side cost of e-mail messages, which makes sending e-mails a perfect way of advertising. For average users it has some serious consequences. First of all, spam lowers users productivity and devalues an e-mail itself as a way of communication. Moreover spam increases costs of e-mail related infrastructure – servers, anti-spam facilities etc.

Many approaches have been investigated to tackle spam problem, but the most important (and widely used) methods for detecting and removing spam are still based on: source address filtering (blacklists, graylists and whitelists), keyword filtering (pattern matching, Bayes filters) and e-mail address hiding (confusing harvesters). Keyword filtering seems to be the most powerful among mentioned methods, but using it causes the risk of classifying a legitimate e-mail as spam. Moreover, content filtering methods needs to be developed and upgraded constantly in order to achieve a high spam

* Partially supported by EU within the 6th Framework Programme under contract 001907 (DELIS).

** Beneficiary of Domestic Grant for Young Scientists awarded by The Foundation for Polish Science.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 720–730, 2008.

detection rate. Recent outbreak of pdf spam (i.e. spam in pdf attachments) illustrates how important are up-to-date methods. For now (August 2007), filters are not prepared sufficiently to remove such spam from our mailboxes.

As an alternative for classical techniques, Dwork and Naor suggested new method that makes sending spam more expensive and time consuming [1]. The core idea is as follows: to each message an unique *proof-of-work* (POW) is attached. It is a short string which is an evidence that the sender devoted some computational resources to create a kind of *electronic stamp*. Checking validity of such a stamp should be easy for the verifier. Moreover creating reasonable number of POWs should also be accessible for regular user. However preparing bulk number of these stamps should be infeasible. This idea can be realized for example by using simple, one-way hash function.

In this paper we extend an idea of POW. Our approach is essentially different from previously presented proofs-of-work, because fulfilling its requirement does not depend on resources owned by the prover (e.g. sender) only. Namely the sender has to prove that he has communicated with particular entity designated in pseudorandom manner. Similarly as in regular proofs-of-work, content of the message as well as some metadata like actual time and receiver's identity is taken into account during pointing the entity in order to ensure that previously created POW cannot be re-used for other messages. The implementation that we are going to present below is based on downloading particular data from publicly accessible servers.

1.1 Organization of the Paper

In section 2 we briefly outline previous concepts. In particular we recall how regular POWs can be used for fighting spam. We also mention other applications of POWs. In section 3 we present the idea of proofs-of-communication outlined above. Section 4 is devoted for presenting the implementation of our concept as well as short analysis of experimental results. We point out possible threats and drawbacks of proofs-of-communication in section 5. In section 6 the promising idea of realization proof-of-communication system in P2P networks is shortly presented. We conclude in section 7.

2 Previous Works

In 1992 Dwork and Naor proposed first *proof-of-work* scheme for controlling access to a shared resource [1] based on computation of a *pricing function*. Authors suggested several pricing functions with different computational complexity: extracting square roots modulo prime number, function based on the Fiat-Shamir signature scheme, and on the Ong-Shnorr-Shamir signature scheme.

Independently, Back described and implemented the *Hashcash* – the system for making a computational proof of works based on *partial collisions* of the SHA-1 hash function [2]. The idea of spending processor time to prove a good intention of the sender was the same as presented in Dwork and Naor article.

An important property of all proof-of-work protocols is that they provide task that is relatively difficult to solve, but it is easy to verify the correctness of its solution. Computational proof-of-works can be computed completely off-line, however the time needed

to accomplish the computation hardly depends on processing power. To address this problem, Dwork, Goldberg and Naor proposed usage of a *memory-bound functions* [3]. To efficiently compute values of these functions one needs also memory as a resource. For that reason it is not possible to produce such POWs having fast processor only.

There were also doubts if proof-of-work systems can help fighting the spam in real world scenarios[9]. Laurie and Clayton estimated how long should a creation of a proof take to have a noticeable impact on spammers' profits. Their calculations led to conclusion, that the time needed for making such a proof is unacceptable for honest users [9].

2.1 Proofs-of-Work for Fighting Spam

To show how proof-of-works are computed let us recall Hashcash system [2]. To send a message $message$ at time $time$ to destination $recipient$, the sender needs to find k such that the l most significant bits of $y = \texttt{SHA-1}(time||message||recipient||k)$ are zeros. The easiest way to do this is to choose the random k and check if l most significant bits of y are in fact zeros. The pseudocode of this procedure is shown below.

```
// computing a Hashcash proof-of-work
//     l is a parameter indicating the hardness of this proof-of-work
function ComputeProofOfWork(message, recipient, time) : k;
begin
  do
    trial := Random();
    hashed := SHA-1(time||message||recipient||trial);
    zeros := CountMostSignificantZeros(hashed);
  while (zeros < l);
  return trial;
end;
```

When following this procedure, the average number of random tries for finding appropriate k is 2^{l-1}. After successful computation, k is attached to an e-mail as a proof-of-work. The recipient makes usage of attached proof k and computes the hash function only once to check if the l most significant bits are zeros in fact. If the proof is correct, the e-mail message is found to be legitimate.

2.2 Other Applications of POWs

Except fighting spam Proofs-of-work have several other applications. Among others they can be used for preventing denial-of-service attacks [7], providing incentives in peer-to-peer systems [6] and metering visits to websites [8]. Generally, access control provided by a POW systems can be used in other situations in which it is important to limit access to shared resources or services.

2.3 Disadvantagies of Computional Proofs-of-Work

The main idea of using proof-of-works for fighting spam is to increase the reliability of legitimate e-mail by attaching them with an evidence of using sender's resources. Up

to now, the processor time was the only resource considered to be useful for making such an evidence and thus, all proof-of-works developed so far can be described as *computational proof-of-works*.

However this kind of proof delays sending e-mail messages for average user, it can be computed faster if only spammer decided to use a better processor. It is one-time investment and it may guarantee him that the future effort of computing proof-of-work will be only slightly increased (compared to sending e-mails without attached proof).

3 Proofs-of-Communication

In this section we introduce *proofs-of-communication* or POC, for short. Instead of computing a moderate hard function [1], a sender is required to process the communication with randomly chosen entity for proving his good intention. In this case the resource used for making a proof is an *access to information*, i.e. access to some data that can be retrieved from the Internet (or other Network). The creation of those proofs is constrained by both sender's **and** data owner's throughput (i.e. throughput of Internet connections), so unlike computational POWs, proofs of communication do not depend only on resource under sender sole control. We believe this property helps with fighting spam, because a simple increase of spammer's throughput will not help him with faster proofs creation.

There are several more advantages of using proofs-of-communication: they do not depend on processor speed and they can be combined with other proof-of-works (i.e. with computational POW). On the other hand, the communication overhead is the most important drawback of POC systems. Potential disadvantages of POC are discussed in section 5.

It is needed to mention that usage of POC apart will not rather be a solution for a spam problem in a real world. Instead, we find POC to be a tool, that combined with other antispam methods may significantly help fighting spam.

3.1 Requirements

Proof-of-communication have to meet several requirements analogical to those defined for regular proofs-of-work in [1]. In particular

1. for a given e-mail it should be moderately hard to create a POC,
2. with given POC and an e-mail it should be very easy to check if POC is correct,
3. any preprocessing should not essentially help to create POC.

Moreover proof-of-communication should meet some additional requirements specific for POCs:

low traffic generation – additional communication load caused by creating POCs should be negligible for all parties operating in the system except the POC creator.

dynamic content tolerance – if creation and verification of POC is based on downloading particular data, proposed solution should take into account that some content can be dynamically changed in the period between POC creation and its verification.

no dedicated infrastructure – solutions should be based only on existing servers and communicational patterns.

3.2 Internet/HTTP-Based POC

In this section we present a simple realization of proof-of-communication dedicated for the Internet and based on HTTP protocol. The idea is as follows: from prepared e-mail body, timestamp, recipient's and sender's address sender generates a list of webpage locations and after transfering all documents from these addresses, he calculates a kind of digest from them. The digest is then attached to an e-mail as a proof-of-communication. The verifier checks the attached proof following the sender's procedure, with the exception that he does not transfer all the documents from generated locations but only its randomly chosen, relatively small subset.

The description above is still not quite exact, but it shows the general concept and leads to following actions that will be investigated more precisely later:

- generating a random list of webpage locations for particular e-mail,
- making a digest from transfered documents,
- verifying proof attached to an e-mail.

3.3 Procedures

Location Generation. Firstly, we need to generate a list of locations from the e-mail body, sender's and recipient's addresses and a timestamp. We must assure, that this locations are generated in unpredictable for the sender manner, i.e. manipulating e-mail body etc. does not enable sender to get a list of locations that he wants. For that reason collision–free hash function is used. In the next step we must generate a concrete location from series of pseudorandom bytes. One of possible solutions is to map pseudorandom bits into subset of words from simple, commonly used dictionary and than use a web page indexing service (search engine). Careful design of query should results in millions of possible web locations. To generate a list of locations, we simply iterate hashing function on a concatenated input.

A pseudocode of generating locations list is as follows:

```
// generating web locations list from an e-mail
function GenerateLocations(message, sender, recipient, time, n) : locations;
begin
i := 0;
byteSequence := time||message||recipient||sender;
  do
     byteSequence := Hash(byteSequence);
     word := dictionary[byteSequence mod dictionarySize];
     additionalData := ExtractData(byteSequence);
     query := MakeQuery(word, additionalData);

     // get first location returned by indexing service
     locations[i] := SearchService(query);
     i := i + 1;
  while (i < n);
  return locations;
end;
```

Careful preparation of the right parameters in this procedure seems to be a key point in POC-system design. Some details can be found in the description of our implementation in section 4.

Preparing Proofs. After transferring resources from generated locations, the next important step is to make a short digest, that would be attached to an e-mail as a proof. This digest is made using all transfered resources. On the other hand system must guarantee flexibility in the sense that attached proof would be accepted with significant probability, even if some of designated resources have changed in the period between POC creation and its verification.

This problem can be solved using some small range hash functions (e.g. regular hash function with the range truncated to only few most significant bytes). After hashing every transfered resource, we simply concatenate them into one list of bytes.

A pseudocode for this solution is shown below:

```
// makes proof from transfered resources
function MakeProof(resources) : proof;
begin
  i:=0;
  do
     proof := Concatenate(proof, Hash(resources[i]);
     i := i + 1;
  while (i < resourcesNumber);
  return proof;
end;
```

Verifing POC. An efficient verification procedure is essential for every proof-of-work system as well as for proof-of-communication. To provide efficiency as well as high level of correctness we assume that the verifier downloads only random subset of the content pointed by the hash function. This trick is similar to approach for verifying MIX-servers behavior described in [4].

Below we show a pseudocode for verifying proofs-of-communication.

```
// verifies the given proof, checks only subsetSize resources,
//  for completly correct proof returns 1
function Verify(proof, subsetSize, message, sender, recipient, time, n) : real;
begin
  i := 0;
  correctParts := 0;
  locations := GenerateLocations(message, sender, recipient, time, n);
  // generate a random subset of 0, ..., n-1
  subset := GenerateSubset(0, n, subsetSize);
  do
     // proof[n] returns a n-th part of proof
     if (proof[i] == Hash(resources[subset[i]])
        correctParts := correctParts + 1;
```

```
  while (i < n);
  return (correctParts / subsetSize);
end;
```

Assume that sender wants to cheat and does not transfer all resources needed to make a proof. Let f be the number of resources' digest forged by the sender, k be the number of resources which correctness is checked by the verifier and n be the number of all resources used for making a proof, then the probability of founding the forgery is

$$Pr(\text{forgery found}) = 1 - Pr(\text{forgery not found}) = \\ = 1 - \binom{n-f}{k} \Big/ \binom{n}{k} = 1 - \frac{(n-f)!(n-k)!}{n!(n-k-f)!}. \quad (1)$$

Using the equation above, we can easily calculate the number of resources that should be checked by the verifier, so that cheating would be unprofitable for sender and the communication effort for the verifier would be still low.

If we need a verification procedure that returns value from the interval [0, 1], the correctness indicator can be used - i.e. the ratio of correct answers. However, sometimes we may need verification that returns *true* for correct proof and *false* when attached proof should be treated as incorrect. In this situation the threshold value needs to be determined. It cannot be too low, because spammers may forge proofs, but also cannot be too high, because some correct proofs may rely on higher then average numbers of dynamic pages. The concrete threshold value would depend on proof parts count and the number of parts checked in verification procedure. In our experiments proofs were composed from 20 parts and during the verification we randomly checked 5 of them. We agreed, that in this situation, all proofs with correctness indicator greater or equal to 0.4 should be treated as correct.

4 Proof of Concept

To show that proofs-of-communication may work even in a presented, simple form, we implement working system in Java. Our system consists of two packages: one for creating and verifying proofs-of-communication in general, and the second one for sending and receiving e-mails with attached proofs.

Our implementation covers exactly the idea presented above, but for precision we list some specific details of the application below:

- we have chosen *SHA-1* as a hash function; when we need a hash function with a small range, we take only few most significant bits from the output of SHA-1 (depended on contex in which we used hash function, e.g. while generating queries for Google we used SHA-1 truncated to 40 most significant bits),
- as a indexer service we have chosen *Google*, because it has indexed the amount of unique resources, which guarantee impossibility of collecting them by spammers; we have made usage of Google's "I'm feeling lucky" option, which redirects the request to the first web page found,
- queries for the indexer service are generated using a word from an English dictionary and the pseudorandom number (the pseudorandom number was obtained from hashed e-mail),
- to limit the time needed for creating POC, we limit the size of any resource to 2MB.

4.1 Experimental Results

Experiments with working implementation of our idea results in some interesting observations. Although obtained results refer to our particular proposal and not to proof-of-communication in general, they can be instructive for further models.

Stability of Webpage Positions. We silently assumed that positions of webpages (or other resources) would not change between creating and verifying proofs. In other words, to make a verifiable proofs we need our indexer service to return the same locations for certain words for at least, say 24 hours (assumed time between sending and receiving an e-mail).

To check if our assumption was correct, we decided to check the stability of webpage positions in Google. The testing procedure was as follows:

1. we generated a random e-mail message M,
2. for message M we computed and collected 2,000 webpage locations using procedure GenerateLocations described before,
3. after 24 hours, we once again computed 2,000 webpage locations for message M.

Our test show that 1976 (it is about 98.8%) webpage locations was the same as generated from the same words 24 hours before. It means that the stability of webpage positions is high enough to use it in proof-of-communication system.

Dynamic Pages. As shown above, changes in webpage positions have not high influence on a proof-of-communication repetitiveness. Dynamic pages seem to be a much greater problem. In order to create a verifiable proof, we need to ensure, that any resource transfered from generated locations two times would contain the same data. Nowadays, when almost every webpage has some dynamic content, it can be a real issue.

To measure the repetitiveness of generated proofs we made usage of *correctness indicator*, i.e. the number of correct parts of proof divided by the total numbers of parts. We repeated the following procedure 500 times

1. we generated a random e-mail message M,
2. we obtained a proof-of-communication POC_1 for the message M following all mentioned procedures,
3. once again we followed all procedures in order to get proof POC_2,
4. we assumed POC_1 to be a model proof and we computed the correctness indicator for POC_2.

We built a histogram of repetitiveness distribution using computed correctness indicators, which is shown in figure 1. The average repetitiveness for proofs-of-communication based on investigated samples was 0.67 (on average 67% of again generated proofs was the same as generated previously), while the minimal repetitivness was 0.2 (for some proof POC_2 matched with POC_1 in 20% only). It is worth to mention that, the queries used to generate the webpage location were built exactly in a way described in the previous section.

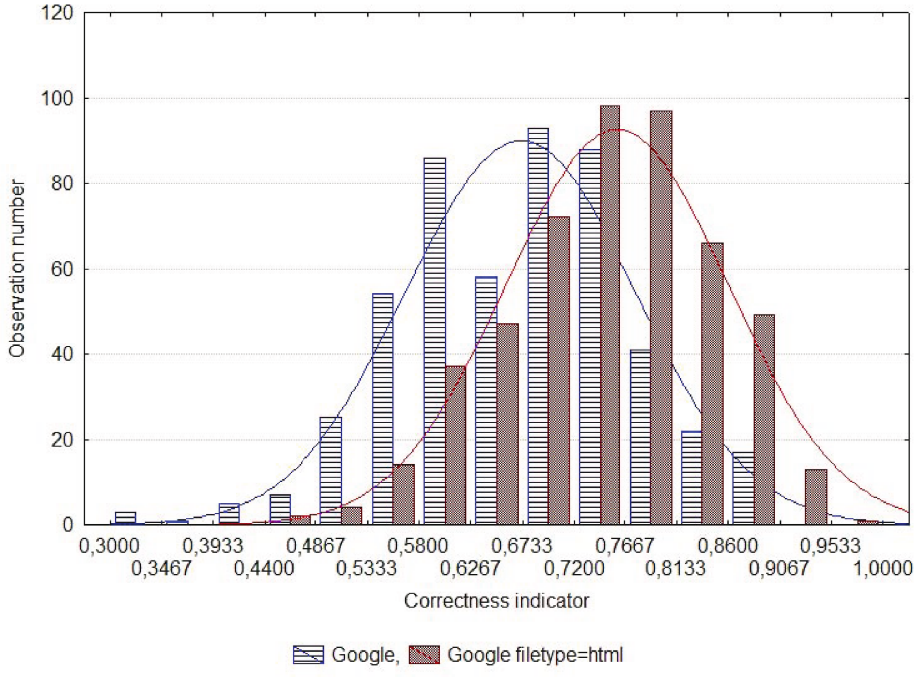


Fig. 1. Distribution histogram of the proof-of-communication repetitiveness

The low repetitiveness of generated proofs is caused by the dynamic contents of webpages. In order to improve the situation, we decided to add a file format filter to generated queries, which would constrain the search results to ”html” files only. Then we repeated the whole repetitiveness test. The average repetitiveness has grown up to 0.76. What is more, the minimal repetitiveness was as high as 0.45, which is a far better result then in the previous test. The histogram of repetitiveness distribution for proofs with extended queries is shown in figure 1.

5 Potential Threats and Drawbacks

The usage of proof-of-communication system is obviously connected with a communication overhead. For systems, in which lots of proofs would be created, the increase of network traffic can become a real issue. What is more, there are some potential situations in which the creation of proofs-of-communication would be too expensive for a regular user (e.g. when user has to pay for the amount of data transfered from Network).

The other threat for proof-of-communication users is a possibility of DoS attack. Assume that the attacker sends hundreds of e-mails with faked proofs and the receiver checks every one of them. If the verification requires significant amount of computation, it can slow down or even completely block receiver’s network connection. In our particular proposal situation can be even worse – the high number of faked proofs results in the high number of queries for Google and that may leads to a ban from Google

web search. In order to protect from DoS attacks, we suggest combining proofs-of-communication with computational proofs-of-work – i.e. only if POW is valid than POC is taken into account. Thanks to this trick, the adversary is not able to to force his victim to verify number of POC so easily.

6 Proofs-of-Communication in P2P Networks

Proofs-of-communication seems to be very useful in P2P-networks. Since operations like searching, transferring and consistency checking are already implemented for P2P-networks, only small changes in protocols and clients' software would be needed.

Some improvements we can achieve when using P2P-networks for creating POC instead of HTTP are as follows:

- higher efficiency for verification of POC; we assume, that in P2P-networks we can ask resource's owners about its digest,
- easier generating of resources' addresses; In particular in DHT paradigm (like Chord or CAN), instead of using indexing service, we could make usage of searching for resource with given hash,
- higher repetitiveness of POC; resources in P2P-networks are more static then webpages – they may appear or disappear but they rather are not changed,
- more fair load balance; instead of utilizing webpage servers, the real clients are involved in creating of POC in P2P-networks.

7 Conclusions

We described a new kind of proving "electronic efforts" and proposed it as a tool for fighting spam. In this proposal the prover (spammer) have limited possibilities of taking advantages over regular users, since creating of proof does not entirely depend on resources owned by him. This property can make spamming even more unprofitable than in case of using computational proof-of-works. Of course, the presented idea have some drawbacks. However we believe that this tool can be very efficient and powerful when combined with other mechanisms e.g. regular computational proof-of-works. We have also pointed out some other possible applications for POCs. Especially using them in P2P networks seems to be very promising.

References

1. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, p. 139. Springer, Heidelberg (1993)
2. Back, A.: The Hashcash Proof-of-Work Function Network Working Group, INTERNET-DRAFT (2003), http://hashcash.org
3. Dwork, C., Goldberg, A., Naor, M.: On Memory-Bound Functions for Fighting spam. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 426–444. Springer, Heidelberg (2003)
4. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. In: Proceedings of the 11th USENIX Security Symposium, pp. 339–353 (2002)

5. Spam-o-meter Statistics By Percentage: Spam statistics, http://www.spam-o-meter.com
6. Serjantov, A., Lewis, S.: Puzzles in P2P Systems. In: 8th CaberNet Radicals Workshop, Corsica (2003)
7. Mankins, D., Krishnan, R., Boyd, C., Zao, J., Frentz, M.: Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing. In: ACSAS 2001. Proceedings of 17th Annual Computer Security Applications Conference (2001)
8. Franklin, M.K., Malkhi, D.: Auditable Metering with Lightweight Security. In: FC 1997. LNCS, vol. 1318, pp. 151–160. Springer, Heidelberg (1997)
9. Laurie, B., Clayton, R.: "Proof-of-Work" Proves Not to Work. In: WEIS 2004. Proceedings of the Workshop on Economics and Information Security (2004), http://www.cl.cam.ac.uk/%7Ernc1/proofwork.pdf

Web Pages Reordering and Clustering Based on Web Patterns

Miloš Kudělka[1], Václav Snášel[1], Ondřej Lehečka[1],
Eyas El-Qawasmeh[2], and Jaroslav Pokorný[3]

[1] Department of Computer Science, VŠB – Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{milos.kudelka, vaclav.snasel, ondrej.lehecka}@vsb.cz
[2] Computer Science Dept., Jordan University of Science and Technology,
Irbid, Jordan
eyas@just.edu.jo
[3] Depatment of Software Engineering, Charles University of Prague,
Czech Republic
pokorny@ksi.ms.mff.cuni.cz

Abstract. In this paper was proposed a method for the description of web pages using web patterns. We will explain what we mean by the term "web pattern". We will present a taxonomy web patterns and a description of some their types. In the description of web patterns we will focus on properties which are useful for automatic detection on web pages. As a result of the detection we get a description of a web page using found web patterns. The description can be used for reordering and clustering of a web page set.

1 Introduction

One way to assist the user, in orientation within a big amount of non-structured data, is clustering according to common key properties. The biggest problem, however, remains in the definition what is the key property which is useful for definition of similarity [12]. In our approach, we work with web pages and web patterns which are presented on these web pages. The web patterns provide, on a certain level, a formalized mechanism for the description of common features of an object which is commonly visible on web pages. For this purpose we have developed some new web patterns, extended web pattern description, and created their taxonomy. The obvious assumption of web page availability is its presence in search engines indexes. The presence of the web page in the index does not mean that the page is always available through simple query. Different methods are used for measuring the relevance of a web page against the query. Considering the huge amount of web pages, it is obvious that current methods are not sufficient. One important feature of our approach is that it provides new information about web pages which is not currently used, but the information is readable for users (users understand it). Using this information, it is possible to provide additional criteria for gauging the relevance of the web page, and for comparing these

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 731–742, 2008.

criteria to user expectations. Our tests and experiments with users [10,18,11] prove the presented approach can assist with orientation of a large amount of nonstructural data. If we can reveal which web patterns a web page contains, user can immediately create a notion of the kind of information they can expect (see Fig. 1). Web patterns are detected using analysis and segmentation of text content [10]. The entire process is displayed in Fig. 2.

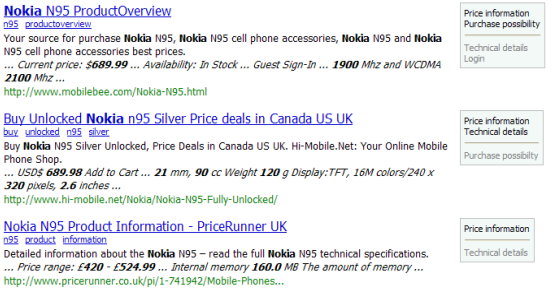


Fig. 1. A sample of three searched pages on the query "nokia" in our experimental search engine www.pattrio.net. There is a "tie-on label" on the right side of each web page. The label contains a list of automatically detected web patterns (font weight was used to designate the relevance of the detection; bold font is designating high relevance). Web page snippets contain best segments from the web patterns found on the page. The segments are highlighted in italics.

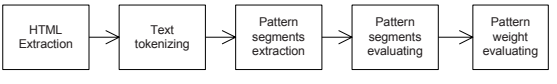


Fig. 2. Web pattern detection process

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 introduces the web patterns basics. Section 4 presents samples of web patterns from our pattern catalogue. In section 5, we describe page similarity based on web patterns. Section 6 and 7 contain a description of experiments and a results analysis. Section 8 is our conclusion.

2 Related Work

In the field of non-structured data analysis there are various approaches (for example information retrieval, semantic web, information extraction ...). The construction of web information extraction systems is an approach similar to ours. There web pages are transformed into program-friendly structures such as a relational database. In our approach we need to analyze the structure and templates of the web page. The survey of major web data extraction approaches is presented in [3]. An interesting approach is mentioned in [12]. By empirically

studying web pages across web sites about the same type of objects, the authors found out many HTML template independent features. In [13] there is presented a method for extracting objects from the web. The authors have written that the main challenge for extraction is that objects of the same type are distributed among diverse web sources, whose structures are highly heterogeneous. In this point our method is related to theirs. The next approach related to ours is described in [21]. The authors propose a vision-based page segmentation algorithm to detect the semantic content structure in a web page. The next similar approach is mentioned in [8]. It analyzes chosen parts of pages to obtain structured domain specific information (tourism domain). Other similar approaches are based on an automatic transformation of arbitrary table-like structures into knowledge models [14] or domain-oriented approach to web data extraction based on a tree structure analysis [15]. There is an interesting conjunction with the paper [7]. Its authors analyzed web pages focusing on web site patterns. In three different time intervals they observed how web designers have changed web design practices. They also realized that the content of web pages remains the same whereas a form is being developed so it better fulfils users' expectation. Our work confirms results mentioned in the paper. For us such characteristics of web patterns that are independent of the web page design are very important.

3 Web Pattern

In this section, we first introduce the concepts of a web pattern and pattern extraction. By the term "web pattern" we mean any high-level object which is on a certain (relatively high) level of abstraction presented often and repeatedly on web pages. We especially consider the web pattern only such an object which can be named, so it is clear from the name what the pattern describes (both users and developers should agree on this). Examples of such web patterns are the Price information pattern or the Purchase possibility pattern (see their descriptions below). These web patterns come along with pages about selling products or services. Both mentioned web patterns provide semantic information to the user. This is one of the key features. There are also web patterns which do not contain any semantic content. An example of such a web pattern can be the Something to read pattern describing pages which contain a bigger amount of text. Another example is the Link list pattern which describes pages containing a group of links. Descriptions of those web patterns can be also found below. It is very important to state, for the purpose of this paper, that we only work with those web patterns which can be automatically found on a web page based on their description with a relatively high relevance. For example, the Welcome page can also be considered a web pattern. However, the detection of this pattern would be much more difficult (or even impossible due to the designer creativity).

3.1 Web Pattern Taxonomy

We divide web patterns into two groups (many details about patterns and pattern classification are in [4]). In first group there, are web patterns providing

semantic information to the user (Domain patterns). This information is connected to a certain domain which the page and expected users belong to. The domain is considered to be composed by web pages with specific content, and users with specific requirements and expectations. The selling product and services domain, tourism domain, culture domain, newscast domain, etc., can serve as examples of domains. The advantage of mentioned domains is that there are a lot of patterns described in domain catalogs [6,19]. Some web patterns which we use in our approach are adopted from these catalogs. The second group contains web patterns independent of a domain. They describe more structural and technical features of typical solutions (Structural patterns).

3.2 Pattern Description and Structure

Patterns are designated for users (web designers in this case) who work with them and use them in production. A pattern description is composed from parts and each part describes a specific pattern feature. Authors usually use the pattern structure introduced in [1]. In the description there is a pattern name, problem description, context, solution and examples of use. Usually, these are also consequences of the use of the pattern and related patterns which relate somehow with the pattern being used. For our description we use the section-oriented structure originated by Kent Beck (*http://c2.com/cgi/wiki?BeckForm*). There is also a Forces section describing details which can help in the automatic detection of web pattern on a web page. The description of such details is derived from our experiments with web pattern detection in a large amount of web pages. This description also helps us to understand how to design detecting algorithms. The example descriptions and examples themselves, presented in a Solution section show how different designers can proceed in the implementation of pages. In the following text, we will describe some web patterns.

Title: An appropriate pattern name.
Problem: A single brief sentence describing the problem which pattern solves.
Context: A list of situations where the pattern occurs.
Forces: A list of details which influence the pattern identification. We are focusing especially on features useful for automatic detection.
Solution: Description of the solution with examples.

The related patterns are also very important. If there is such a pattern name in the description, we highlight it in italic for clarity.

4 Web Pattern Samples

We choose patterns from a collection (or corpus) C which we use for automatic detection on web pages. We use more than twenty web patterns (domain and structural) for the analysis. We choose three domain and two structural patterns.

4.1 Domain Patterns

Price Information
Problem: How can sales information be displayed to the user, graphically?
Context: Selling products, services, etc.
Forces: A lone page fragment is usually bound to a small space. Keywords are used to label a price. Numbers are used with currency symbols. A picture is displayed near the product description on the page. All mentioned elements are displayed within a small space for a current product offer.
Solution: In different contexts there should be more different implementations. It is the case of pages with a single offer or pages with more offers (a catalogue with offers). The patterns may occur at a page border as an advert. The pattern is usually present with patterns Purchase possibility, Special offer and Repayment. See Fig. 4(a) (left).

Technical Details
Problem: How can a product's technical details be shown lucidly?
Context: Selling products like electronics, appliances (fridge, etc.). Personal website (for example a product fan). Manufacturer's website.
Forces: A page fragment with a headline and a list of single rows describing product parameters. Key words labeling details section on the page (details, parameters...). Keywords labeling parameters (size, weight, frequency...). Numbers in combination with unit s symbols (cm, kg, MHz...). All mentioned pattern elements are placed on a larger section of the page so the user can read them continuously.
Solution: Usually an implementation using a table layout (or similar technology leading to the same-looking result) is used. If the pattern is on a selling product website there are usually Purchase possibility, Selling information patterns and often Login pattern. See Fig. 4(a) (right).

Discussion (Forum)
Problem: How can a discussion about a certain topic be held? How can a summary of comments and opinions be displayed?
Context: Social field, community sites, blogs, etc. Discussions about products and service sales. Review discussions. News story discussion.
Forces: A page fragment with a headline and repeating segments containing individual comments. Keywords to labeling discussion on the page (discussion, forum, re, author,...). Keywords to labeling persons (first names, nicknames). Date and time. There may be a form to enter a new comment. Segments with the discussion contributions are similar to the mentioned elements view, in form.
Solution: Usually, an implementation using a table layout with an indentation for replies (or similar technology leading to the same-looking result) is used. The pattern is often together with the Login pattern. If the pattern is on a selling product website there are usually Purchase possibility, Price information patterns. The pattern can be alone on the page. In other case there is also the Something to read pattern. In different domains the pattern can be displayed with patterns Review, News, etc. See Fig. 4(c).

4.2 Structural Patterns

Something to Read
Problem: How can text be lucidly displayed on the web page?
Context: The pattern is used often and regardless of the domain.
Forces: A fragment occupying almost a whole page. There are usually longer, continuous paragraphs. If the text is long there can be a short heading among some paragraphs. Within paragraphs, there may be a few links or images.
Solution: It is simply implemented using line spacing and headings of a certain level. See Fig. 4(c).

Link List
Problem: How to lucidly show list of links to related pages?
Context: The pattern is used often and regardless of the domain.
Forces: The page fragment with links. Each link is usually in the form of intelligible text within the scope of single sentence (few words). Each link can be appended with a short text or URL address. Each link is on a single row.
Solution: There is usually implementation with single continuing rows or a similar strategy leading to the same-looking result (for example using enumerations, lists, table layouts). See Fig. 4(d).

5 Page Similarity

5.1 Human View

With the assumption that it is possible to detect web patterns automatically on the web pages, it is possible to describe each page with the patterns. Using pattern names the description may look like this example: "The page contains the Price information, the Purchase possibility and the Special offer. There are also Technical details and the Discussion at the bottom." Using such a description, the user does not know which product is presented on the page. However, the user, as well as the page designer, can imagine how the page looks. So the group of patterns characterizes a relatively wide set of pages which has a similar purpose (and belongs to the same domain). We call such a group of patterns a page profile.

5.2 Technical View

Concerning pattern detection, in the paper [10] we provided a general description of algorithm based on Gestalt principles (proximity, closure, similarity, continuity [19]). The algorithm can detect domain patterns on web pages with a high relevance (about 80%). As described above the algorithm makes the content segmentation and segment extraction of a web page, segment evaluation and evaluation of relevance (weight) of the found pattern. In page segmentation we work with dynamic generated dictionaries of patterns containing frequently

used words and data types in the pattern context. In the papers [5,20] was proposed algorithm for identification of design patterns as part of the reengineering process. The identification of implemented design patterns could be useful for the comprehension of an existing design and provides the ground for further improvements. This idea is very similar to our approach. For the detection of structural patterns we use specialized algorithms. In the HTML code extraction phase, we preserve only few elements for example links and paragraphs. These elements are foundations for algorithms design. Page representation is in the vector model [16,17]. There a document (web page) $W_j \in W$ (a collection of web pages) is represented as a vector w_j of pattern weights, which record the extent of importance of the pattern for the web page. To portrait the vector model, we usually use an $n \times m$ pattern-by-page matrix A, having n rows pattern vectors $p_1, \ldots, p_n$ where n is the total number of patterns in collection C and m columns page vectors $w_1, \ldots, w_m$, where m is the size of collection W. The dimension n is never too big. The source for patterns is a catalogue C. These catalogues are not too large. However, the web patterns are not described with regard to automatic detection. This is a reason why we have our own catalogue and describe web patterns according to the patterns described in our paper. Weights of patterns are obtained as the result of the pattern detection algorithm application. Such a value determines the level of certainty whether a searched pattern was detected on th e web page (a pattern weight). The similarity between two web page vectors w_j and w_k can be determined by computing the cosine of angle between them:

$$Sim(w_i, w_j) = \frac{w_i \cdot w_j}{||w_i|| \cdot ||w_j||}$$

6 Experiment: Reordering

We wanted to prove our approach with a chosen set of users. We were interested in whether the automatic web pattern detection on web pages may be useful for the users. It may help then with orientation while searching through a vast amount of web pages. We designed an experimental web interface which is similar to a common search engine (www.pattrio.net). In the interface the user could write a query in a common way. For the purpose of this experiment we use data provided by the Czech search engine www.jyxo.cz. After that the search engine returned a set of 100 pages including page text content. The set was analyzed. For each web page the vector was computed which represents the page and which aggregated the best segments from found web patterns. Then the page set was displayed to the users in the original order from the search engine. The vector (page profile) was displayed as a tie-on label (see Figure 1). Altogether we used 13 domain and 5 structural patterns. By clicking on the tie-on label the result set would be reordered according to the profile of the selected page and less according to the original order. The selected page was on the first position in the result set and the least similar page was moved to the end of the set. In this reordering there were many pages from the second half of the original order moved to the first ten of pages. The users (students and teachers) were instructed that by clicking on the tie-on label of the web page the result set will

be reordered. The interaction between the user and the result set was completely recorded for further analysis. The users were asked to answer three questions. The answer could be only Yes or No. There were 34 users answering.

1. Is the information about the searched page which was being displayed useful for selecting the page? There were richer snippets below the title and the tie-on label on the right side.
2. Does the displayed information correspond with the page content?
3. Is the reordering, according to the tie-on label, helpful in searching?

We got 21 Yes answers on the first question. For the second question there were 34 users answering Yes with one note. The note is that what was written on the tie-on label was mostly O.K. but some web patterns presented on the page were sometimes not detected. For the last questions we got 26 positive answers with a note that the reordering was useful but only for pages where web patterns were detected. The most common comments were:

1. It is good that the pages which I am not interested in at the moment are moved to the end of the result set.
2. There are too many pages with no web pattern detected and the reordering according to their profile does not bring the expected result (in the sense of similarity with selected page).
3. The pages are not always ordered as I expected.
4. Sometimes the result set is reordered toward product selling but I get pages about a different product than I wanted.

The third criticism relates with use of cosine measure. The best results can not always be reached with this measure.

According to the last point we added links with words from a title below each title. By clicking on the link the word was added to the query (see Figure 1). The idea that the title usually contains words important for the page content can be considered as a web pattern.

7 Experiment: Profiles and Clusters

In the previous experiment we worked with a profile of each page in the reordering of the result set. The goal of the second experiment was to find whether some static or common profiles exist. The profiles can be used for reordering of the resulting set. Intuition suggests that there should be profiles for internet shops, advertising and forums for sharing experiences For this experiment we used the recorded pages from querying products. The analysis was performed on 23,422 pages. Each page contained at least one domain pattern from the selling products domain (altogether there was 9 domain patterns). We wanted to visualize the analysis results so, we used the SOM method. The self-organizing map (SOM) is a type of neural network. This network is trained using unsupervised learning and produces low dimensional representation of the training samples while preserving the topological properties of the input space. The model is also

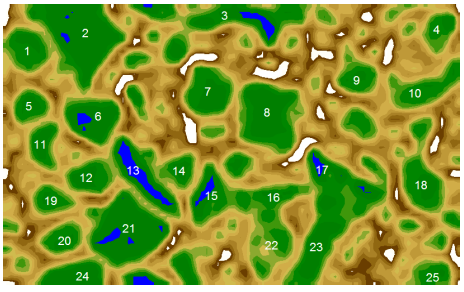


Fig. 3. SOM — web pages from selling product domain

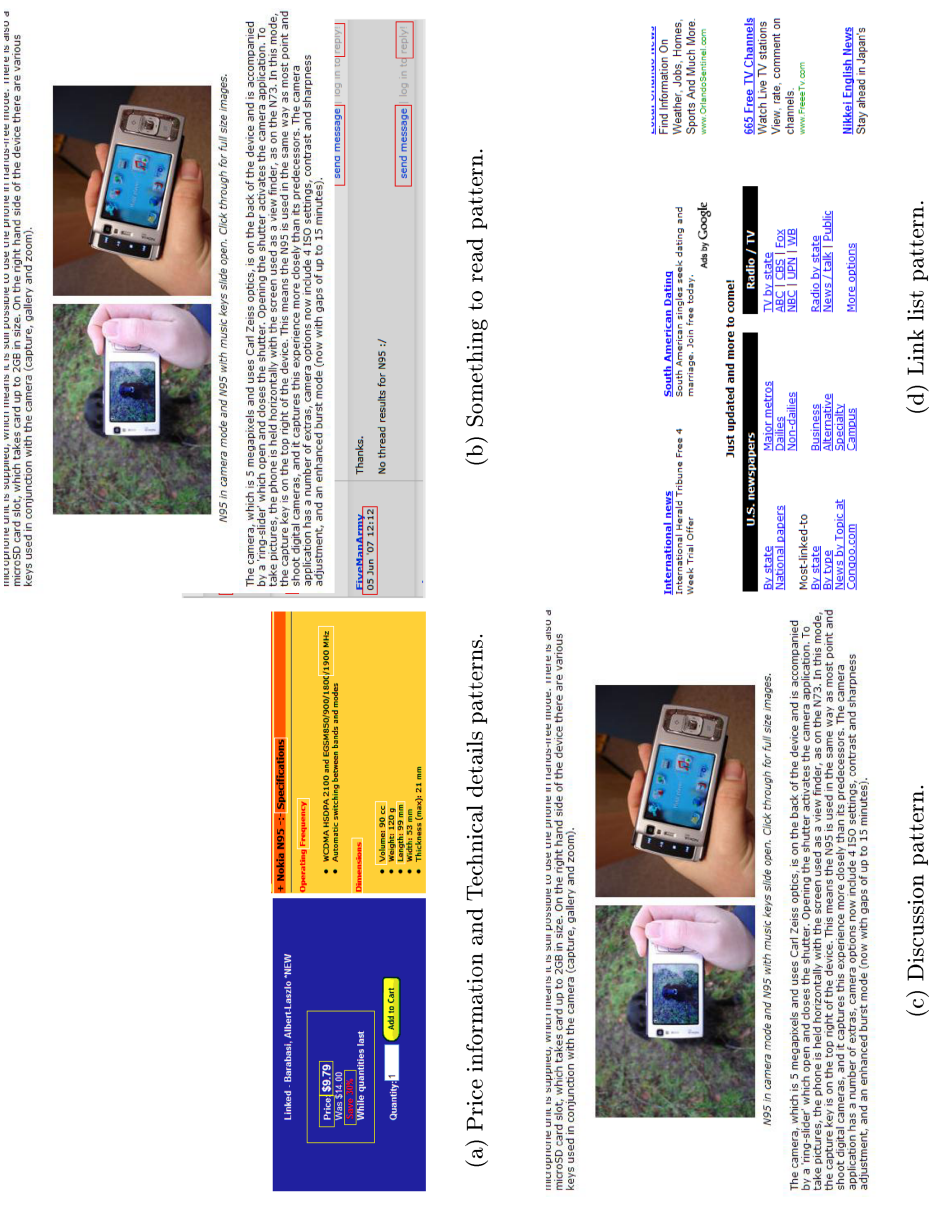
Table 1. Found clusters

#	Patterns	Pages
1	Price, Purchase, Special, Details	260
2	Price, Purchase, Special	1280
3	Review	1380
4	Price, Purchase, Special, Repayment, Details, Login	170
5	Price, Purchase, Special, Advert	200
6	Price, Special	630
7	Price, Purchase, Login	580
8	Price, Purchase, Special, Login	650
9	Price, Purchase, Details, Login	200
10	Price, Purchase, Special, Details, Login	340
11	Price, Special, Advert	270
12	Price, Advert	360
13	Price	1410
14	Price, Login	240
15	Login	660
16	Discussion, Login	340
17	Discussion	750
18	Price, Purchase, Special, Repayment	350
19	Price, Purchase, Advert	220
20	Price, Purchase, Details	260
21	Price, Purchase	1360
22	Discussion, Review, Login	300
23	Discussion, Review	800
24	Price, Purchase, Special	350
25	Price, Purchase, Special, Repayment, Details	140

known as a Kohonen map [9]. In Fig. 3, the Kohonen map is displayed with numbered clusters.

In each cluster there are pages which

1. contain mentioned web patterns in the table (moreover with weight greater than 0.6),
2. do not contain other web patterns (with weight lesser than 0.3).



(a) Price information and Technical details patterns.

(b) Something to read pattern.

(c) Discussion pattern.

(d) Link list pattern.

Fig. 4. Samples of web patterns

The clusters are numbered and described in Table 1. The web patterns names are abbreviated. There are four profile types represented by searched cluster:

1. Selling: 1, 2, 4, 6, 7, 8, 9, 10, 18, 20, 21, 24, 25 (6,570 pages)
2. Information: 3, 16, 17, 22, 23 (3,570 pages)

3. Advertising: 5, 11, 12, 19 (1,050 pages)
4. Some clusters: 13, 14, 15 (2,310 pages)

There are approximately 13,500 pages in all the mentioned clusters. For approximately 11,200 pages it is possible to assign their type (Selling, Information, and Advertising). On approximately 2,300 pages, it is undecidable whether they are from another domain or if it is an error in the web pattern detection. The pages are probably from a different domain than the selling product domain. They may be pages from a tourism domain which requires different web patterns for use (patterns Price and Login can occur). Aside from the discussed pages, there are approximately 10,000 pages which are not in any cluster. This is the case when a different combination of detected web patterns with different weights was detected.

8 Conclusion

We were focusing on selecting and describing web patterns with regard to the analysis of the web pages. The key factor for us is the human factor. We are convinced that using the web patterns which are developed within interaction between users and web page designers is very useful. Our experiments show that there are two perspectives. The first perspective is purely technical. It is possible to extend meta- information about a web page with a page profile (extracted from web patterns). The information should be used for search engines. The second perspective is user based. In web interface tests, we tried to involve users for the process of searching and reordering the result set. Our experiments imply that users understand the interface. Currently, we are preparing other experiments using web pattern detection in web searching.

References

1. Alexander, Ch.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
2. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann Publishers, San Francisco (2003)
3. Chang, Ch.H., Kayed, M., Girgis, M.R., Shaalan, K.F.: A Survey of Web Information Extraction Systems. IEEE Transactions on Knowledge and Data Engineering 18(10), 1411–1428 (2006)
4. Dearden, Finlay, J.: Pattern Languages in HCI: A critical review. Human-Computer Interaction 21(1), 49–102 (2006)
5. Dong, J., Zhao, Y.: xperiments on Design Pattern Discovery. In: PROMISE 2007. Third International Workshop on Predictor Models in Software Engineering, p. 12 (2007)
6. Van Duyne, D.K., Landay, J.A., Hong, J.I.: The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience. Pearson Education (2002)
7. Ivory, M.Y., Megraw, R.: Evolution of Web Site Design Patterns. ACM Transactions on Information Systems 23(4), 463–497 (2005)

8. Kiyavitskaya, N., Zeni, N., Cordy, J.R., Mich, L., Mylopoulos, J.: Text Mining Through Semi Automatic Semantic Annotation. In: PAKM 2006, pp. 143–154 (2006)
9. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Heidelberg (2006)
10. Kudělka, M., Snášel, V., Lehečka, O., El-Qawasmeh, E.: Semantic Analysis of Web Pages Using Web Patterns. In: WI 2006. International Conference on Web Intelligence, Hong Kong, pp. 329–333 (2006)
11. Kočibova, J., Klos, K., Lehečka, O., Kudělka, M., Snášel, V.: Web Page Analysis: Experiments Based On Discussion and Purchase Web Patterns. In: WI 2006. International Conference on Web Intelligence, Silicon Valley, CA, USA, pp. 221–225 (2007)
12. Nie, Z., Wen, J-R., Ma, W-Y.: Object-level Vertical Search. In: CIDR 2007, Asilomar, CA, USA, pp. 235–246 (2007)
13. Nie, Z., Ma, Y., Shi, S., Wen, J-R., Ma, W-Y.: Web Object Retrieval. In: WWW 2007, pp. 81–90 (2007)
14. Pivk, A.: Automatic Ontology Generation from Web Tabular Structures, PhD thesis, University of Maribor (2005)
15. Reis, D.C., Golgher, P.B., Silva, A.S., Laender, A.F.: Automatic web news extraction using tree edit distance. In: WWW 2004, pp. 502–511. ACM Press, New York (2004)
16. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM 18(11), 613–620 (1975)
17. Snášel, V., Řezanková, H., Húsek, D., Kudělka, M., Lehečka, O.: Semantic Analysis of Web Pages Using Cluster Analysis and Nonnegative Matrix Factorization. In: AWIC 2007, Fontainebleau, France, pp. 328–336. Springer, Heidelberg (2007)
18. Snášel, V.: GUI Patterns and Web Semantics. In: CISIM 2007, pp. 14–19. IEEE, Elk, Poland (2007)
19. Tidwell, J.: Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media, Inc. (2006)
20. Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., Halkidis, S.T.: Design Pattern Detection Using Similarity Scoring. IEEE Transactions on Software Engineering 32(11), 896–909 (2006)
21. Yu, S., Cai, D., Wen, J-R., Ma, W-Y.: Improving Pseudo-Relevance Feedback in Web Information retrieval Using Web Page Segmentation. In: World Wide Web conference (WWW 2003), Hungary, pp. 203–211 (2003)

Compression of Concatenated Web Pages Using XBW*

Radovan Šesták and Jan Lánský

Charles University, Faculty of Mathematics and Physics,
Department of Software Engineering
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
radofan@gmail.com, zizelevak@gmail.com

Abstract. XBW [10] is modular program for lossless compression that enables testing various combinations of algorithms. We obtained best results with XML parser creating dictionary of syllables or words combined with Burrows-Wheeler transform - hence the name XBW. The motivation for creating parser that handles non-valid XML and HTML files, has been system EGOTHOR [5] for full-text searching. On files of size approximately 20MB, formed by hundreds of web pages, we achieved twice the compression ratio of bzip2 while running only twice as long. For smaller files, XBW has very good results, compared with other programs, especially for languages with rich morphology such as Slovak or German. For any big textual files, our program has good balance of compression and run time.

Program XBW enables use of parser and coder with any implemented algorithm for compression. We have implemented Burrows-Wheeler transform which together with MTF and RLE forms block compression, dictionary methods LZC and LZSS, and finally statistical method PPM. Coder offers choice of Huffman and arithmetic coding.

1 Introduction

In this article we list results of compression of big XML files. Motivation for this work has been compression of data from web. Speed is very important for full-text searching and hence compression is not always the best choice. On the other hand archiving of old versions of web pages requires vast amount of space on disk. Also this data is not often used and hence compression could help with insufficient dist space. XML format is very redundant and therefore very good compression ratio can be achieved. Furthermore related web pages, with regard to its origin, contain long sequences of identical data. These properties of data enabled us to compress the data to tenth of original size.

We used for testing XML files from system *EGOTHOR* [5]. These files have size around 20MB and were formed by concatenation of hundreds of web pages and contain lots of text. Big files can be compressed more effectively due to

* This work was supported by Charles University Grant Agentur in the project "Text compression" (GAUK no. 1607, section A) and by the Program "Information Society" under project 1ET100300419.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 743–754, 2008.

following reasons. With the use of dictionary there is better ratio of size of dictionary to size of file. And most importantly entropy of text files is decreasing which means that following characters can be better predicted. Problematic is the fact that these files do not have valid XML structure and often they are not well formed. This lead us to creating our own parser since the parsers we know of could not handle these files.

In the next section we describe parts of program XBW and their influence on compression. Then we list results of measurements and comparison with common compression programs.

1.1 Conflicting Name XBW

The XBW method was not named very appropriately, because it can be easily mistaken by the name xbw used by the authors of paper [4] for XML transformation into the format more suitable for searching. In another article these authors renamed the transformation from xbw to XBW. Moreover they used it in compression methods called XBzip and XBzipIndex.

Another confusion comes from the fact that originally XBW stood in our articles for combination of methods BWT+MTF+RLE using alphabet of syllables for valid XML files. In what we know present as XBW the main idea of compression XML using BWT is present, but otherwise significantly differs from original work.

2 Implemented Methods

In Figure 1 the connections of parts of program is shown. All parts are optional. From algorithms RLE, LZC, LZSS and PPM at most one can be used because all of these algorithms use coder. In XBW Huffman (HC) and arithmetic coder (AC) are implemented. We have obtained the best results with combination of Parser + BWT + MTF + RLE + AC, which is used as default settings of the program.

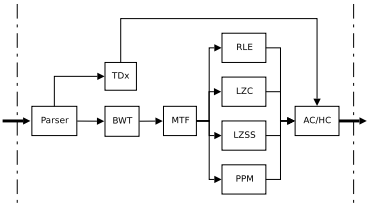


Fig. 1. XBW architecture

2.1 Parser and TDx

Implemented parser uses syntax of XML files for shortening the output. It omits closing characters and dynamically creates dictionary of tags and attributes. The rest of the data is split into characters, syllables or words and these are added to

trie. This choice is one of the parameters of parser. The use of words as alphabet is quite common for text compression, but its use for XML is not. The splitting into words or syllables and coding of dictionary is based on work of Lnsk [11].

It is also possible to use parser in text mode in which the structure of XML files is not taken into account and the input is only split into symbols of chosen alphabet (characters, syllables or words). Another parameter for parser is the choice of file coding; supported are dozens of codings for which we used library *iconv* [16].

Dictionary, that is stored in memory in form of trie during processing the file, is saved to output by coding its structure using methods TD1, TD2 or TD3 [12]. In method TD1 for each node we code the distance from left son, number of sons and boolean value which determines if the node represents a string. Methods TD2 and TD3 take advantage of the fact that we are coding syllables and words from natural languages which satisfy special conditions and hence better compression is achieved. Default method is TD3.

2.2 Block Compression

The class of methods for compression based on work of Burrows and Wheeler [1], uses reversible transformation which is called Burrows-Wheeler Transform (BWT). Often the combination of this transformation together with following effective coding is called block compression. The name comes from the fact that the input is split into block of fixed size and BWT is performed for each block. In default settings of our implementation we combine BWT with algorithms MTF and RLE. Decoding is significantly faster for BWT than coding.

Very important parameter of BWT is size of block to used. Larger blocks have better compression ratio at cost of longer run time and higher memory requirements. For example, in bzip2 algorithm, the maximum block size is 900 KB. In our program we use default setting of 50MB which can be changed.

Algorithm MTF used after BWT renumbers input and the result is string which contains relatively small numbers and sequences of zero. We implemented MTF using splay tree [7] which improves its speed especially for large alphabet. After MTF algorithm RLE is run which outputs the character and number of its occurrences. This is written in form of bits using coder.

We have RLE in three variants RLE1, RLE2 and RLE3 using RLE2 as default. In variant RLE1 sequence of repeating characters is replaced by three symbols. First one is escape sequence followed the character and number of occurrences. Alphabet used is increased by this escape symbol. In variant RLE2 each character there is also special escape symbol hence the resulting alphabet is twice the size of original. Repeating sequence of characters is replaced by escape symbol of the character and number of occurrences. In variant RLE3 we add to alphabet for each character special symbol for each possible number of repeatings of the character. We limit the length of sequence by fixed number. Hence in output we replace each sequence by special symbol. This method is suitable for small alphabets with small limit to length of sequence.

2.3 Dictionary Methods

Dictionary compression methods are usually adaptive; during coding they update dictionary of phrases. During compression we search for longest match of uncoded part of input with some phrase from dictionary. These methods work especially well for inputs where short sequences are (max 5-15 characters) repeating which is true for textual data. These algorithms are frequently used, because they are relatively fast and use little memory. When these algorithms are used, XBW has similar results as Gzip which also uses dictionary methods.

There are two main types of methods one based on LZ77 [17] and on LZ78 [18]. Method LZ77 has dictionary represented by compressed part of document in form of sliding window of fixed size which is moving right during compression. Method LZ78 builds the dictionary explicitly. In each step of compression a phrase from dictionary used for coding is lengthened by one character which follows after this phrase in uncompressed part of input.

LZC. LZC [6] is an improved version of LZ78 which uses trie data structure for dictionary and the phrases are numbered by integers in order of adding. During initialization, the dictionary is filled with all characters from the alphabet. In each step we search for maximal string S in dictionary being a prefix of non-coded part of input. The number of phrase S is then sent to the output. Actual input position is moved forward by the length of S. If the compression ratio starts to get worse the dictionary is cleaned. During decoding, if we get number of phrase that is in the dictionary we output the phrase. If the number does not stand for any phrase in the dictionary we can create that phrase by a concatenation of the last added phrase with its first character.

LZSS. LZSS [15] is improved version of LZ77 where the dictionary is represented by sliding window which we shift to the right during compression. We search for the longest prefix of non-coded input which matches string S from sliding window. LZ77 outputs always ordered triple $< D, L, N >$, where D is the distance of found string S in sliding window from its beginning, L is the length of S and N is the symbol following S in non-coded part of input. LZ77 has the disadvantage that if no match is found, hence S has zero length, output has been unnecessarily long. So in LZSS minimal threshold value is used when to code string S using D and L. If it is shorter, we code it as sequence of characters N. We use one bit to signal if the match for S has been long enough. So the output in each step is either $< 0, D, L >$ or several tuples $< 1, N >$.

2.4 PPM Method

The newest implemented method is statistical method called Prediction by Partial Matching (PPM) [2], which codes characters based on their probability after some context. Probabilities of characters after contexts are counted dynamically. This method, designed for compression of texts in natural languages, is quite slow and requires lots of memory. We have implemented variants PPMA, PPMB, PPMC with optional exclusions and setting for maximal context.

2.5 Coder

Final bit output is provided by coder. We implemented Huffman and arithmetic coder. Both variants are implemented in static and adaptive version. Arithmetic coder uses Moffat data structure and Huffman coder is implemented in canonic version. The choice of either Huffman or arithmetic coder is given at compile time. Default is the arithmetic coder, because it yield slightly better compression ratio than Huffman, and it is significantly faster when adaptive versions are used.

Huffman coding is compression method which assigns symbols codes of fixed length. It generates optimal prefix code which means that no code is a prefix of another code. Codes of all symbols are in binary tree and the edges have values 0 and 1. The code for symbol is given by the path from root to node representing the symbol. In static version we know the frequencies of symbols before construction of the tree. We sort symbols by their frequencies. In each step we take two symbols A and B with the smallest frequencies and create new one C which has frequency the sum of A and B they are his sons. In adaptive version we start with tree with one node representing escape symbol which is used for insertion of unencountered symbols. When adding node for unencountered symbol we create node A for this symbol and node B for escape symbol. Both have frequency one C representing original escape sequence is their father. When we increase frequency for symbol we first move its node to the right of nodes with equal frequencies. Then we increase its frequency and recursively repeat this step on its father.

The idea of arithmetic coding is to represent input as number from interval $[0, 1)$. This interval is divided into parts which stand for probability of occurrence of symbols. Compression works by specifying the interval. In each step interval is replaced by subinterval representing symbols of alphabet. Since the arithmetic coding does not assign symbols codes of fixed length, arithmetic coding is more effective than Huffman coding.

3 BWT

We describe Burrows-Wheeler transform in more detail, because its use in optimized version with input modified by parser allowed to get results we present. BWT during coding requires lexicographical order of all suffixes. The resulting string has on i-th place last symbol of i-th suffix. We assume that we have linear order on set of symbols Σ, which we call alphabet. Symbol in this sense can be character, syllable or word.

$X \equiv x_0x_1...x_{n-1}$, $\forall i \in \{0, .., n-1\}$, $x_i \in \Sigma$ is string of length n. i-th *suffix* of string X is string $S_i = x_ix_{i+1}..x_{n-1} = X[i..n-1]$. i-th suffix is smaller than j-th suffix, if first symbol in which they differ, is smaller, or i-th suffix is shorter. $S_i < S_j \iff \exists k \in 0..n-1 : S_i[0..k-1] = S_j[0..k-1]$ & $(S_i[k] < S_j[k] \vee (i+k = n \;\&\; j+k < n))$. We store the order of suffixes in *suffix array* SA, for which the following holds: $\forall i, j \in \{0..n-1\}$, $i < j \rightarrow S_{SA[i]} \leq S_{SA[j]}$.

The result of BWT for string X is $\tilde{X}$. $\tilde{X} \equiv \tilde{x}_0..\tilde{x}_{n-1}$ where $\tilde{x}_i = x_{|SA[i]-1|_n}$. Absolute values stand for modulo n, which is necessary in case $SA[i] = 0$.

Repetitiveness of the file influences the compression ration and run time of coding phase of BWT. We denote longest common prefix or match length by $lcp(S_i, S_j) = max\{k; S_i[0..k-1] = S_j[0..k-1]\}$. *Average match length* $AML \equiv \frac{1}{n-1} \sum_{i=0}^{n-2} lcp(S_{SA[i]}, S_{SA[i+1]})$ is the value we use in text for measuring repetitiveness of files.

We have implemented a few algorithms for sorting suffixes with different asymptotic complexity. The fastest algorithm for not too much repetitive files is ($AML < 1000$) is Kao's modification of Itoh algorithm [9], which has time complexity $O(AML \cdot n \cdot log\, n)$. For very repetitive files algorithm due to Krkkainen and Sanders [8] with complexity $O(n)$. Note that the choice of algorithm for BWT does not influence compression ratio, but only time and memory requirements.

In block compression the file is split into blocks of fixed size and BWT is run on each block. This method is used to decrease memory and time requirements, because BWT requires memory linear in size of input. Time complexity of most of algorithms for BWT is asymptotically super linear and BWT is the slowest part of block compression during compression. However, use of smaller blocks worsens the compression ratio.

The main reason why XBW has significantly better compression ration than bzip2 is the use of BWT on whole file at once. Our program runs in reasonable time thanks to preprocessing of the input by parser and the use of alphabet of words, which shortens the input for BWT. Very important consequence of the use of parser is the decrease of the value of AML. However use of parser requires use of algorithms, which do not work with byte alphabet of size 256 characters, but can work with 4 byte alphabet. When words are used as alphabet, for the tested files, the size of alphabet created by parser is approximately 50 thousand.

4 Corpora

Our corpus is formed by three files which come from search engine *EGOTHOR*. The first one is formed by web pages in Czech, the second in English and third in Slovenian. Their size in this order are: 24MB, 15MB, 21MB and the values of AML, describing their repetitiveness, are approximately 2000. Information about compression ratio of XBW on standard corpora Calgary, Cantebury and Silesia can be found in [10].

5 Results

First we list results of program XBW for various compression methods and the effect of parser on the results. Then we show influence of alphabet. At the end we compare results of XBW using optimal parameters with commonly used programs Gzip, Rar and Bzip2.

All results have been obtained using arithmetic coder. BWT has been run over whole input at once followed by MTF and RLE (parameter RLE=2). PPM has run with parameters PPM_ exlusions=off a PPM_order=5.

The size of compressed files includes coded dictionary which is created always when parser is used. Compression ratio is listed in bits per byte.

The run time has been measured under Linux and stands for sum of system and user time. This implies that we list time without waiting for disk. Measurements has been performed on PC with processor AMD Athlon X2 4200+ with 2GB of RAM. The data is in megabytes in second where the uncompressed size of file is used both for compression and decompression.

Table 1 shows the results of compression ratio for various methods for alphabet of characters. These results show effect of XML, which improves the compression ratio by approximately ten percent.

Next in Tables 2 and 3 we list the speed of program with and without parser using alphabet of characters. Results show that in almost all cases the parser degrades the speed. The reason is that we have to work with dictionary and the time saved by slightly shortening the input does not compensate for the work with dictionary. The exception is compression using BWT. Here the shortening the input and decreasing its repetitiveness significantly fastens BWT, which is the most demanding part of block compression.

Method commonly used for text compression is the use of words as symbols of alphabet. In Table 4 we show the influence of alphabet on compression ratio. For textual data in English best compression ratio is achieved with using words and method BWT. For Czech and Slovenian the syllables are better, because these languages have rich morphology. One word occurs in text in different forms and each form is added into dictionary. With the use of syllables core of the word is added, which can be formed by more syllables, and the end of word. But these last syllables of words are common for many words and hence there are more occurrences of then in the text. For dictionary methods LZx the words are by far the best choice.

The effect of large alphabet on speed varies and is shown in Tables 5 and 6. For all algorithms the decompression is faster for words than for characters. On the other hand decompression when parser with words has been used is still slower than decompression without parser see Table 1. The use of words increases the speed of compression only when BWT is used. Significant increase in speed for BWT is due to shortening the input and decreasing approximately three times AML. Results for PPM and words are not shown since the program did not finish within hour.

Previous results show that the best compression ratio has the algorithm BWT. Also it is evident that parser improves compression ratio for all algorithms. The fastest compression is achieved using LZC and fastest decompression using LZSS.

Our primary criterion is compression ratio and since method BWT has by far the best compression ratio, we focus mainly on BWT. In case the speed is priority choice of dictionary methods is advisable.

Table 7 contains comparison of compression ratios for different choices of parser, which shows that words are best for English and syllables for Czech and Slovenian. The choice of either words or syllables depends on the size of file and on morphology of the language. For languages with rich morphology,

Table 1. Influence of parser on compression ratio for alphabet of symbols

bpB	No Parser				Text Parser				XML Parser			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,907	2,217	2,322	1,399	0,906	2,206	2,296	1,395	**0,894**	2,073	2,098	1,320
xml_en	0,886	2,044	2,321	1,292	0,887	2,044	2,321	1,292	**0,874**	1,915	2,115	1,239
xml_sl	0,710	1,982	2,010	1,205	0,710	1,979	2,003	1,204	**0,700**	1,850	1,797	1,129
TOTAL	0,834	2,093	2,213	1,305	0,833	2,087	2,200	1,303	**0,822**	1,957	1,998	1,234

Table 2. Influence of parser on compression speed

MB/s	No Parser				XML Parser - Symbols			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,368	3,587	1,498	0,106	0,457	2,668	1,418	0,091
xml_en	0,419	4,028	1,297	0,125	0,544	2,915	1,249	0,104
xml_sl	0,386	4,258	1,638	0,119	0,500	2,915	1,497	0,091
TOTAL	0,386	3,906	1,485	0,115	0,491	2,810	1,397	0,094

Table 3. Influence of parser on decompression speed

MB/s	No Parser				XML Parser - Symbols			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	4,260	4,724	5,257	0,117	2,577	3,156	3,415	0,096
xml_en	4,417	4,999	5,417	0,142	2,705	3,397	3,606	0,110
xml_sl	4,918	5,236	5,946	0,134	3,012	3,299	3,672	0,097
TOTAL	4,509	4,960	5,519	0,128	2,747	3,263	3,548	0,099

Table 4. Influence of alphabet on compression ratio

bpB	XML Parser											
	Symbols				Syllables				Words			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,894	2,073	2,098	1,320	0,854	1,796	1,841	N/A	0,857	1,683	1,654	N/A
xml_en	0,874	1,915	2,115	1,239	0,836	1,626	1,785	N/A	0,830	1,514	1,558	N/A
xml_sl	0,700	1,850	1,797	1,129	0,664	1,559	1,541	N/A	0,668	1,457	1,390	N/A
TOTAL	0,822	1,957	1,998	1,234	0,783	1,672	1,723	N/A	0,785	1,563	1,539	N/A

Table 5. Influence of Alphabet on compression speed

MB/s	XML Parser - Symbols				XML Parser - Words			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	0,457	2,668	1,418	0,091	1,587	0,279	1,477	N/A
xml_en	0,544	2,915	1,249	0,104	2,009	0,920	1,093	N/A
xml_sl	0,500	2,915	1,497	0,091	1,566	0,443	1,349	N/A
TOTAL	0,491	2,810	1,397	0,094	1,666	0,399	1,319	N/A

Table 6. Influence of Alphabet on decompression speed

MB/s	XML Parser - Symbols				XML Parser - Words			
	BWT	LZC	LZSS	PPM	BWT	LZC	LZSS	PPM
xml_cz	2,577	3,156	3,415	0,096	3,986	3,951	3,923	N/A
xml_en	2,705	3,397	3,606	0,110	4,006	4,443	4,523	N/A
xml_sl	3,012	3,299	3,672	0,097	4,241	4,157	4,237	N/A
TOTAL	2,747	3,263	3,548	0,099	4,076	4,135	4,167	N/A

Table 7. Compression ratio for BWT

bpB	No Parser	Text Parser			XML Parser		
		Symbols	Syllables	Words	Symbols	Syllables	Words
xml_cz	0,907	0,906	0,859	0,862	0,894	**0,854**	0,857
xml_en	0,886	0,887	0,842	0,836	0,874	0,836	**0,830**
xml_sl	0,710	0,710	0,669	0,672	0,700	**0,664**	0,668
TOTAL	0,834	0,833	0,789	0,790	0,822	**0,783**	0,785

Table 8. Compression speed for BWT

MB/s	No Parser	Text Parser			XML Parser		
		Symbols	Syllables	Words	Symbols	Syllables	Words
xml_cz	0,368	0,324	1,056	**1,767**	0,457	1,073	1,587
xml_en	0,419	0,364	1,225	**2,128**	0,544	1,330	2,009
xml_sl	0,386	0,331	1,102	**1,790**	0,500	1,135	1,566
TOTAL	0,386	0,336	1,110	**1,853**	0,491	1,150	1,666

Table 9. Decompression speed for BWT

MB/s	No Parser	Text Parser			XML Parser		
		Symbols	Syllables	Words	Symbols	Syllables	Words
xml_cz	4,260	2,628	4,277	**4,817**	2,577	3,710	3,986
xml_en	4,417	2,494	4,612	**4,764**	2,705	3,981	4,006
xml_sl	4,918	2,639	4,686	**5,442**	3,012	3,685	4,241
TOTAL	4,509	2,598	4,494	**5,002**	2,747	3,765	4,076

Table 10. Running time for different parts of XBW

Seconds	Compression				Decompression			
	Parser	BWT	MTF	RLE	Parser	BWT	MTF	RLE
xml_cz	4,668	7,788	0,748	0,72	1,98	0,764	0,868	1,328
xml_en	2,364	3,800	0,388	0,448	1,112	0,716	0,440	0,796
xml_sl	3,352	7,404	0,496	0,504	1,592	0,676	0,556	0,916
TOTAL	10,384	18,992	1,632	1,672	4,684	2,156	1,864	3,04

Parser in text mode using words; BWT using Itoh; RLE - version 3

Table 11. Comparison of compression ratio

bpB	XBW	Gzip	Bzip2	Rar
xml_cz	**0,857**	1,697	1,406	1,161
xml_en	**0,830**	1,664	1,299	0,851
xml_sl	**0,668**	1,373	1,126	0,912
TOTAL	**0,785**	1,584	1,275	0,998

XBW: parser in text mode using words, Kao's algoritm for BWT
Gzip: gzip -9; Rar: rar -m5; Bzip2: bzip2 -9

Table 12. Comparison of compression speed

MB/s	Compression				Decompression			
	XBW	Gzip	Bzip2	Rar	XBW	Gzip	Bzip2	Rar
xml_cz	1,732	**10,320**	3,170	2,708	4,087	**25,004**	9,430	3,955
xml_en	2,058	**11,587**	3,454	2,689	4,309	**46,926**	11,722	6,137
xml_sl	1,758	**13,713**	3,245	3,190	4,614	**46,986**	13,132	4,775
TOTAL	1,812	**11,634**	3,262	2,853	4,313	**34,629**	11,045	4,640

XBW: parser in text mode using words, Kao's algoritm for BWT
Gzip: gzip -9; Rar: rar -m5; Bzip2: bzip2 -9

and for smaller files the syllables are better. Choice of either words or syllables effects the number of occurrences of symbols from dictionary for the input text. In program XBW we have implemented a few methods for splitting words into syllables. Results have been obtained using the choice *Left*. More details can be found in [10]. Interesting is the fact that the XML mode of parser has small influence on compression ratio. This is not due to incorrect implementation of parser, but due to properties of BWT for large blocks. For example for LZx methods the effect is significant. Again more detailed results are in [10].

Table 8 show the influence of parser on the speed of program. The fastest by far is the choice of words as symbols of alphabet for compression. For decompression (see table 9) the differences are small. In order to improve the speed it is better to use parser in text mode instead of XML mode for words.

There are many algorithms for sorting suffixes in BWT. The choice of this algorithm has big impact of overall performance of compression. Without the use of parser, sorting suffixes for big blocks amount up 90% of run time of whole program. More details are in [14]. For all files the fastest is Kao's modification of Itoh's algorithm [9] and it has been used in all measurements when BWT has been used.

Run time of separate parts of program are in Table 10. These times show in which parts there is the most room for improvement.

6 Comparison with Other Programs

For comparison we show the results of programs Gzip, Rar and Bzip2. Programs for compression of XML data such as XMLPPM [3] and Xmill [13] can not cope

with non-valid XML files. Hence we could not get their results on our data. For programs Gzip, Rar and Bzip2 we used parameters for the best available compression. In Table 11 we list compression ratios. Our program compresses all files the best and is significantly better for files which are not in English.

In Table 12 are the results for speed of compression and decompression. The fastest is Gzip, but it also has the worst compression ratio and hence we compare speed of XBW only with Rar and Bzip2. Compression for XBW takes less twice the minimum of Rar and Bzip2. Decompression if comparably fast as for Rar and Bzip2 is approximately three times faster.

The performance of XBW is sufficient for common use, however it is slower than the speed of hard disks, and hence where speed is priority, it is better to use program based on dictionary methods such as Gzip. XBW has the best compression ratio and therefore it is suitable especially for long term archiving.

7 Future Work

In future work on XBW we aim to focus on two directions. The first is creation of parser which could be used also on binary data. The later is improving the run time of program where again we see the biggest potential in parser.

References

1. Burrows, M., Wheeler, D.J.: A Block Sorting Loseless Data Compression Algorithm. Technical report, Digital Equipment Corporation, Palo Alto, CA, U.S.A (2003)
2. Cleary, J.G., Witten, I.H.: Data compression using adaptive coding and partial string matching. IEEE Transactions on Communications COM-32(4), 396–402 (1984)
3. Cheney, J.: Compressing XML with Multiplexed Hierarchical PPM Models. In: Storer, J.A., Cohn, M. (eds.) Proceedings of 2001 IEEE Data Compression Conference, p. 163. IEEE Computer Society Press, Los Alamitos, California, USA (2001)
4. Ferragina, P., Luccio, F., Manzini, G., Muthukrishnan, S.: Structuring labeled trees for optimal succinctness, and beyond. In: FOCS 2005. Proc. 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 184–193 (2005)
5. Galamboš, L.: EGOTHOR, http://www.egothor.org/
6. Horspool, R.N.: Improving LZW. In: Storer, J.A., Reif, J.H. (eds.) Proceedings of 1991 IEEE Data Compression Conference, pp. 332–341. IEEE Computer Society Press, Los Alamitos, California, USA (1991)
7. Jones, D.W.: Application of splay trees to data compression. Communications of the ACM 31(8), 996–1007 (1988)
8. Kärkkäinen, J., Sanders, P.: Simple linear work suffix array construction. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 943–955. Springer, Heidelberg (2003)
9. Kao, T.H.: Improving suffix-array construction algorithms with applications. Master Thesis. Gunma University, Japan (2001)
10. Lánský, J., Šesták, R., Uzel, P., Kovalčin, S., Kumičák, P., Urban, T., Szabó, M.: XBW - Word-based compression of non-valid XML documents, http://xbw.sourceforge.net/

11. Lánský, J., Žemlička, M.: Compression of a Dictionary. In: Snášel, V., Richta, K., Pokorný, J. (eds.) Proceedings of the Dateso 2006 Annual International Workshop on DAtabases, TExts, Specifications and Objects. CEUR-WS, vol. 176, pp. 11–20 (2006)
12. Lánský, J., Žemlička, M.: Compression of a Set of Strings. In: Storer, J.A., Marcellin, M.W. (eds.) Proceedings of 2007 IEEE Data Compression Conference, p. 390. IEEE Computer Society Press, Los Alamitos, California, USA (2007)
13. Liefke, H., Suciu, D.: XMill: an Efficient Compressor for XML Data. In: Proceedings of ACM SIGMOD Conference, pp. 153–164 (2000)
14. Šesták, R.: Suffix Arrays for Large Alphabet. Master Thesis, Charles University in Prag (2007)
15. Storer, J., Szymanski, T.G.: Data compression via textual substitution. Journal of the ACM 29, 928–951 (1982)
16. The Open Group Base: iconv. Specifications Issue 6. IEEE Std 1003.1 (2004), http://www.gnu.org/software/libiconv/
17. Ziv, J., Lempel, A.: A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory 23(3), 337–342 (1977)
18. Ziv, J., Lempel, A.: Compression of Individual Sequences via Variable-Rate Coding. IEEE Transactions on Information Theory 24(5), 530–536 (1978)

The Dynamic Web Presentations with a Generality Model on the News Domain

Hyun Woong Shin[1], Eduard Hovy[2], and Dennis McLeod[3]

[1] SAMSUNG ELECTRONICS CO., LTD.,
416 Maetan-3Dong, Yeongtong-Gu, Suwon-City, Gyeonggi-Do, Korea 443-742
hws52.shin@samsung.com
[2] Information Sciences Institute of the University of Southern California
4676 Admiralty Way. Marina del Rey, CA, USA
hovy@isi.edu
[3] Computer Science Department, Integrated Media Systems Center,
University of Southern California, 941 W. 37th Place, Los Angeles, CA, USA
mcleod@usc.edu

Abstract. Over the last decade, Web and multimedia data have grown at a staggering rate. Users of new media now have great expectations of what they can see on the Web. In addition, most information retrieval systems, including Web search engines, use similarity ranking algorithms based on a vector space model to find relevant information in response to a user's request. However, the retrieved information is frequently irrelevant, because most of the current information systems employ index terms or other techniques that are variants of term frequency. This paper proposed a new approach, named "the dynamic multimedia presentations with a Generality Model," to offer a customized multi-modal presentation for an intended audience. Moreover, we proposed a new criterion, "generality," that provides an additional basis on which to rank retrieved documents. To support multi-modal presentation, our proposed story model created story structures that can be dynamically instantiated for different user requests from various multi-modal elements. The generality is a level of abstraction to retrieve results based on desired generality appropriate for a user's knowledge and interests. We compared traditional web news search functions and our story model by using usability test. The result shows that our multimedia presentation methodology is significantly better than the current search functions. We also compared our generality quantification algorithm with human judges' weighting of values to show that the developed algorithm is significantly correlated.

1 Introduction

The rapid growth of online digital information over the last decade has made it difficult for a typical user to find and read information he/she wants. A recent study shows there are around 40 million Web sites. The amount of digital media, nontextual information including images, audio, and video on the World Wide Web is enormous and is growing at a staggering rate. Users of new media now have great expectations about what they can access online and are demanding more powerful technologies. To

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 755–765, 2008.

satisfy those user's expectations, there have been several studies for multimedia presentations in various domains [4, 8]. One way to address the problem of information overload is to tailor that information to specific user interests, needs and knowledge base. If there is an approach that responds to individual information requests with an original, dynamically built story, several problems are solved.

First, in today's Web service industry, information presentations and collections of data are static and having limited multi-modal presentations. Critically, there is little capability to dynamically adapt an integrated presentation of information to a user. We believe that a user engages deeply into a story when he/she not only reads text articles but also watches videos and/or listens to audio clips in a coordinated manner. Second, most of current web search engines deliver a huge amount of hyperlinks. Although this helps improve accuracy (recall), an end user has a trouble deciding which results are what he/she wants. Finally, most web services do not instantiate and customize (for an individual user) "generic" stories. Even though the accuracy of results in terms of precision and recall may be acceptable, the results might not be relevant to a user's intention(s). The crux of retrieving more-relevant information is better characterizing a user's request. Unfortunately, this is not a simple problem Most traditional IR systems operationalize "relevant" as the word frequency in a document of a set of keywords (or index terms) [3, 12, 13, 19]. In other words, they characterize a user's request as a term frequency. However, there is another aspect to characterizing a user's request: the appropriate level of generality that is specification of desired information based on a user's knowledge and interests.

The proposed system creates story structures that can be dynamically instantiated for different user requests from various multi-modal elements. In addition, the proposed system focuses on quality of the results not quantity of results. Furthermore, the system leverages information so that a user reads an appropriate level of story depending upon the user's intention level ranging from general to specific. For example, a user might impress the USC football game, but the user has very little knowledge of the USC football team. The user then wants to read very general information instead of specific information regarding the team. When the user requests a general level of the USC football team, the system delivers a customized (in this case, a general story) dynamically generated multi-modal story.

To determine a user's intention and goal, a general knowledge-based process, with selection (information filtering) heuristics, is used. A key to the successful use of story types is the ability to relate and connect the user requests to the Content Database. A domain dependent ontology is essential for capturing the key concepts and relationships in an application domain [1, 9, 11]. Finally, meta-data descriptions connect a modified user request (by using domain ontology) to the Content Database for retrieving proper content elements.

The remainder of this paper is organized as follows. Section 2 provides an architectural overview of the proposed system. Section 3 introduces a new story model with visual techniques and presentation constraint specifications. Section 4 delineates a new criterion, generality, for measuring the generality of documents. In Section 5, experiments methodology and metrics are explained. In Section 6, discussion and results are illustrated. Concluding Remarks are presented in Section 7.

2 Overall Functional Architecture

The overall functional architecture of a proposed system is illustrated in figure 1. The model has two key phases: story assembly and content query formulation. In the story assembly phase, a structured rule-based decision process is introduced to determine a proper story type and to invoke a primary search and a secondary search in the content query formulation phase. At the beginning, the story assembly module receives a modified user's request from a query processing procedure, which consists of related concepts, a level of generality spectrum, media types that a user prefers and so on. Note that a requirement for a query processing procedure is to include a domain dependent ontology so that all involved components can understand the semantics of parameters without other overheads. These inputs then invoke a primary search to retrieve multimodal content objects, along with a constraint-based k-nearest neighbor search. These results are sent to the story type decision module to determine a proper story type and then fill in the chosen story type with multi-modal elements (content objects). If it is necessary, this decision module also invokes a secondary search to get extra elements.

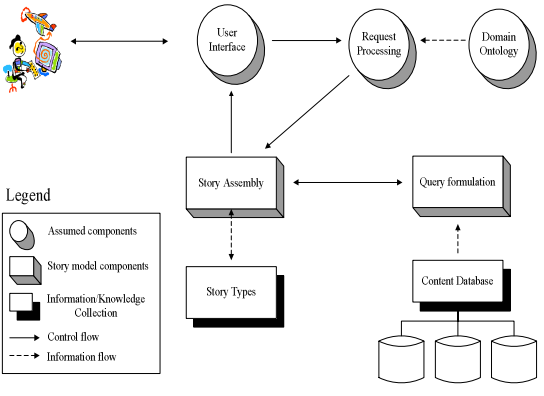


Fig. 1. Overall functional Architecture

3 Story Model

The proposed story model defines four story types that lay out an appropriate presentation style depending on a user's intention and goal. In order to provide an efficient presentation, the story model needs quantification for the size of each icon based on ranking in the retrieved content objects. Furthermore, the story model employs visual techniques that solve layout problems such as combining and presenting different types of information and adopts presentation constraint specifications to abstract higher levels of presentation so that lower levels of presentation can automatically generate a story that meets those specifications.

A visual technique depends on traditionally accepted visual principles to provide an arrangement of layout components [2]. This conventional arrangement, called a layout grid, consists of a set of parallel horizontal and vertical lines that divide the layout into units that have visual and conceptual integrity [14].

According to Vanderdonckt et al. [16], there are five sets of visual techniques: physical techniques (e.g. balanced vs. unbalanced layout), composition techniques (e.g. simple vs. complex layout), association and dissociation techniques (e.g. grouped vs. splitted layout), ordering techniques (e.g. sequential vs. random layout), and photographic techniques (e.g. round vs. angular layout). We focus only on balance and symmetry of physical techniques because balance is a highly recommended technique evoked by many authors [5, 7, 10]. Balance is a search for equilibrium along a vertical or horizontal axis in layouts. Symmetry consists of duplicating visual images along with a horizontal and/or vertical axis [6, 10]. Thus, achieving symmetry automatically preserves balance.

Presentation constraints are typically expressed in terms of a timeline, screen layout, or navigation structure. In most constraint systems, only certain aspects of the presentation are adapted to satisfy each constraint. Multimedia presentation structures consist of multiple dimensions, primarily including space, time and navigation [15, 17, 18]. Our approach is only concerned with a spatial constraint because time and navigational constraints are not relevant to our presentation goal. Our presentation goal is to deliver an integrated multi-modal presentation with a balanced layout in response to a user's objective. Thus, timeline and navigation constrains are not considerable constraint specifications of our approach.

Graphical icons, including a scrollable box for a text, a fixed size window for images, and control boxes for audio and video clips are containers of elements in story types. In spatial constraint specifications, each container has a fixed size to be filled in by an element. This higher level of abstraction allows a consistent final presentation for the user.

4 Generality Model

The degree of generality can be quantified by the number of index terms in the document that belong to specific word sets which distinguishes a subject from others. For its quantification, we introduced the concept "specific word set" that consists of index terms not belonging to any other ontology nodes. Here, generality is quantified by the appearance of specific index terms t_i within document D_j.

The following is the formal definition of the algorithm.

Let $D = \{D_j \mid j \in J\}$ be a document containing a set of words from an index set J, and $S = \{t_i \mid i \in I\}$ be a set of specific terms from an index set I. The index set J is used to differentiate documents and the index set I contains specific words.

Define a characteristic function $\chi : I \times J \to \{0, 1\}$

by $\chi(i, j) = \begin{cases} 1, t_i \in D_j \\ 0, t_i \notin D_j \end{cases}$.

By using the characteristic function, we define the generality of a document D_j as follows:

$$g_j = \frac{\sum_{i \in I} \chi(i, j)}{|D_j|}$$ for the number $|D_j|$ of terms in D_j

If $D_j \cap S = \varnothing$, then $g_j = 0$ as a special case.

Once the degree of generality is determined for each document, we adjust the degree of generality based on the concept hierarchy. The concept hierarchy is a hierarchical structure of related concepts. For example, "Sports" may have a child node, the Olympic-Sports (which we abbreviate to "Olympics"). In addition, the node "Olympics" may have the children nodes "Boxing" and "Taekwondo." In this case, there is a conceptual hierarchy starting from "Sports" to "Olympics" to "Boxing" and "Taekwondo".

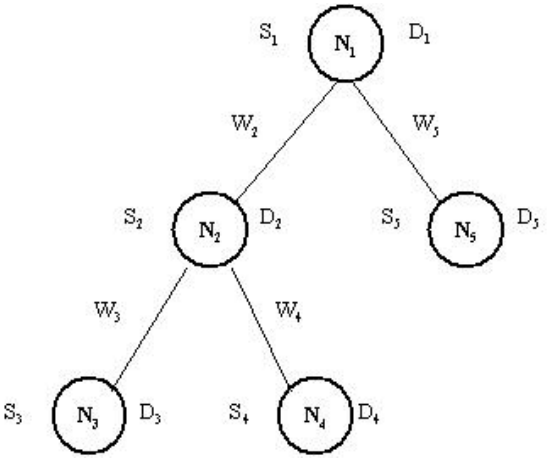


Fig. 2. Sample ontology with weights w_i, index term lists D_i, and specific term lists S_i

In many IR systems that utilize ontologies, a user submits a query, and the system determines the proper ontology node(s) that is (are) best matched with a given query. After the appropriate node(s) is (are) determined, the system retrieves the information from the concept hierarchy that is rooted at the chosen node. In a domain dependent ontology, an instance of a child node is also an instance of a parent node. Thus, the parent node might provide its own instances and instances of children. The generality of its own, direct instances can be calculated utilizing the above algorithm. For the other instances, the algorithm needs to adjust the degree of generality based on their position in the concept hierarchy. This adjustment is necessary for two reasons.

First, a child node represents more specific information than a parent node does. In other words, the degree of generality for all documents in a parent node must be assigned a higher value (be more general) than the documents in the child node. This

assumption we base on the intuition that domain-specific Ontologies will generally lie mostly below the level of Basic Concepts [19], and hence tend to become of more general interest as one moves upward through them.

Second, documents at the top level of a domain dependent ontology are in practice the most general stories, using the journalism definition. These documents should contain the largest number and variety of specific words. However, it is very unlikely that a document will contain all specific words. To overcome the problem, an adjusted value needs to be added to the degree of generality for each document in child nodes.

Figure 2 depicts a sample ontology graph with weights (for adjusting generality), index term lists, and specific term lists. The basic idea behind the degree of generality reflects the differences of specific word sets within the concept hierarchy. Unlike the basic generality computation that focused on each document, the adjustment is determined by the specificity of the ontology node. In other words, it focused on node itself. The adjustment is calculated by the number of specific words over the number of index term set on each ontology node. The following is the formal definition for the adjustment algorithm:

Suppose that S_k and S_{k+i} are sets of specific terms of a parent node and its children nodes, respectively $for\ i = 1, 2, ..., c$ (number of the children node for parent node N_k. Let $N = \{N_k \mid k \in A\}$ be a set of nodes for an index set A and k is ordering by depth, and $\{g_{i,m} \mid i \in A\ and\ m \in I\}$ be a set of the degree of generality, $g_{i,m}$ for a document d_m in a node N_i. The adjusted generality of a document d_m on an ontology node N_k is defined as follows:

$$d(k,m) = g_{k+i,m} + w_{k+i}$$

where $w_{k+i} = \frac{|S_k - S_{k+i}|}{|D_k|}$ for the number $|D_k|$ of terms in a node N_k and some child node N_{k+i} containing the document d_m. The node N_k is a parent node of a child node N_{k+i} and the $|D_k|$ is the total number of index terms that are used in the parent node N_k.

5 Experiments

5.1 Evaluation Plan for Story Model

To exam the usability of our system, we designed a controlled experiment. In our experiments, the total 25 students from the engineering and journalism schools (17 and 8, respectively) were selected and asked to fill up a questionnaire after experiencing the system. Appendix A shows the sample questionnaire and the figure of our system used in the experiment. The subjects were provided by an experimenter with a

brief instruction about the experiment and asked to have experience with two sites - the traditional news search functions such as CNN, LA Times, and Washington Post and our systems. 13 subjects began with the traditional news search function site and then our system site, while the other 12 subjects started with our system site and then the traditional web news search function site in order to avoid any possible order effect. At the end of each site, the subjects were asked to fill out an online questionnaire, which was hyperlinked from the last page of each site, with radio-button scaled responses and some open-ended questions which asked him/her to evaluate the four categories - Overall satisfaction; Functionality and capability; Learnability; and Interface design. Finally, all subjects were debriefed and thanked.

Table 1. Index terms and specific words

	$\lvert D \rvert$	$\lvert S \rvert$	$\lvert S' \rvert$
Sports	3148		2222
Olympics	1727	976	819
Baseball	757	303	
Football	960	420	
Golf	616	248	
Tennis	700	275	
Olympics-boxing	469	147	
Olympics-Basketball	424	111	
Olympics-Taekwondo	249	78	
Olympics-Running	597	222	
Olympics-Hockey	475	165	
Olympics-Baseball	381	96	

5.2 Corpus Analysis for Generality Model

Having developed a method to quantify generality, we now determine whether it conforms to human intuitions. Before starting work on the system, we collected and analyzed terms from a corpus to empirically guide the design of generality and generation of the domain dependent ontology. We studied the terms and conceptual hierarchy used to convey information from multiple sources including Associated Press, ESPN, and current newswire. We created a hierarchy of 12 nodes and three levels based on 62 articles.

In order to generate the proper degree of generality, we analyzed the underlying corpus. Table 1 depicts the total number of index terms and specific words in the corpus. In the table, D indicates the total number of index terms at or below a node. The total number of specific terms for each node alone is in S. S' is a summation of children node's specific terms. It is safe to say that $S \geq S'$ between a parent node and children nodes because some specific words S of the parent node are general words for children nodes. For example, the parent node "Olympics" contains "Olympiad," "Gold," "Bronze," and some country names that repeat in children nodes. Those specific words, however, are not included in the specific word set for children nodes.

For the node "Sports," specific terms cannot be determined because this node is the summation of all other nodes. In the node "Olympics," the number of specific terms ($S = 976$) is more than the summation of children's specific terms ($S' = 819$). According to our assumption, it indicates that the node "Olympics" may contain more topics than the summation of topics in children nodes.

5.3 Evaluation Plan for Generality Model

Our verification of the measure of generality is performed between a domain dependent ontology and human judges. Given documents, human judges were asked to mark the degree of generality for each document. The judges used a ten-point scale (as a continuous value) and assigned a score for each document based on their observation of the degree of generality. The judges were instructed that there are no right or wrong answers.

Since the human judges' values are critical to evaluate the algorithm, the selection of human judges is a huge problem. How many human judges are necessary? How should we adjust for individual variability? To lessen the problem, professional journalists were chosen as human judges. If random people from the general population were included, individual variation would be huge due to their backgrounds and education levels. This variation would increase the standard deviation and bring errors into the study. However, professional journalists are trained to obtain proper writing and reading skills in terms of journalism, so we assume that the individual variation among them in scoring new articles is not too large. The judges are a professor and a graduate student in the school of journalism at USC. We assumed that they already have developed adequate comprehension and writing skills, so no training session was carried out.

We used the Pearson correlation coefficients to evaluate the relationship between the scores from two human judges as well as from the human judges and from the algorithm. Also, we used a t-test (t distribution and n-2 degrees of freedom) to determine whether these relationships are statistically significant. If a p-value from the t-test is less than 0.05, we conclude there is a statistically significant correlation between the judges and the system.

6 Results and Discussion

6.1 Story Model

The online questionnaire was composed of based on the Cronbach's alpha showing that the questions in each category are highly reliable (Table 1). All questions are included in the results.

The paired t-tests were performed at each category, and the results were showed in table 2. In the all categories, our system is statistically significantly batter than the traditional web search engines (all p-values are less than 0.05). Our system's overall satisfaction, functionality and capability and interface design were more than one level up than the traditional ones'.

Table 2. Cronbach's alpha value of two sessions

	Cronbach's alpha	
	Session 1[1]	Session 2[2]
Overall	.8690	.9150
Functionality and Capability	.7747	.6941
Learnability	.8027	.7829
Interface Design	.8392	.8056

Table 3. Paired t-test value

		Mean of Paired Differences	t (df = 16)	Sig. (2-tailed)
Pair 1	Q101 - Q111	2.35	6.305	.000
Pair 2	Q111 - Q131	-2.29	-5.376	.000
Pair 3	Q102 - Q112	1.71	3.237	.005
Pair 4	Q112 - Q132	-1.88	-4.157	.001

6.2 Generality Model

The Pearson correlation coefficient always lies between -1 and +1 $(-1 \le r \le 1)$, and the values $r = 1$ and $r = -1$ mean that there is an exact linear relationship between the two values. Over 70% is generally considered a good correlation. Also, the significance of a correlation coefficient is examined by a t-test (n-2 degree of freedom).

We first test the generality between two judges' value to show that there is a common generality between human judges. This evaluation assures us that there is a phenomenon to be modeled and computationalized.

Table 4. Pearson correlation coefficient between two human judges[3]

	Level 0 (n=62)	Level 1 (n=62)	Level 2 (n=29)
Pearson correlation coefficient (r)	0.84	0.81	0.81
p-value from the t-test	< .0001 (df=60)	< .0001 (df=60)	<.0001 (df=27)

[1] Traditional Web Search Functions.
[2] Our system.
[3] n= number of articles used for the test, df= degrees of freedom.

Table 4 shows the Pearson coefficients and the corresponding p-values from the t-test between the two human judges. This result shows that their evaluations are statistically significantly (more than 80%, p<.001), in spite of individual variability. Level 0 is a parent node of Level 1, Level 1 is a parent node of Level 2, and so on. Although the human judges and the algorithm assign scores for each document in an ontology node, the correlation should be tested among siblings (i.e. same level nodes) because a correlation of each mode cannot provide the correlation in general. Table 5 shows the Pearson coefficients and the corresponding p-values from the t-test between human judges and the algorithm.

The results show that there are 73% and 68% correlations between the two at Level 1 and Level 2, respectively, and these relationships are statistically significant (p<0.0001). The scores from the human judges are competitive with those from our algorithm. At the top level, however, the correlation between human judges and the algorithm is very low because no matter what the algorithm calculates as the degree of generality, the judges determine it as 10.

7 Concluding Remarks

We have proposed a new dynamic generation of user-customized multimedia presentations. The proposed system defined four domain independent story types to generate a dynamic multimedia presentation in response to a user's intension. – a summary story type, a text-based story type, a non-text based story type and a structured collection story type. We conducted an experiment to examine user satisfaction of our system comparing with that of traditional web news search functions. The experimental result shows that our system is statistically significantly better in user satisfaction.

In addition to the story model, we introduced, defined, and confirmed the notion of generality that is used to indicate how general or specific a document is. A basic idea for quantification of generality and the algorithm has been devised and developed. We employed the Pearson's correlation coefficient to evaluate the relationships of the degrees of generality between the human judge and the algorithm. The experimental results show these relationships are statistically significant ($p < .0001$). As seen in Table 5, the Pearson correlation coefficients are 73% and 68% for Level 1 and Level 2, respectively.

References

[1] Alexakos, K., Vassiliadis, B., Likothanassis, S.: A Multilayer Ontology Scheme for Integrated Searching in Distributed Hypermedia. Springer, Heidelberg (2006)

[2] Bodart, F., Vanderdonckt, J.: Visual Layout Techniques in multimedia Applications. In: CHI Companion 1994, pp. 121–122 (1994)

[3] Buckley, C., Walz, J.: SMART in TREC 8. In: Proc. Eighth Text Retrieval Conf., pp. 577–582 (November 1999)

[4] Colineau, N., Phalip, J., Lampart, A.: The delivery of multimedia presentations in a graphical user interface environment. In: Proceedings of the 11th international conference on Intelligent user interfaces (2006)

[5] Davenport, G., Murtaugh, M.: ConText: Towards the Evolving Documentary. Proceedings ACM Multimedia, 381–389 (1995)
[6] Dondis, D.A.: A Primer of Visual Literacy. The MIT Press, Cambridge (1973)
[7] Dumas, J.S.: Designing User Interface for Software. Prentice-Hall, Englewood Cliffs (1988)
[8] Elias, S., Easwarakumar, K., Chbeir, R.: Dynamic consistency checking for temporal and spatial relations in multimedia presentations. In: Proceedings of the 2006 ACM symposium on Applied computing (2006)
[9] Khan, L., McLeod, D., Hovy, E.H.: Retrieval effectiveness of an ontology-based model for information selection. The VLDB Journal 13(1), 71–85 (2004)
[10] Kim, W.C., Foley, J.D.: Providing High-Level Control and Expert Assistance in the User Interface Presentation Design. In: Proceedings of Inter-CHI 1993, pp. 430–437. ACM Press, New York (1993)
[11] Mao, M., Peng, Y., He, D.: DiLight: an ontology-based information access system for e-learning environments. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (2006)
[12] Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
[13] Salton, G.: Automatic Text Processing – the Transformation, Analysis and Retrieval of Information by Computer. Addison-Wesley Publishing Co., Reading, MA (1989)
[14] Shneiderman, B., Kang, H.: Direct Animation: A Drag and Drop Strategy for Labeling Photos. In: VI2000. Proceedings of International Conference on Information Visualization, pp. 88–95 (2000)
[15] Smolensky, P., Bell, B., King, R., Lewis, C.: Constraint-based Hypertext for Argumentation. In: Proceedings of Hypertext, pp. 215–245 (1987)
[16] Vanderdonckt, J., Gillo, X.: Visual Techniques for Traditional and Multimedia Layouts. In: Proceedings of the workshop on advanced visual interfaces (1994)
[17] Weitzman, L., Wittenberg, K.: Automatic Presentation of Multimedia Documents Using Relational Grammars. Proceedings of ACM Multimedia, 443–451 (1994)
[18] Zhouh, M.X.: Visual Planning: A Practical Approach to Automated Presentation Design. In: IJCAI. Proceedings of the International Joint Conference on Artificial Intelligence, pp. 634–641 (1999)
[19] Zobel, J., Moffat, A.: Exploring the Similarity Space. In: Proc. ACM SIGIR Forum, vol. 32, pp. 18–34 (Spring 1998)

A Highly Efficient XML Compression Scheme for the Web

Przemysław Skibiński[1], Jakub Swacha[2], and Szymon Grabowski[3]

[1] University of Wrocław, Institute of Computer Science
Joliot-Curie 15, 50–383 Wrocław, Poland
inikep@ii.uni.wroc.pl
[2] Institute of Information Technology in Management, Szczecin University
Mickiewicza 64, 71–101 Szczecin, Poland
[3] Technical University of Łódź, Computer Engineering Department
Politechniki 11, 90–924 Łódź, Poland

Abstract. Contemporary XML documents can be tens of megabytes long, and reducing their size, thus allowing to transfer them faster, poses a significant advantage for their users. In this paper, we describe a new XML compression scheme which outperforms the previous state-of-the-art algorithm, SCMPPM, by over 9% on average in compression ratio, having the practical feature of streamlined decompression and being almost twice faster in the decompression. Applying the scheme can significantly reduce transmission time/bandwidth usage for XML documents published on the Web. The proposed scheme is based on a semi-dynamic dictionary of the most frequent words in the document (both in the annotation and contents), automatic detection and compact encoding of numbers and specific patterns (like dates or IP addresses), and a backend PPM coding variant tailored to efficiently handle long matching sequences. Moreover, we show that the compression ratio can be improved by additional 9% for the price of a significant slow-down.

Keywords: XML compression, semi-structural data compression, text transform, prediction by partial matching.

1 Introduction

Year after year, XML consolidates its hold as a standard for interchange of structured information. With the recent international standardization of the OpenDocument format and the introduction of the Open XML format in Microsoft Office 2007, this trend can only be accelerated.

As the information technology usage becomes more and more Web-centric, there is a considerable amount of XML documents exchanged through the Web every day. Contemporary XML documents can be tens of megabytes long, and although, from the standpoint of Internet providers, they represent only a small share of traffic dominated by video sharing websites and peer-to-peer file exchange networks, reducing the size of XML documents, thus allowing to transfer them faster, poses a significant advantage for their users.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 766–777, 2008.

There are several XML compression algorithms available, renowned for their high compression ratios. The ratios are *high* in absolute terms, still, XML data can be squeezed further with more advanced techniques.

In this paper, we describe a new XML compression scheme which sets the new state-of-the-art in XML compression. The scheme attains impressive compression ratios with reasonable compression/decompression times and supports streamed decompression, which makes it especially suitable for web applications.

The outline of this paper is as follows. Section 2 presents earlier achievements in XML compression. Section 3 compares transform-based approach to XML compression against other possibilities. The next section presents in detail our proposal, XWP transform designed for efficient PPM-like compression, and two compressors that we use together with XWP. Results of our experiments are shown in Section 5. The final section concludes.

2 Related Work

The high redundancy of XML documents makes them easily compressible even with general-purpose compressors. Among the streamlined compression methods (which are highly welcome in web applications), the two most important families are LZ77 [20] and PPM algorithms [4].

The former ones parse the text into literals and matches to text sequences seen earlier. There are plenty of LZ77 variants, differing, e.g., in literal and match encoding, or parsing strategies. Let us mention only two of them, Deflate [6], used in the extremely popular zip and gzip formats, and LZMA [12], the current state-of-the-art in LZ77 compression.

Unlike LZ77 compressors, the algorithms from prediction by partial matching (PPM) family [4,18] encode characters on their context basis. This approach usually provides higher compression, but may be unsatisfactory for texts with many long matches, mostly due to physical (CPU time, memory occupation) limitations imposed on PPM models in practice.

Yet another line of general-purpose compressors are those employing the Burrows–Wheeler transform (BWT) [2], interesting in their own, but their block-oriented nature seriously hampers their use in web applications.

Still, XML can be compressed better with dedicated schemes. The first truly successful dedicated XML compressor was XMill [9], compressing data-centric documents to about half their gzipped size, and to even less in a human-assisted regime. It was achieved thanks to three features of XMill. The first was splitting XML document content into three distinct parts containing respectively: element and attribute symbol names, plain text, and document tree structure. Every part has different statistical properties, therefore it helps compression to process them with separate models. The second idea was to group contents of same XML elements into so-called containers. In this way similar data are stored together, helping compression algorithms with limited history buffer, such as LZ77 derivatives. Finally, each container was compressed with a dedicated method, exploiting the type of data stored within it (such as numbers or dates). What makes

this feature not so useful is that XMill requires the users themselves to choose methods to encode specific containers. Such human-assisted compression can hardly be regarded practical. XMill originally used gzip to compress the transform output. Although newer versions added support for BWT-based bzip2 and PPM implementations, yet in these modes XMill succumbs to other programs employing such algorithms, for instance, those described below.

The first published XML compression scheme based on a high-compression PPM algorithm was XMLPPM [3]. XMLPPM replaces element and attribute names with their dictionary indices, removes closing tags as they can be reconstructed in a well-formed XML document only provided their positions are marked. The most vital trait of XMLPPM is multiplexed hierarchical modeling which consists in encoding data with four distinct PPM models: one for element and attribute names, one for element structure, one for attribute values, and one for element contents. In order to exploit some correlation between data going to different models, the previous symbol, regardless of the model it belongs to, is used as a context for the next symbol.

XMLPPM was extended into SCMPPM [1], in which a separate PPM model is maintained for every XML element. This helps only in case of large XML documents, as every PPM model requires a due number of processed symbols to become effective. The main flaw of SCMPPM is its very high memory usage.

Exalt [19], presented in 2004 by Toman, was a pionieering work in compressing XML by inferring a context-free grammar describing its structure. Among the following solutions along these lines, the most successful was probably XAUST [8] by Hariharan and Shankar, employing finite-state automata (FSA) to encode XML document structure. In XAUST, element contents are put into containers and encoded incrementally with arithmetic encoding based on a single statistical model of order 4 (i.e., treating at most four preceding symbols as the context for the next one). The published results show that XAUST beats XMLPPM on some test files, yet the great drawback of this scheme is that it requires the XML document to have a document type definition (DTD) to which it conforms, as the compressor needs to construct the FSA.

There are many XML compressors that allow search without a need for full XML document decompression (references can be found e.g. in [7]). Nevertheless, these compressors achieve compression ratios only comparable to XMill , with the positive exception of XBzip, which is based on a variant of the Burrows–Wheeler transform adapted to tree structures [7]. In this line of our research we focus on publishing XML datasets on the Web, not on processing them server-side, therefore we omit the design issues related to fast query execution. See [17] for a description of another scheme suited for such applications, offering fast query processing at the cost of compression ratio.

3 Transform-Based Approach to XML Compression

There are three ways to improve the efficiency of general-purpose compression algorithms on XML documents: use a dedicated algorithm (e.g., XAUST); modify

an existing algorithm to exploit the specifics of XML (e.g., XMLPPM); or, define a transform whose output can be handled better by the general-purpose compression algorithms than actual XML is (e.g., XMill).

The first of the aforementioned approaches leads to solutions which are perfectly tuned for XML, but it requires large amount of work, and, after implemented, it cannot benefit from the improvement in general-purpose compression. In the second approach the XML specialization can go almost as far as in the first one, and the amount of work is (theoretically) smaller as we modify existing code. Still, additional work is required to make such scheme benefit from improvements in the underlying algorithm, and not every algorithms source code is available and free to modify. In the third case the role of the transform is to transcode an XML document in a way exposing its redundancy to the subsequent general-purpose compression algorithm. The level of specialization is much lower, as it is impossible to expose every kind of redundancy to the back-end compressor. Still, once implemented, this sort of solution can be used with any general-purpose compressor, no matter if it existed when the transform was implemented or whether its source code is available. However, being ready to work with any general-purpose compression algorithm does not mean that it cannot be tuned for a specific one, or that a specific algorithm cannot be tuned for the transform output.

The transform can help the subsequent compressor in several ways: by efficient representation of XML structure, better to handle than the robust textual tagging; by replacing with mere flags those tokens which can be reconstructed based on XML constraints (like closing tags); by separating contents of different XML elements, so that the algorithms which prefer short-range correlation (like the LZ77-based) can do better; by separating content tokens which have low frequency and break the context for the subsequent data; by efficient representation of content tokens which have long textual form (like words, numbers or dates).

The logical separation of the transform stage from the compression stage does not imply two levels of coders/decoders; for most practical purposes, a better solution is that a single coder/decoder should perform both stages. Moreover, most web applications require the decoder to support streaming. That is, the operations of decompression and reverse transform should be interleaved for small blocks of data. This brings additional constraints for the transform architecture.

4 XWP: A Web-Compression-Oriented XML Transform

In this section we present XWP (short from XML-WRT+PPM), a compression-oriented XML transform designed with two goals in mind: very high compression efficiency and suitability for web applications.

XWP is derived from the root of XML-WRT [15], a basic compression-oriented XML transform lacking the desired features of XWP. Another line of research was aimed at fast decompression capability through the use of LZ-based back-end compressors (XWRT2) [16], and its variant offering partial decompression for

faster search (QXT) [17]. This line lead to solutions which are not perfect for web usage, as XWRT2 is an off-line transform, and QXT reduces compression ratio for the sake of partial decompression which is irrelevant if the whole document is to be decompressed anyway.

Since the introduction of the XMLPPM compression scheme in 2001 [3], the PPM algorithm has been used in several successful XML-specialized compressors (including SCMPPM [1] and XBzip [7]).

Unfortunately, pairing up XML-WRT with PPM was not an easy task. Most of the techniques developed for XWRT2 worked only for LZ-based compressors, and could even hurt compression efficiency of compressors of other type. This is due to the differences between PPM and LZ77-based compressors behavior, as given in Table 1.

Table 1. A comparison of LZ77 and PPM compressors

	LZ77	PPM
Short sequences	Will not be substituted unless longer than minimum match length	Encoded efficiently thanks to context modeling
Long sequences	Encoded efficiently provided the original sequence is within the sliding window	Coding efficiency limited by maximum PPM order bound
Distance between a match and its source	Longer distance decreases compression efficiency (as the distance must be encoded); distance larger than the sliding window size makes it impossible to find a match	Practically irrelevant
Inserting unusual symbols in typical contexts	Irrelevant, as a contextless model is used to encode single symbols	Coding efficiency hurt as the context is disrupted

4.1 Transform Components

The backbone of the proposed transform is to replace the most frequent words with references to a dictionary. XWP works in two passes. The dictionary is obtained in a preliminary pass over the data, and contains sequences of length at least l_{min} characters that appear at least f_{min} times in the document. The dictionary is sorted by word frequency and divided into four groups in a way described in the next section. Then each group of words is sorted alphabetically. The complete dictionary is front compressed [14] and stored within the compressed file, thus making the reverse operation faster. We set $l_{min} = 2$ and $f_{min} = 64$, as these values gave good results for most files used in the experiments.

We have also tried a fully dynamic (one-pass) transform variant, but it gives much worse compression ratio. Also using a static dictionary (i.e., a single dictionary embedded in the coder/decoder) is questionable because of the practical impossibility to select a set of words relevant across a wide range of real-world XML documents.

In the second pass, the parsed data items are encoded in a byte-oriented manner (words with a prefix code; numbers, IP numbers, dates, and times with respective coding schemes; details will be presented later), and then compressed with a context-based compression algorithm and written to disk. We chose two algorithms of this kind: PPMVC and FastPAQ, which are described in detail in the following sections.

Our notion of a "word" is broader than its common meaning. Namely, XWP dictionary contains items from the following classes:

- ordinary words – sequences of lowercase and uppercase letters (a–z, A–Z) and 128–255 (which supports, e.g., all languages with a Latin-based alphabet);
- start tags – sequences of characters that start with `<`, contain letters, digits, underscores, colons, dashes, or dots, and end with `>`. Start tags can also include one or more preceding spaces as XML documents usually have regular arrangements of the lines in which individual tags very often begin in the same column, preceded with the same number of spaces,
- URL address prefixes – sequences of the form http://domain/, where domain is any combination of letters, digits, dots, and dashes,
- e-mails – patterns of the form login@domain, where login and domain are any combination of letters, digits, dots, and dashes,
- words in form "&data;", where data is any combination of letters, representing XML entities,
- special digrams – sequences `="` and `">`, which appear very frequently with attribute values,
- runs of spaces – sequences of spaces that are not followed by start tags (for documents with regular layouts).

XWP also employs several heuristics known from other schemes, such as replacing end tags with a flag, and omitting single spaces before encoded words, as they can be reconstructed on decompression provided only the positions where they should not be inserted are marked with a respective flag.

The transform can handle any XML documents with 7-bit or 8-bit encodings. It can be adapted to handle 16-bit encoded documents as well.

XWP is truly lossless, i.e., the decoded file is an exact copy of the encoded one. Some XML compressors ignore the document layout (e.g., trailing spaces), but preserving it may be useful for human editors of a document. Moreover, the exact fidelity of the decompressed document allows to use a cyclic redundancy check or hash functions to verify the document integrity.

4.2 Token Encoding

Succinct word encoding appears to be the most important idea in our scheme. Dictionary references are encoded using a byte-oriented prefix code, where the

length varies from one to four bytes. Although it produces slightly longer output than, for instance, Huffman coding, the resulting data can be easily compressed further, which is not the case with the latter. Obviously, more frequent words should be assigned shorter codewords. What is less obvious is that the dictionary encoding should be different for back-end LZ77 and PPM compressors [14].

The codeword alphabet consists of symbols very rarely used in most XML documents: byte values above 127 and most values in range 0–31, plus a few more. If, however, one of those symbols occurs in the document, and is not part of an encoded word, the coder prepends it with a special escape symbol.

The XWP dictionary encoding is less dense than the LZ77-optimized one used in XWRT2 [16], and uses non-intersecting value ranges for different codeword bytes. The four ranges are of size w, x, y and z, respectively.

Namely, we have w one-byte codewords, $x \cdot w$ two-byte codewords, $y \cdot x \cdot w$ three-byte codewords, and $z \cdot y \cdot x \cdot w$ four-byte codewords. The first byte of a codeword unambiguously defines its length. For instance, when encoding a two byte long codeword, a byte from the range of size x will be followed by a byte from the range of size w. The parameters w, x, y, z are selected according to the size of the created dictionary, with the principle of maximizing the number of short codewords.

Certain non-word patterns are also encoded in a special way, and their collection has been broadened compared to XWRT2 [16]. Numbers (e.g., “1978”, “102.01”, “1.2”), IP numerical addresses (e.g., “129.121.34.214”), dates (e.g., “1980-02-31”, “01-MAR-1920”), times (e.g. “11:30pm”, “23:20”, “23:30:59”), and numerical ranges (e.g. “1020-1042”) are encoded densely as binary numbers (base 256). For example, any integer in the range from 1900 to 2155 is considered to be a year, and encoded on a single byte. Numbers from 1 to 12 followed by the suffix “am” or “pm” (e.g., “11:30pm”) are interpreted as times, and encoded on three bytes: the first byte is a flag signaling a time pattern (conforming to the presented notation), the second is the hour in 24-hour convention, the other specifies the minutes. Some documents, e.g. containing bibliographical information, are flooded with page ranges. They are usually in the format x–y, and usually $0 < y - x < 256$. If this happens, four bytes are enough to encode such a range: a flag, two bytes for x, one byte for the difference $y - x$. Several other pattern types are handled similarly. Note that this is an extension of the features found in XMill. As opposed to it and the other mentioned schemes, XWP automatically detects such data types, with neither DTD nor user assistance.

4.3 Back-End Compression Algorithms

To avoid confusion, we shall refer to the XML transform preceding statistical compression as XWP, and to the entire compression scheme (including the XWP transform and back-end compression) as XWRT3 (with XWRT1 standing for the original, non-specialized scheme [15], and XWRT2 for the scheme specialized for LZ-based compressors).

The XWRT3 implementation provides two back-end compressors to choose: PPMVC and FastPAQ. Both have been implemented by the first author. PPMVC is faster while FastPAQ provides better compression ratios. We present both algorithms in the following subsections.

PPMVC. PPM is an adaptive statistical compression method. A statistical model accumulates counts of symbols (usually 8-bit characters) seen so far in a given context. Thanks to that, an encoder can predict probability distribution for new symbols from the input data. Increasing the context length is beneficial for encoding symbols known in a given context, but amplifies the problem of efficient encoding of the symbols yet unseen.

XML data contain long repeated strings. These data are compressed with most PPM variants in a way far from optimal, as the highest order used by e.g. Shkarin's PPMd is only 16. In 2004 Skibiński and Grabowski [13] presented the PPMVC algorithm (PPM with variable-length contexts), a variant of PPM* [5] adapted to cooperate with modern PPM mechanisms. PPMVC extends the character-based PPM with LZ-style string matching.

The PPMVC mechanism works on maximum order contexts only; in shorter contexts the current symbol is encoded with an ordinary PPM model (namely, Sharkin's PPMd [18] model was used). In PPMVC (called PPMVC2 in [13]) each maximum order context holds a pointer to reference context (the previous occurrence of the context) and the minimum left match length. The left match length (*LML*) is the length of the common part of the active context and the reference context. *LML*, by definition, is always at least as large as the maximum PPM order. The right match length (*RML*) is defined as the length of the matching sequence between symbols to encode and symbols followed by the reference context.

When a character is encoded from the maximum order context, the longest *LML* is evaluated, using the last contexts appearance. If it is below the minimal left match length (*minLML*), then the encoder uses ordinary PPM encoding (without emitting any escape symbol). In the other case, the encoder uses this context to find the *RML* (zero or more) and encodes it using an additional global *RML* model. Length is truncated to the closest multiple of the parameter d. For example, if there is a match of length 14, and d is 3, then only the first 12 characters of the match are encoded (the truncated characters might however be part of the next *RML*).

The higher is the order, the better is the PPM effectiveness in deterministic contexts, so the parameters d, *minLML* and *minRML* must respectively be greater. In the PPMVC variant implemented in XWRT3 (i.e., tuned for XWP output), the parameter d was set to $2 \cdot (maxOrder + 1)$. The other parameters, *minLML* and *minRML*, are set to $2 \cdot (maxOrder - 1)$ and $2 \cdot d$, respectively.

The decoder basically mimics the steps of the encoder.

FastPAQ. PAQ [10] is a family of compressors, originally developed by Matthew Mahoney, based on context modeling. As opposed to most PPM variants, PAQ works on the bit level, thus avoiding the fundamental zero-frequency problem

[4]. Mahoney's algorithm uses several predicting models, including (as in PAQ8 [11]), e.g., order-n models (n up to 16), similar to the one used in PPM; a string matching model, similar to one used in the LZ77 algorithm; and a number of text, multimedia, tabular, or binary data oriented models (e.g., for x86 executables or BMP images).

Mixing the prediction of individual models in PAQ8 is performed with several neural networks. The outputs of these networks are combined using a second-level neural network. Before submitted to an arithmetic coder, the outputs go through two stages of adaptive probability maps (APM). The APM mechanism is related to the secondary symbol probability estimation (SSE), known from the PPMII algorithm [18]. It updates the probability considering previous experience and the current context.

The main disadvantage of the PAQ8 algorithm are very high resource requirements, both in memory and CPU computing power. It makes this algorithm totally impractical, at least at current hardware. This is why we prepared FastPAQ, a stripped-down version of PAQ8, intended to improve compression and decompression speed. From PAQ8 we have left only the order-n models, and we have also simplified APM stages, in overall making it more similar to PPM. FastPAQ is still much slower than fast PPM variants, but achieves better compression ratios.

5 Experimental Results

To compare the performance of our scheme with existing, XML-specialized and general-purpose, compressors, we measured the compression ratio and compression and decompression times on individual files from *Wratislavia XML Corpus* (http://www.ii.uni.wroc.pl/~inikep/research/Wratislavia/index.htm), created to provide a diverse collection of large XML files: it comprises textual, numerical and mixed-content documents, most of which have been used for benchmarking in earlier papers.

The test machine was an Intel Core 2 Duo E6600 2.40 GHz system with 1 GB memory and two Seagate 250 GB SATA drives in RAID mode 1, running Windows XP 64-bit edition. We tested the current version of software implementing our transform, XWRT 3.1, available from http://www.ii.uni.wroc.pl/~inikep/research/XML/XWRT31.zip.

In Table 2, the compression results obtained for XWP-transformed datasets using PPMVC and FastPAQ as back-end compressors are compared to those achieved by the same compression algorithms on the datasets in their original form. Existing XML-aware compressors are represented in the results with the fast XMill 0.9, and the current state-of-the-art XML compressors, XMLPPM 0.98.2 and SCMPPM 0.93.3. We used maximum compression settings (-9/-l9) for all these compressors; also, the preserve spaces option was turned on, where it was available. We were not able to test XBzip, as it crashes on files larger than 100 MB on our test machine. For a reference, we also present results of the LZ77-optimized version of XWRT (2.1) [16]. The same settings of model order

Table 2. Compression test results

File	XWRT2 Defl.	XWRT2 LZMA	XMill PPM	XML-PPM	SCM-PPM	PPM-VC	XWRT3 PPMVC	F.PAQ	XWRT3 F.PAQ
all_shakes	1.481	1.348	1.459	1.295	1.293	1.245	1.199	1.219	1.180
dblp	0.864	0.747	0.863	0.802	0.693	0.726	0.687	0.638	0.593
enwikibooks	1.734	1.505	1.707	1.621	1.621	1.555	1.471	1.328	1.257
enwikinews	1.590	1.300	1.533	1.379	1.398	1.277	1.191	1.149	1.080
lineitem	0.276	0.243	0.262	0.261	0.242	0.346	0.243	0.236	0.226
swissprot	0.475	0.388	0.455	0.416	0.417	0.392	0.362	0.389	0.312
uwm	0.315	0.278	0.310	0.259	0.274	0.313	0.240	0.254	0.228
Average	0.962	0.830	0.941	0.862	0.848	0.836	0.770	0.745	0.697
Comp. time	49.22	141.48	75.34	90.54	180.01	90.89	117.83	2321.3	924.37
Dcmp. time	15.88	20.05	78.92	–	179.37	97.94	96.61	2324.1	890.35

Compression and decompression bitrates are expressed in output bits per input character (lower values are better). Compression and decompression times are given in seconds. XMLPPM failed to decompress two of the test files (*enwikibooks* and *enwikinews*).

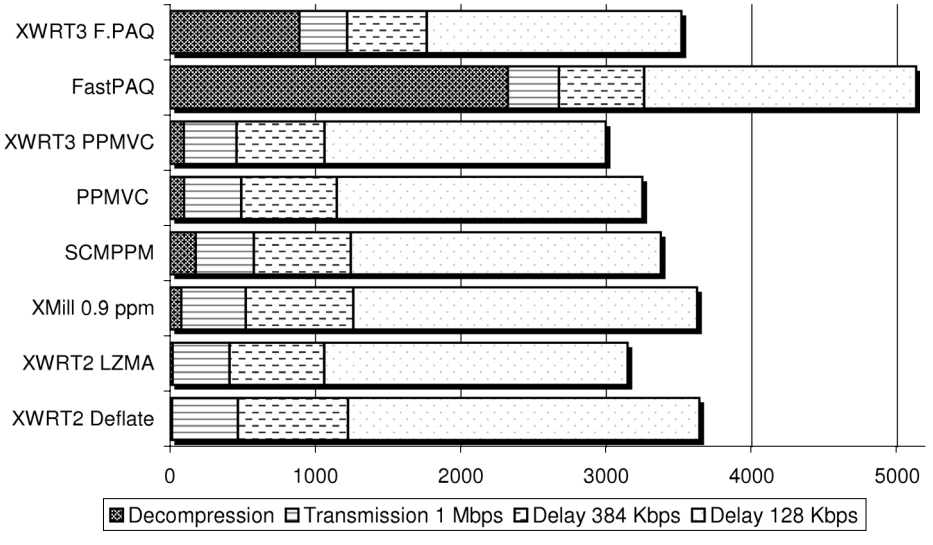


Fig. 1. Decompression and transmission times (s)

(8) and size (64 MB) were used for PPMVC and XWRT3-PPMVC. FastPAQ (also in XWRT3-FastPAQ) memory usage was limited to 74 MB.

As Table 2 shows, XWP transform managed to improve the compression ratio of both PPMVC and FastPAQ – notice that the original compression ratios attained by these algorithms were in almost all cases already better than those of the remaining XML-specialized compressors. XWRT3-FastPAQ sets the new state-of-the-art in XML compression; however, although its processing times are

over two times shorter than the original FastPAQs, they are still far too long for practical applications, especially for the Web. For a more practical-minded comparison of the results, see Fig. 1 showing the time required to download the whole test suite and have it decompressed (on the test machine). The bars are divided into sections representing decompression time, transmission time over a 1 Mbps connection, the transmission delay if a 384 Kbps connection was used instead, and the additional delay from using a 128 Kbps connection.

XWRT3-PPMVC appears to be the best available solution for transmission speeds of 384 Kbps and below. For a 1 Mbps connection it succumbs only to our previous scheme, XWRT2. Yet the streamlined decompression introduced with XWP may be started as soon as the compressed header is downloaded (thus reducing the users perceived document acquisition time), which is not the case with XWRT2, requiring off-line decompression. Also, the documents compressed with XWRT3 are shorter (see Table 2 again), thus saving server storage space and available bandwidth.

6 Conclusions

We presented a lossless XML transform designed for subsequent compression. The main components of XWP are semi-dynamic dictionary of frequent alphanumerical phrases (not limited to "words" in a conventional sense), and binary encoding of popular patterns, like numbers, dates or IP addresses. We believe that those ideas, albeit quite familiar in the XML compression community, have not been fully exploited in earlier research.

The transform, XWP, together with a suitable PPM variant (PPMVC) enables to outperform the compression ratio of the state-of-the-art SCMPPM algorithm by 9% on average, using only up to 86 MB of memory. This result can be improved by another 9%, when PPMVC is replaced by a stronger but much slower back-end compressor, FastPAQ. The transform requires no metadata (such as XML Schema or DTD) nor human assistance, and its decoding is streamlined, which is a precious feature in web applications. We also show that using XWP with PPMVC for XML documents gives the greatest time saving to an end user, measured as the time to retrieve the (compressed) document and to decompress it, if the network transfer is slow (up to 384 Kbps), while for moderate-speed transfers (1 Mbps) it loses only to another variant of our transform with LZMA compression as the back-end.

References

1. Adiego, J., de la Fuente, P., Navarro, G.: Using Structural Contexts to Compress Semistructured Text Collections. Information Processing and Management 43(3), 769–790 (2007)
2. Burrows, M., Wheeler, D.J.: A block-sorting data compression algorithm. SRC Research Report 124. Digital Equipment Corporation, Palo Alto, CA, USA (1994)

3. Cheney, J.: Compressing XML with multiplexed hierarchical PPM models. In: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, USA, pp. 163–172 (2001)
4. Cleary, J.G., Witten, I.H.: Data Compression Using Adaptive Coding and Partial String Matching. IEEE Trans. on Comm. 32(4), 396–402 (1984)
5. Cleary, J.G., Teahan, W.J., Witten, I.H.: Unbounded length contexts for PPM. In: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, USA, pp. 52–61 (1995)
6. Deutsch, P.: DEFLATE Compressed Data Format Specification version 1.3. RFC 1951 (1996), http://www.ietf.org/rfc/rfc1951.txt
7. Ferragina, P., Luccio, F., Manzini, G., Muthukrishnan, S.: Compressing and searching XML data via two zips. In: WWW. Proceedings of the International World Wide Web Conference, Edinburgh, Scotland, pp. 751–760 (2006)
8. Hariharan, S., Shankar, P.: Compressing XML documents with finite state automata. In: CIAA. Proceedings of the Tenth International Conference on Implementation and Application of Automata, Antipolis, France, pp. 285–296 (2005)
9. Liefke, H., Suciu, D.: XMill: an efficient compressor for XML data. In: Proceedings of the 19th ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, pp. 153–164 (2000)
10. Mahoney, M.: Adaptive Weighing of Context Models for Lossless Data Compression. Technical Report TR-CS-2005-16, Florida Tech., USA (2005)
11. Mahoney, M.: The PAQ Data Compression Programs (2007), http://www.cs.fit.edu/~mmahoney/compression/paq.html
12. Pavlov, I.: LZMA Software Development Kit (2007), http://www.7-zip.org/sdk.html
13. Skibiński, P., Grabowski, Sz.: Variable-length contexts for PPM. In: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, USA, pp. 409–418 (2004)
14. Skibiński, P., Grabowski, Sz., Deorowicz, S.: Revisiting Dictionary-Based Compression. Software – Practice and Experience 35(15), 1455–1476 (2005)
15. Skibiński, P.: Grabowski, Sz., Swacha, J.: Fast transform for effective XML compression. In: CADSM. Proceedings of the IXth International Conference, Lviv, Ukraine, pp. 323–326 (2007)
16. Skibiński, P.: Grabowski, Sz., Swacha, J.: Effective Asymmetric XML Compression. Software – Practice and Experience (to appear)
17. Skibiński, P., Swacha, J.: Combining efficient XML compression with query processing. In: ADBIS 2007. LNCS, vol. 4690, pp. 330–342. Springer, Heidelberg (2007)
18. Shkarin, D.: PPM: One step to practicality. In: Proceedings of the IEEE Data Compression Conference, Snowbird, UT, USA, pp. 202–211 (2002)
19. Toman, V.: Syntactical compression of XML data. In: Presented at the doctoral consortium of the 16th International Conference on Advanced Information Systems Engineering, Riga, Latvia (2004), http://caise04dc.idi.ntnu.no/CRC_CaiseDC/toman.pdf
20. Ziv, J., Lempel, A.: A Universal Algorithm for Sequential Data Compression. IEEE Trans. Inform. Theory 23(3), 337–343 (1977)

Improving Semantic Search Via Integrated Personalized Faceted and Visual Graph Navigation

Michal Tvarožek, Michal Barla, György Frivolt,
Marek Tomša, and Mária Bieliková

Institute of Informatics and Software Engineering, Faculty of Informatics
and Information Technologies, Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia
{Name.Surname}@fiit.stuba.sk

Abstract. Growing need for information retrieval, information processing, and the associated need for navigation in existing information spaces resulted in several approaches that aim to improve efficiency of the respective user tasks. However, problems related to user navigation and orientation in large open information spaces still persist possibly due to increasing demands and the imperfections of individual approaches. We propose an integrated search and navigation solution that takes advantage of the faceted browsing paradigm and visual navigation in graphs both extended with support for automatic personalization based on user context also taking advantage of a user's social network. The proposed solution is primarily evaluated in the domain of scientific publications, i.e. digital libraries, with possible extensions to other application domains.

1 Introduction

Growing information processing requirements continuously challenge our information retrieval, navigation and visualization capabilities. Effective use of the available information is becoming ever more difficult as the size, changeability and complexity of the information space increase as do the diversity of users and their requirements.

Consequently, issues such as the infamous navigation problem, high navigation recurrence, information overload or insufficient personalization support arise and significantly decrease overall user productivity in demanding information processing tasks [1]. The typical steps performed by users when seeking information are the query, selection, navigation and query modification. However, most current approaches provide very little support for the crucial navigation step where users locate the desired information. For example, current web search engines are able to locate a set of good starting points for navigation yet offer users no navigation support that would help them in realizing their goals [1].

Moreover, several studies indicated that keyword-based queries tend to be short (two to three words) [2] and that advanced search interfaces of keyword-based search engines are impractical to use [3], while also being unsuitable for

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 778–789, 2008.

browsing and exploratory tasks [4]. Thus more advanced search and navigation paradigms such as faceted and visual graph navigation were explored:

- *Faceted navigation* used in faceted browsers is based on the use of faceted classification – an orthogonal multidimensional classification of information artifacts, which was originally developed in library sciences [5]. Its basic principle lies in the use of facets, describing individual properties of instances in an information space, to specify the desired properties of instances in the visible information space. The final search query combines restrictions from individual facets via the logical AND function resulting in an unordered list of instances that satisfy all specified restrictions.
- *Graph navigation* is based on a graph representation of the information space. A graphical visualization of the respective (sub)graph of the information space is used, where individual nodes are annotated to aid user navigation. Two modes of operation can be used. Vertical navigation in the graph based on, for example hierarchical clustering of instances, where multiple clustering criteria can be used to select subspaces of the original information space similarly to faceted navigation. Horizontal navigation in the graph allows users to explore the relations between individual instances (nodes), typically employing either a local cluster, centered on a specific instance or a window, which displays instances and relations up to a certain distance.

In this paper we aim to improve existing navigation approaches by combining personalized faceted navigation with visual navigation in graphs, taking advantage of ontologies and users' social networks. Section 2 describes current approaches that improve upon existing navigation methods, while our extensions to faceted navigation and its corresponding extension with visual navigation in search results and associated instances are described in sections 3 and 4. We describe our evaluation in section 5 and summarize the contribution in section 6.

2 Related Work

The field of generic information retrieval has already been explored in much detail resulting in several methods and commercially successful search engines. However, the specific area of navigation also in Semantic Web data has not yet been sufficiently explored. Thus our focus is information navigation in ontological instances taking advantage of semantic metadata available in OWL ontologies.

In [6], the authors describe BrowseRDF – a faceted browser for RDF data, which generates the faceted browsing interface from the supplied RDF data. They extend the faceted query model with new types of queries (e.g., existential, inverse existential, inverse join) and define metrics and algorithms for automatic facet ranking based on statistics computed from the source RDF data to prevent information overload. However, since only RDF data without additional metadata are used, all properties are used as facets disregarding their semantics (e.g., numbers, dates, enumerations, hierarchies, direct/indirect properties of the desired search results) making efficient use for large information spaces impractical.

Similarly, the Semantic Web portal tool OntoViews [7] publishes RDF content via a predefined faceted browser interface and provides the user with a content-based search engine and link generation/recommendation based on relationships between ontological concepts. OntoViews also presents results in clusters based on the semantic similarity of instances (i.e., sharing some common properties).

Authors in [4] stress the importance of user interface usability and divide the faceted search process into three phases. The *opening* gives users a broad overview of the scope, size and content of the collection, the *middle game* allows users to narrow down the result set by refining the query, the *end game* shows the final search result and allows users to navigate laterally through the collection.

Neither of these solutions however provides personalized features based on individual users' characteristics nor their context. Furthermore, the faceted classification of an information domain can be somewhat difficult to understand resulting in orientation problems, while the access to popular items possibly nested deep within a classification hierarchy can be tedious and impractical. Moreover, from the Semantic Web perspective, none of these approaches take advantage of semantic metadata (i.e., OWL), which in addition to instances also describe the structure of the information domain, thus being of only limited use for search and navigation in OWL ontologies.

Navigation in a graph heavily depends on the adequate visualization of the graph, i.e., on the use of suitable graph drawing styles. In *force-directed* algorithms nodes repel each other while edges are interpreted as springs exerting attracting forces on the nodes they connect. The algorithm then iteratively moves some nodes according to the overall force acting on each of them until a reasonably stable configuration is found [8]. *Layered drawing* is another drawing style where the majority of the edges follows some overall direction that shall be emphasized by drawing as many edges as possible in one specific direction, e. g., form top to bottom.

Drawing clustered graphs is more complicated because of the additional inclusion hierarchy. Possible solutions to use an additional dimension or to emphasize the underlying graph by drawing the inclusion tree as nested regions (inclusion edges are not drawn explicitly, but they are visualized through the geometric nesting of the corresponding regions – mostly rectangles or circles).

Another vital issue is the visualization of graph changes – how efficiently can users follow the expansion or contraction of the graph as they they interact with it. With poor visualization it takes the user considerable time to become familiar with the graph's drawing again. This means that searching for details of interest has to be performed without losing orientation within the graph. According to [9], this is one of the main problems, when exploring large structures. Therefore, often different representations for overview and detail are linked together in such a way that manipulating one view automatically updates the others. When the user searches for some structures of interest in one view, these structures are also accentuated in the other views, e.g. by highlighting or color-coding. Working with different views at once, the user can effectively change the representations and choose the best for the task at hand.

In our approach, we build upon existing solutions – we assume an ontological domain and user model representation in accordance with the Semantic Web initiative [10], and propose an integrated navigation approach combining faceted navigation and visual navigation in graphs with support for personalization based on an automatically acquired user model.

3 Personalized Faceted Navigation

Faceted browsers provide graphical user interfaces which enable users to interactively select one or more restrictions in the set of available facets thus visually constructing (semantic) search queries via navigation instead of inventing and writing keywords. In practice, faceted browsers can be effectively used for faceted browsing of an information space without a specific goal, e.g. to get an overview of what information is available, or for faceted search if the specific properties of the desired search results are known, e.g. if journal papers on adaptive hypermedia and the Semantic Web not older than 3 years should be returned.

Some disadvantages of faceted browsers include a somewhat more complex user interface, the need to understand the used faceted classification and faceted browsing paradigm as well as the possibly difficult access to popular items, which might be nested deep in the classification hierarchy.

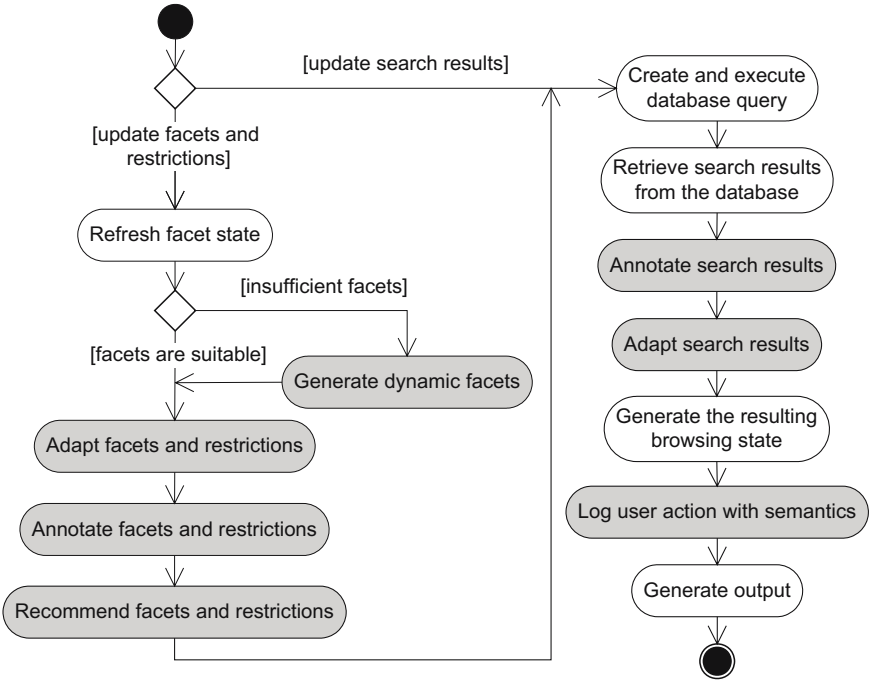


Fig. 1. Request handling of the adaptive faceted semantic browser, extensions in gray

We proposed personalization, adaptation, and the use of ontologies to address the aforementioned shortcomings of "classical" faceted browsers, and to improve user orientation and guidance, and overall search efficiency [11]. The request processing of our adaptive faceted semantic browser, which allows users to navigate in a domain ontology (in OWL format) and adapts the navigation based on an automatically acquired user model [12], is shown in Figure 1.

- *Dynamic facet generation* takes advantage of metadata stored in the domain ontology to create at run-time new facet descriptions relevant to the user's information goals and characteristics in the user model (Figure 1, center).
- *Facet and restriction adaptation, annotation and recommendation* adapt the set of available facets and restrictions based on the in-session user behavior and based on more long term user characteristics stored in the user model also considering the characteristics of other users. Individual facets are hidden or disabled if they seem less relevant to the current user task, or reordered based on their relevance. Restrictions are annotated with additional information (e.g., instance count or relevance based on the user model) and/or recommended (e.g., shown with different background color) (Figure 1, left).
- *Search results annotation and adaptation* improve user orientation and guidance by providing additional information about individual search results (e.g., how close is the topic of a paper to the user's interests) and displaying only relevant attributes of instances. (Figure 1, right)
- *User action logging with semantics* facilitates automatic user characteristics acquisition by logging evidence of user actions with the associated semantics via a user modeling server (Figure 1, bottom right).

We also enhanced the faceted browser with support for nested facets for indirect properties of instances, derived from ontological metadata describing the structure of the domain ontology (i.e., classes), and additional facet combination functions (e.g., OR, NOT).

Since relations between users represented as social networks can significantly improve the user model and thus indirectly improve user experience, we extend the existing relevance evaluation model of the adaptive faceted semantic browser [12] with support for social networks of users who may be associated via different relationships (e.g., friend, coauthor, colleague).

We use social network data to:

- recommend and annotate search results based on social networks of individual users (e.g., sorting of results based on others' ratings);
- recommend and annotate restrictions based on social networks of individual users (e.g., highlighting restrictions popular with the user's colleagues);
- create new facets and restrictions based on social networks of individual users (e.g., a facet which restricts the information space to publications which were visited and/or rated by the user's colleagues).

4 Integrating Visual Navigation with Faceted Browsers

Faceted browsers typically present detailed information about search results in tables or simple textual overviews. However, this approach becomes ineffective for more complex information spaces, which consist of many tightly interconnected instances (e.g., data in OWL ontologies).

We employ visual navigation in graphs to present data about individual search results (ontological instances) and their respective associated instances. For example, in the domain of scientific publications, papers might represent search results, while authors, conferences, or individual research fields might represent associated instances.

Ontologies serve as initial data sources for both faceted navigation and navigation in graphs which use the semantics of the stored information. Furthermore, we take advantage of personalization based on the user context, which is automatically constructed, e.g., based on observed user behavior by specialized user modeling server and the corresponding user modeling agents, or based on the used device.

Our integrated navigation approach is outlined by these steps:

1. The user starts a navigation session by using the faceted browser interface and narrowing down the visible information space via the use of facets and restrictions.
2. The list of search results is displayed either in "classical" textual visualization or using a personalized hierarchically grouped textual or visual view based on a given grouping/clustering criterion. During this stage, the user can either further refine the query via facets or by selecting a group/cluster of search results.
3. Once the level of individual instances is reached, the user can view detailed information about specific instances and their associated instances using visual navigation in the graph starting from the selected search result instance. Alternatively, a (nested) "classical" textual table of attributes and their values can be used.

4.1 Hierarchical Search Results Refinement

Currently the most predominant browsing model is based on hierarchically organized information spaces. Multidimensional faceted navigation integrated with hierarchical search results refinement introduces new alternative views on the available information spaces. Moreover, a visual view of the information space restricted via facets, based on a cluster hierarchy, improves user orientation.

We visualize the internal structure, i.e. the relations between the nodes in clusters (see Figure 2) by showing related vertices closer to each other. Since we visualize only two levels of clustering simultaneously – the current level and the level directly below, multiple layers are not mixed together and thus do not confuse users. This is further supported by visualizing the relations between clusters on the current level only implicitly via the structure of the next level.

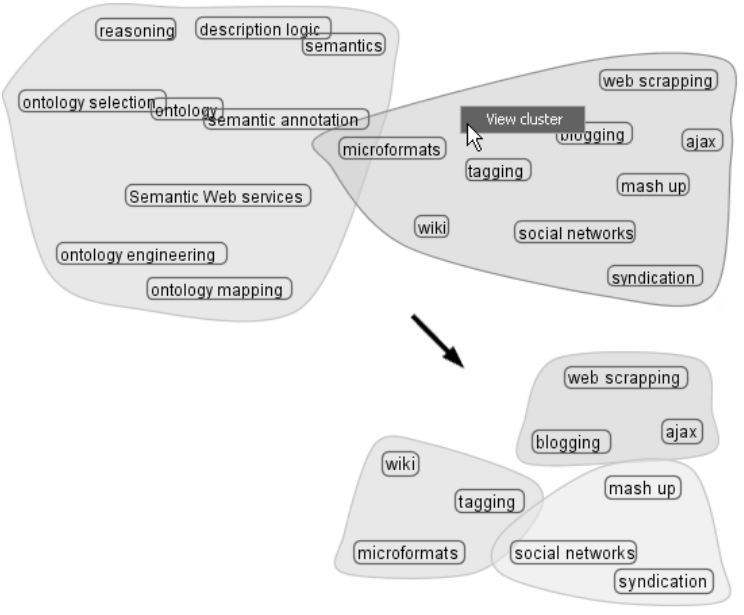


Fig. 2. Example of visual graph navigation in clusters. After selecting the right cluster (top), the view shows its internal structure divided into three smaller clusters (bottom).

Since users choose clusters for further navigation based on their annotations and the annotations of individual nodes, their proper choice is crucial for any successful visual navigation approach. Node and cluster annotations must be distinctive from other annotations of and should clearly describe the respective instance or instances (for a cluster).

In the domain of scientific publications, nodes are authors, which are connected by edges if a relation such as co-author, friend, colleague, or reviewer exists between them. The authors' publications are indexed using keywords and topics. Consequently, clusters are annotated with the keywords and topics of the respective authors' publications contained within a cluster.

These visualization techniques describing the features of nodes are used:

- *Tooltips*[1] provide extended information about nodes. In the domain of publications this information may include the most frequent authors or institutions, number of publications in the cluster or the most frequent conferences, workshops where the papers where presented.
- *Font size* emphasizes the size of the cluster (i.e., the number of instances).
- *Background color* used for personalization indicates suitability similarly as in [13]. For example, clusters often visited by related users can be recommended over less popular clusters.

[1] Tooltips are small boxes with supplementary information that appear when the user hovers the cursor over an item.

4.2 Horizontal Navigation in Instances

Horizontal navigation is traditionally considered as the main benefit of hypertext. Browsing instances/groups of instances visually by taking advantage of their similarity improves orientation and usability over the classical tabular view of faceted browsers.

In [14] we proposed incremental browsing of an ontology's structure. We use the same principle is for instances/group of instances visualized in a graph. In every moment only a specific part of the information space – a window is visible, based on the evaluation of the presented entities (e.g., ontology instances). The rest of the graph is visualized on demand when needed (e.g., based on further user clicks navigation). Entity evaluations are computed based on user interaction with the ontology (e.g., people browsing manually or applications).

4.3 User Modeling

The acquisition of user characteristics is fundamental for personalization and can be performed implicitly, e.g. by monitoring and analyzing an individual user's activity, by evaluating his or her activities in the context of social relations, or explicitly by getting feedback from users either by answering manually prepared or system generated questions based on a domain ontology, or by evaluating other forms of feedback (numeric content rating). All knowledge about the user is stored in a centralized user model separated from the rest of the adaptive system using a user modeling server.

Characteristics Based on Social Networks. We consider social networks to be an interesting and valuable source of characteristics about users since we can infer information from the size and nature of the social network or import user characteristics from other known members of the network.

One important property of the scientific publications domain is that users who search for publications can also be authors. Furthermore, there are social relations between authors which can be reflected in a social relation model between users and used for personalization.

Present day social portals usually provide only a single type of relation – *a-friend-of*. However, it seems appropriate to take advantage of typed relations (e.g., *a colleague*, *close friend*, *co-author* and *friend*) for potential publication authors and use these typed relations to detect additional user interests or to make recommendations. Individual relations can be strongly-typed (i.e., defined by an ontology) or labeled as tags created by users.

Detection of social relations can be automated or performed manually using forms. Certainty can be achieved only if users provide explicit confirmation of detected relationships.

Authorship identification. A possible exception is the co-authorship relation, which can be reliably detected without its explicit definition. However, we use explicit confirmation of publication authorship to eliminate cases of name coincidence or ambiguity.

Information about publication authorship is gained from users during their profile creation. If a user claims to be the author of at least one scientific publication he or she gets a list of publications from the information space he or she may have possibly authored. This list is obtained by selecting publications with author names similar to or derived from the name provided by the user, who then confirms publication authorship by selecting and checking publications from the offered list, thus updating the corresponding user model. If all selected publications share the same author instance, equality between the user identity and the author instance in the domain model can be assumed.

Usage of authorship information to solve the cold start problem. We derive initial weights for user interests from properties (keywords, index terms etc.) of publications for which a user confirms authorship. Furthermore, using co-authorship information we extend interests in the user model provided that people who co-authored some publication share some common interests.

Navigation-Related Characteristics. The filling-in of questionnaires is often considered an annoying and time-consuming activity. Therefore, we minimize user involvement required for the user modeling process and employ automated means of user model construction based on the monitoring and evaluation of user behavior.

Our approach is similar to other user modeling server based approaches [15]. Each (adaptive) application (either the faceted browser, cluster navigator or their combination) sends records about user actions (evidence) performed within the application to a user modeling server. Each record contains information about the respective event, its attributes and the consequence of the event (change of the user interface) as well as the timestamp and user–session identification [16].

Our logging approach is unique as it preserves the semantics of user interaction and allows for better separation (and thus re-usability) of user modeling functionality from the rest of the system. Interaction records are processed by user modeling agents which perform knowledge inference based on the collected evidence and update the user model respectively. Our inference agent is driven by a set of rules [17] which were devised to capture various interesting navigational, domain and system dependent or independent patterns, and to connect them with a set of consequences on user characteristics stored in the user model. The example of a rule (heuristic) is e.g., *result browsing* when user starts to explore the details of displayed instances *without* changing any restrictions in the faceted browser. We can reason that the currently selected restrictions represent user interests and incorporate this knowledge into the user model.

5 Evaluation

We evaluate the proposed approach in the domain of scientific publications in project MAPEKUS (`mapekus.fiit.stuba.sk`), where we developed the *Publication Presentation Portal – P3* based on the portal engine part of the Cocoon framework (`cocoon.apache.org`).

P3 provides its users with several options for exploring and navigating in the information space in accordance with the personalized presentation layer approach [18]. Users use either a special portlet for faceted browsing by means of the tool *Factic* (see Figure 3) or a portlet for visual graph navigation (tool *Cluster navigator*), which operates on domain clusters prepared by the *Clusterer* tool. Graph navigation is realized using Prefuse (`prefuse.org`) – a Java framework for graph visualization, which works as a Java applet. A third portlet allows users to use the combined approach of faceted and graph navigation. All presentation tools can take advantage of an ontology-based user model and contribute to it by logging user interaction via the user modeling server part of the system.

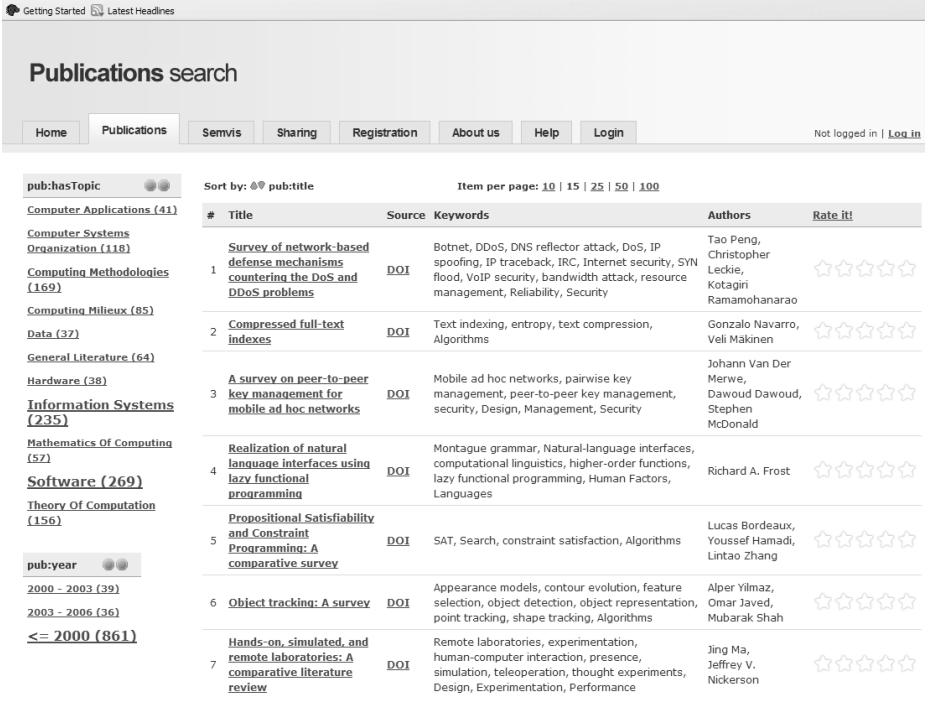


Fig. 3. Sample GUI of an enhanced faceted browser

Our underlying publication ontology was populated with metadata acquired from two well-known digital libraries (ACM DL and SpringerLink) and the scientific publications portal DBLP. We wrapped metadata about 83,000 publications (A-Box) (49,000 from ACM DL, 35,000 from SpringerLink, and 48,000 from DBLP) which were conceptualized into 390 classes in our ontology (T-Box). We used Sesame (`openrdf.org`) as an RDF data store.

For graph navigation, we extract instances of the class `Author` from the domain ontology, while the edges between the authors are defined by the property `isAuthorOf`. As the graph is too large, we applied off-line clustering of the

network. Raising the number of instances stored in Sesame led to serious degradation of user experience with the whole system since query response times increased exponentially. We consider the lack of proper and mature technologies for Semantic Web applications as a serious drawback for successful widespread acceptance of RDF/OWL and the Semantic Web.

6 Conclusions

We presented an original approach for personalized navigation via integrated faceted and visual graph navigation in Semantic Web data taking advantage of personalization together with a user's social network. Semantic Web technologies allowed us to enhance the traditional approaches either directly, for example by introducing features like dynamic facet generation or indirectly by enhancing the underlying data (e.g., intelligent clustering of semantically described data).

An important feature of our approach is adaptivity. Both navigation approaches log user interaction via the user modeling server and take advantage of the created user model to improve the user's browsing experience and offer navigation support and guidance.

Acknowledgment. This work was partially supported by the State programme of research and development "Establishing of Information Society" under the contract No. 1025/04, the Slovak Research and Development Agency under the contract No. APVT-20-007104, and the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07.

References

1. Levene, M., Wheeldon, R.: Navigating the World-Wide-Web. In: Levene, M., Poulovassilis, A. (eds.) Web Dynamics: Adapting to Change in Content, Size, Topology and Use, pp. 117–151. Springer, Heidelberg (2004)
2. Jansen, B.J., Spink, A., Bateman, J., Saracevic, T.: Real life information retrieval: a study of user queries on the web. SIGIR Forum 32(1), 5–17 (1998)
3. Markkula, M., Sormunen, E.: End-user searching challenges indexing practices in the digital newspaper photo archive. Inf. Retr. 1(4), 259–285 (2000)
4. Yee, K.P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: CHI 2003. Proc. of the SIGCHI conf. on Human factors in computing systems, pp. 401–408. ACM Press, New York (2003)
5. Wynar, B.S., Taylor, A.G.: Introduction to Cataloging and Classification. Libraries Unlimited Inc. (1992)
6. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for rdf data. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 559–572. Springer, Heidelberg (2006)
7. Mäkelä, E., Hyvönen, E., Saarela, S., Viljanen, K.: Ontoviews - a tool for creating semantic web portals. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 797–811. Springer, Heidelberg (2004)

8. Raitner, M.: Efficient Visual Navigation of Hierarchically Structured Graphs. PhD thesis, University of Passau (2004)
9. Schulz, H.J., Schumann, H.: Visualizing Graphs - A Generalized View. In: IV 2006. 10th International Conference on Information Visualisation, pp. 166–173 (2006)
10. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems 21(3), 96–101 (2006)
11. Tvarožek, M., Bieliková, M.: Adaptive faceted browser for navigation in open information spaces. In: WWW 2007: Proc. of the 16th Int. Conf. on World Wide Web, pp. 1311–1312. ACM Press, New York (2007)
12. Tvarožek, M., Bieliková, M.: Personalized faceted navigation for multimedia collections. In: SMAP 2007: Proc. of the 2nd Int. Workshop on Semantic Media Adaptation and Personalization (accepted, 2007)
13. Brusilovsky, P., Rizzo, R.: Map-based horizontal navigation in educational hypertext. In: HYPERTEXT 2002. Proc. of the 13th ACM Conf. on Hypertext and Hypermedia, pp. 1–10. ACM Press, New York (2002)
14. Bielikova, M., Jemala, M.: Incremental visual browsing of ontology structure based on metadata evaluation and usage. In: HYPERTEXT 2007. Proc. of the 13th ACM Conf. on Hypertext and Hypermedia, ACM Press, New York (2007)
15. Yudelson, M., Brusilovsky, P., Zadorozhny, V.: A User Modeling Server for Contemporary Adaptive Hypermedia: An Evaluation of the Push Approach to Evidence Propagatation. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007, Corfu, Greece. LNCS (LNAI), vol. 4511, pp. 27–36 (2007)
16. Andrejko, A., Barla, M., Bieliková, M., Tvarožek, M.: User Characteristics Acquisition from Logs with Semantics. In: Kelemenová, A., Kolář, D., Meduna, A., Zendulka, J. (eds.) ISIM 2007. 10th Int. Conf. on Information System Implementation and Modeling, Hradec nad Moravicí, Czech Republic, pp. 103–110 (2007)
17. Barla, M., Bieliková, M.: Estimation of User Characteristics using Rule-based Analysis of User Logs. In: UM 2007. Data Mining for User Modeling, Proc. of Workshop held at the Int. Conf. on User Modeling, Corfu, Greece, pp. 5–14 (2007)
18. Tvarožek, M., Barla, M., Bieliková, M.: Personalized Presentation in Web-Based Information Systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 796–807. Springer, Heidelberg (2007)

Author Index