

Joachim Hertzberg
Michael Beetz
Roman Englert (Eds.)

LNAI 4667

KI 2007: Advances in Artificial Intelligence

30th Annual German Conference on AI, KI 2007
Osnabrück, Germany, September 2007
Proceedings

 Springer

Lecture Notes in Artificial Intelligence 4667

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Joachim Hertzberg Michael Beetz
Roman Englert (Eds.)

KI 2007: Advances in Artificial Intelligence

30th Annual German Conference on AI, KI 2007
Osnabrück, Germany, September 10-13, 2007
Proceedings

 Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Joachim Hertzberg
University of Osnabrück
Institute of Computer Science
49069 Osnabrück, Germany
E-mail: hertzberg@informatik.uni-osnabrueck.de

Michael Beetz
Technische Universität München
Computer Science Department
Boltzmannstr. 3, 85748 Garching München, Germany
E-mail: beetzm@in.tum.de

Roman Englert
Deutsche Telekom Laboratories
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
E-mail: roman.englert@telekom.de

Library of Congress Control Number: 2007934041

CR Subject Classification (1998): I.2, I.2.6, F.1.1, I.5.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-74564-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-74564-8 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2007
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12115082 06/3180 5 4 3 2 1 0

Preface

The 30th Annual German Conference on Artificial Intelligence (KI-2007) took place in the University of Osnabrück, September 10–13, 2007. In this volume, you will find papers or abstracts of its six invited talks, 25 full papers, and 21 posters. The full papers were selected from 81 submissions, resulting in an acceptance rate of 32%.

As usual at a KI conference, an entire day was reserved for targeted workshops – ten of them this year – and two tutorials. They are not covered in this volume, but the conference Web site www.ki2007.uos.de will keep providing information and references to their contents. Some topic clusters are apparent in the overall conference program, which reflect recent trends in AI research, convolved with foci of work in Germany and Europe. Examples are learning and data mining, robotics and perception, knowledge representation and reasoning, planning and search – all of them including a healthy number of approaches dealing with uncertainty, contradiction, and incompleteness of knowledge. All in all, KI-2007 provided a cross section of modern AI research and application work.

KI-2007 also constituted a “small anniversary,” being the 30th exemplar of its kind. The invited talk by Wolfgang Bibel (accompanied by a paper in this volume) picked up on that occasion by recalling what the field of automated deduction was like 30 and more years ago – in general, and in Germany. He also paid homage to Gerd Veenker, who organized the first KI conference (which had a different name at the time) in 1975 and whose field of research was deduction. We were very happy that Wolfgang Bibel accepted our invitation to give this type of talk, as AI – developing swiftly as it does – is in permanent danger of forgetting its earlier days and the lessons that can be learned from them.

Our thanks and gratitude, first and foremost, go to the colleagues who accepted our invitations or submitted workshop proposals and papers and posters and all sorts of input: Imagine you organize a conference, and no one responds – we thank all of them that this was definitely not the case! Next, we thank all those who helped organize KI-2007 and who are listed on the next few pages. Part of the organization was, of course, handling the submissions and the reviewing. In that respect, the EasyChair conference system was of enormous help, and we would like to thank its main developer Andrei Voronkov for not only developing it, but also providing it for free to the scientific community. And finally, we thank the responsible and sponsoring institutions of this conference, also listed in the following pages: Without their support, a KI conference might have been possible, in principle, but it would have been much more stressful, much less successful and much less enjoyable!

July 2007

Joachim Hertzberg
Michael Beetz
Roman Englert

Table of Contents

Invited Talks

| | |
|--|----|
| The Role of AI in Shaping Smart Services and Smart Systems | 1 |
| <i>Sahin Albayrak</i> | |
| Early History and Perspectives of Automated Deduction | 2 |
| <i>Wolfgang Bibel</i> | |
| Cognitive Technical Systems—What Is the Role of Artificial Intelligence? | 19 |
| <i>Michael Beetz, Martin Buss, and Dirk Wollherr</i> | |
| Artificial Intelligence Is Engineering Intelligence – Why Should We Care About Natural Intelligence? | 43 |
| <i>Thomas Christaller</i> | |
| Applying Machine Learning Techniques for Detection of Malicious Code in Network Traffic | 44 |
| <i>Yuval Elovici, Asaf Shabtai, Robert Moskovitch, Gil Tahan, and Chanan Glezer</i> | |
| Location-Based Activity Recognition | 51 |
| <i>Dieter Fox</i> | |

Papers

| | |
|---|-----|
| Pinpointing in the Description Logic \mathcal{EL}^+ | 52 |
| <i>Franz Baader, Rafael Peñaloza, and Boontawe Sontisrivaraporn</i> | |
| Integrating Action Calculi and Description Logics | 68 |
| <i>Conrad Drescher and Michael Thielscher</i> | |
| Any-World Access to OWL from Prolog | 84 |
| <i>Tobias Matzner and Pascal Hitzler</i> | |
| Applying Logical Constraints to Ontology Matching | 99 |
| <i>Christian Meilicke and Heiner Stuckenschmidt</i> | |
| Resolving Inconsistencies in Probabilistic Knowledge Bases | 114 |
| <i>Marc Finthammer, Gabriele Kern-Isberner, and Manuela Ritterskamp</i> | |
| Extending Markov Logic to Model Probability Distributions in Relational Domains | 129 |
| <i>Dominik Jain, Bernhard Kirchlechner, and Michael Beetz</i> | |

| | |
|--|-----|
| A Multilingual Framework for Searching Definitions on Web Snippets . . . | 144 |
| <i>Alejandro Figueroa and Günter Neumann</i> | |
| A SPARQL Semantics Based on Datalog | 160 |
| <i>Simon Schenk</i> | |
| Negation in Spatial Reasoning | 175 |
| <i>Stefan Schleipen, Marco Ragni, and Thomas Fangmeier</i> | |
| Relational Neural Gas | 190 |
| <i>Barbara Hammer and Alexander Hasenfuss</i> | |
| A General Framework for Encoding and Evolving Neural Networks | 205 |
| <i>Yohannes Kassahun, Jan Hendrik Metzen, Jose de Gea, Mark Edgington, and Frank Kirchner</i> | |
| Making a Robot Learn to Play Soccer Using Reward and Punishment | 220 |
| <i>Heiko Müller, Martin Lauer, Roland Hafner, Sascha Lange, Artur Merke, and Martin Riedmiller</i> | |
| Perception and Developmental Learning of Affordances in Autonomous Robots | 235 |
| <i>Lucas Paletta, Gerald Fritz, Florian Kintzler, Jörg Irran, and Georg Dorffner</i> | |
| A Computational Model of Bistable Perception-Attention Dynamics with Long Range Correlations | 251 |
| <i>Norbert Fürstenau</i> | |
| On Constructing a Communicative Space in HRI | 264 |
| <i>Claudia Muhl, Yukie Nagai, and Gerhard Sagerer</i> | |
| Natural Language Descriptions of Human Behavior from Video Sequences | 279 |
| <i>Carles Fernández Tena, Pau Baiget, Xavier Roca, and Jordi González</i> | |
| Detecting Humans in 2D Thermal Images by Generating 3D Models | 293 |
| <i>Stefan Markov and Andreas Birk</i> | |
| Extent, Extremum, and Curvature: Qualitative Numeric Features for Efficient Shape Retrieval | 308 |
| <i>B. Gottfried, A. Schuldt, and O. Herzog</i> | |
| Extraction of Partially Occluded Elliptical Objects by Modified Randomized Hough Transform | 323 |
| <i>Kwangsoo Hahn, Youngjoon Han, and Hernsoo Hahn</i> | |
| Solving Decentralized Continuous Markov Decision Problems with Structured Reward | 337 |
| <i>Emmanuel Benazera</i> | |

| | |
|--|-----|
| Options in Readylog Reloaded – Generating Decision-Theoretic Plan Libraries in Golog | 352 |
| <i>Lutz Böhnstedt, Alexander Ferrein, and Gerhard Lakemeyer</i> | |
| On the Construction and Evaluation of Flexible Plan-Refinement Strategies | 367 |
| <i>Bernd Schattenberg, Julien Bidot, and Susanne Biundo</i> | |
| Learning How to Play HEX | 382 |
| <i>Kenneth Kahl, Stefan Edelkamp, and Lars Hildebrand</i> | |
| Stochastic Functional Annealing as Optimization Technique: Application to the Traveling Salesman Problem with Recurrent Networks | 397 |
| <i>Domingo López-Rodríguez, Enrique Mérida-Casermeyro, Gloria Galán-Marín, and Juan M. Ortiz-de-Lazcano-Lobato</i> | |
| A Stochastic Local Search Approach to Vertex Cover | 412 |
| <i>Silvia Richter, Malte Helmert, and Charles Gretton</i> | |
| Posters | |
| A Connectionist Architecture for Learning to Play a Simulated Brio Labyrinth Game | 427 |
| <i>Larbi Abdenebaoui, Elsa A. Kirchner, Yohannes Kassahun, and Frank Kirchner</i> | |
| Divergence Versus Convergence of Intelligent Systems: Contrasting Artificial Intelligence with Cognitive Psychology | 431 |
| <i>Stefan Artmann</i> | |
| Deep Inference for Automated Proof Tutoring? | 435 |
| <i>Christoph Benz Müller, Dominik Dietrich, Marvin Schiller, and Serge Autexier</i> | |
| Exploiting Past Experience – Case-Based Decision Support for Soccer Agents | 440 |
| <i>Ralf Berger and Gregor Lämmel</i> | |
| Externalizing the Multiple Sequence Alignment Problem with Affine Gap Costs | 444 |
| <i>Stefan Edelkamp and Peter Kissmann</i> | |
| Text Generation in the SmartWeb Multimodal Dialogue System | 448 |
| <i>Ralf Engel and Daniel Sonntag</i> | |

| | |
|--|-----|
| A Method to Optimize the Parameter Selection in Short Term Load Forecasting | 452 |
| <i>Humberto F. Ferro, Raul S. Wazlawick, Cláudio M. de Oliveira, and Rogério C. Bastos</i> | |
| Visual Robot Localization and Mapping Based on Attentional Landmarks | 456 |
| <i>Simone Frintrop</i> | |
| Bridging the Sense-Reasoning Gap Using DyKnow: A Knowledge Processing Middleware Framework | 460 |
| <i>Fredrik Heintz, Piotr Rudol, and Patrick Doherty</i> | |
| Emotion Based Control Architecture for Robotics Applications | 464 |
| <i>Jochen Hirth, Tim Braun, and Karsten Berns</i> | |
| Inductive Synthesis of Recursive Functional Programs | 468 |
| <i>Martin Hofmann, Andreas Hirschberger, Emanuel Kitzelmannn, and Ute Schmid</i> | |
| Training on the Job—Collecting Experience with Hierarchical Hybrid Automata | 473 |
| <i>Alexandra Kirsch and Michael Beetz</i> | |
| Selecting Users for Sharing Augmented Personal Memories | 477 |
| <i>Alexander Kröner, Nathalie Basselin, Michael Schneider, and Junichiro Mori</i> | |
| Semantic Reflection – Knowledge Based Design of Intelligent Simulation Environments | 481 |
| <i>Marc Erich Latoschik</i> | |
| Prolog-Based Real-Time Intelligent Control of the Hexor Mobile Robot | 485 |
| <i>Piotr Matyasik, Grzegorz J. Nalepa, and Piotr Zięcik</i> | |
| Improving the Detection of Unknown Computer Worms Activity Using Active Learning | 489 |
| <i>Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher, Roman Englert, and Yuval Elovici</i> | |
| The Behaviour-Based Control Architecture iB2C for Complex Robotic Systems | 494 |
| <i>Martin Proetzsch, Tobias Luksch, and Karsten Berns</i> | |
| Concept for Controlled Self-optimization in Online Learning Neuro-fuzzy Systems | 498 |
| <i>Nils Rosemann and Werner Brockmann</i> | |

| | |
|---|-----|
| LiSA: A Robot Assistant for Life Sciences | 502 |
| <i>Erik Schulenburg, Norbert Elkmann, Markus Fritzsche, Angelika Girstl, Stefan Stiene, and Christian Teutsch</i> | |
| Semantic Graph Visualisation for Mobile Semantic Web Interfaces | 506 |
| <i>Daniel Sonntag and Philipp Heim</i> | |
| A Qualitative Model for Visibility Relations | 510 |
| <i>Francesco Tarquini, Giorgio De Felice, Paolo Fogliaroni, and Eliseo Clementini</i> | |
| Author Index | 515 |

The Role of AI in Shaping Smart Services and Smart Systems

Sahin Albayrak

TU Berlin

sahin.albayrakdai-labor.de

Services and Systems must include a set of features to remain competent and future conform: intelligent behaviour, personalisation, adaptivity, scalability, manageability, ease of use and user friendliness, security, and self-healing capabilities. As a consequence, new architectural models are needed, which provide the users with access to a cognitive behaviour aspect of the system, and which may draw inspiration from the brain sciences. On the other hand, we have to use knowledge representation and semantic modeling, e.g., ontologies for representing our environment or basic properties of services and systems. This would naturally involve Agent Technology, AI, and Software Technology. So, approaches from many different disciplines have to work in integration.

Integrated frameworks handling such different aspects are called “Service-ware Frameworks”. They contain a scalable Service Architecture, which facilitates merging different selected features into a service, as well as a scalable so-called Service Engine with a Serviceware Infrastructure. For creating Smart Services and Smart Systems, we use engineering approaches that include innovative service description languages and tools. In this presentation, a framework with the properties and features just described will be presented. A sample application developed with this framework will also be presented: the “Smart Energy Assistant”.

Early History and Perspectives of Automated Deduction

Wolfgang Bibel

Darmstadt University of Technology
Also affiliated with the University of British Columbia
Bibel@gmx.net

Abstract. With this talk we want to pay tribute to the late Professor Gerd Veenker who deserves the historic credit of initiating the formation of the German AI community. We present a summary of his scientific contributions in the context of the early approaches to theorem proving and, against this background, we point out future perspectives of Automated Deduction.

Formal logic is still often looked upon as a kind of esoteric doctrine.

Evert W. Beth 1958

The fundamental scientific progress lies in the area of logic and the cognitive sciences.

Pierre Papon 2006

1 Introduction

Gerd Veenker is known in the German Artificial Intelligence (AI) community for his initiative and organisation of the first national AI meeting in Bonn in 1975 and the second one in Dortmund in the same year. This year we celebrate the thirtieth German AI conference and for this reason commemorate of him and of his work. Had he not died so prematurely we could as well have celebrated his seventieth birthday.

Veenker's scientific contributions are in the field of Automated Deduction (AD). In fact, he was the very first German scientist who contributed to this fruitful and still promising field. In the mid-sixties of the last century with his theoretical work and his working systems he was at the forefront of AD internationally. For instance, his system NEU of 1966 realized what only a decade later was reinvented and called UR-resolution (for Unit Resulting). Unfortunately, he was totally isolated in those days when Informatics did not yet exist in Germany, let alone an "esoteric doctrine" like computational logic [Bet58, p.50]. So his contributions have stayed totally unnoticed.

As a courageous pioneer he deserves to be commemorated. We therefore summarize in Section 4 of this paper some of his early contributions. Because these cannot be appreciated without some knowledge about the state of the art in

AD at those days, we give in Section 2 an account of the first complete theorem proving procedures in first-order logic by Prawitz and Gilmore, with a brief mention of other early deductive systems by Dunham et al., Newell et al., and Davis as well as of McCarthy's seminal contribution of LISP.

Section 3 then describes the advances in AD made in the early sixties. These include unification, Skolem functions, Herbrand universe, clause form, unit resolution, and especially Robinson's resolution. We also point out the circumstances under which these achievements could be obtained and draw a lesson from these observations. This is contrasted with Veenker's situation and his work is analysed in comparison with those advances in the subsequent section as already mentioned. Again we draw a lesson from this comparison for the discipline of Informatics of our days in Germany (or in Europe for that matter).

The paper concludes with some perspectives for AD in the future. Although AD is extremely successful already and offers even more potential, a substantial advance of our systems' performance would require a much better support especially in terms of the quality of education and of the research environment. This is because the formidable challenge of an integration of the many different features in one single system as well as solving important remaining problems will hardly be achievable in the current splintered manner. Even further, there are deep remaining issues to be solved concerning the nature of the underlying logic. Under these considerations we are led to the proposal of the foundation of some European center of excellence for semantics, logic and computation in order to come a step closer to Leibniz' dream of a reasoning machine in the not so distant future.

2 How Automated Deduction Started

In 1957 Aridus Wedberg taught a first year logic course at the University of Stockholm. On one occasion he mentioned to the class the possibility of proving mathematical theorems in first-order logic on a machine. This remark raised the interest of one of the students in the class, namely Dag Prawitz, who decided to realize this idea in practice [SW83, p.200].

First he developed a general procedure for the predicate calculus which we illustrate with the valid formula $\forall x Px \rightarrow \exists y Py$, shortly F . In order to prove F , we assume it were false and infer a contradiction. That is, we start by assigning the truth value f to it and let the pair (F, f) be the first in a list of subformulas along with truth values. Given the semantics of implication, for F assumed to be false this means that $\forall x Px$ must be true, or t , and $\exists y Py$ must be false, yielding the next two pairs in the list. Taking the first list item not considered so far, this means that for any constant c in the universe under consideration Pc must be true. In order to mechanize this step for the general case, assume that all constants are enumerated as c_1, c_2, \dots . Since no constant of this enumeration was used before in our example, we simply take its first one and, hence, add the pair (Pc_1, t) at the end of our list. The final item yet to be considered in the list, $(\exists y Py, f)$, implies that for any constant it must be false. The procedure in

general selects one already used before in this case, ie. here c_1 , leading to the final list element (Pc_1, f) . Now the list contains two occurrences of the literal Pc_1 with opposite truth values, indicating the expected inconsistency so that the proof now is complete in this case.

For those readers with some familiarity in AD it is clear that this procedure is generating a so-called *tableau* for the given formula except that today the truth values are coded by adding a negation sign in front of the subformula instead of falsehood. In generating the tableau Prawitz' procedure follows precise rules for each of the possible cases, characterized by the outermost logical symbol determining the form of the formula as well as by the associated truth value. Since, for instance, false conjunctions could lead to alternative subcases such a tableau in general consists of a tree with each of its branches being a list like the one in our example, especially in terms of closing the branch by some contradictory pair of literals. Also the selection of constants is a little more complicated than illustrated by our simple example.

Prawitz coded this procedure in a programming language which he designed himself for this special task and wrote a report in Swedish. His father, Håkan Prawitz, hand-translated the program into machine code in 1957. The result was worked over and tested by Neri Voghera, a software expert, for a number of examples in 1958. Thereby he used a computer named Facit EDB, built in Sweden. It featured a core memory of 2048 40-bits machine words and a drum with a capacity of 8192 words. In other words, the first experiments with a general theorem prover for first-order logic were performed in Stockholm in 1958. In 1959 the work was outlined in the discussion of the session on theorem proving at the First International Conference on Information Processing (IFIP) in Paris, the discussion being contained in the proceedings. In 1960 the full paper describing the work appeared in the Journal of the ACM [PPV60].

This short description of the very first work in first-order AD needs to be complemented by a number of comments. First, Prawitz' procedure did not fall from heaven but rooted in well-known work done in Mathematical Logic. Second, there were several other efforts undertaken in that period of time. Third, progress in the early years of AD depended a lot on the programming infrastructure available at the respective location. Let us discuss each of these three important issues in turn.

To begin with the first point, this is not the place to give an outline of the history of logic. There are excellent sources for this purpose such as [KK84]. Also the article [Dav83] summarizes this history with an emphasis on AD and the author's work in it. We want to point out the following highlights in this remarkable history.

Leibniz was the visionary for an instrument to increase the powers of reasoning [Dav83, pp.2ff,14]. Frege's Begriffsschrift [Fre79], with explicit reference to this vision, laid the grounds for all formal languages, logical or programming ones, as well as for logical calculi. Around the 1920's and early 30's the work of Skolem, Herbrand, Gödel, Gentzen, and Jaśkowski as well as the book by Hilbert and Ackermann [HA28] clarified the most important logical concepts and issues

such as completeness, decidability, Skolem functions, Herbrand's theorem, Herbrand universe which for historical correctness should actually be named "Skolem universe", Gentzen calculi, cut elimination, and so forth.

In the mid 1950's a proof procedure by W.V. Quine [vOQ55a] as well as four new and simplified completeness proofs for first-order logic by E.W. Beth [Bet55], K.J.J. Hintikka [Hin55], S. Kanger [Kan57], and K. Schütte [Sch56] were published independently which had an immediate impact on the way early theorem provers were designed. For instance, Prawitz followed closely Beth's formalism in developing his procedure discussed above. While Beth's and Hintikka's systems used proof by contradiction, Kanger and Schütte pursued an affirmative approach. The difference is completely irrelevant from a logical or deductive point of view. But Prawitz and others introduced the contradictory approach which led later researchers to follow this habit.

Let us now come to the second point concerning other early AD efforts. Here we may distinguish four different lines of research, namely first-order theorem proving, propositional methods, heuristic approaches, and decision procedures, which are discussed again in turn.

In 1958 Paul Gilmore, teaching at Penn State (Pennsylvania State University, State College PA), a place made famous by the great logician (and amateur ornithologist) Haskell B. Curry, read an advertisement in the New York Times for a mathematician interested in assisting in a project for proving theorems in Euclidean Geometry. He applied and eventually joined the Mathematics Department of IBM Research at the Lamb Estate in Croton-on-Hudson NY in July 1958 where he worked with Herbert Gelernter (of whom more below). Gilmore had a solid background in Mathematical Logic from a course in Mathematical Logic of S.W.P. Steen at Cambridge University, his studies in Amsterdam with E.W. Beth and A. Heyting, and his earlier collaboration as a postdoc with Abraham ("Abbie") Robinson at the University of Toronto in Canada. Since he had had no experience with or knowledge of electronic computers at the time, Gilmore decided to learn by implementing in assembly language on an IBM 704 Beth's method of semantic tableaux for first-order logic, although eventually the implemented method was "closer to the work of Hintikka".

The resulting program, described in [Gil60], took as input any negated first-order formula in prenex form with its matrix in disjunctive normal form (called standard form in the paper). In this relatively unimportant aspect it differed from Prawitz' procedure (applicable to arbitrary formulas) but otherwise used the same crude search technique for appropriate substitutions. Gilmore thought that his "work is the first working program for quantification theory". He learned of Prawitz' working program only, when he met him at the Paris conference in 1959, and acknowledged the fact in a footnote in [Gil60].

Due to the focus of the present paper we will treat the other three mentioned lines of AD research only in passing. Abbie Robinson had already pointed out in an influential talk at the important five weeks Summer Institute for Symbolic Logic at Cornell University in 1957¹ [Ref03] which was attended also by some

¹ In [Dav83, p.16] the year was stated incorrectly as 1954.

twenty people working in the computer industry including G.W. Collins, B. Dunham, R. Fridshal, H. Gelernter, J.H. North, who presented talks there and are mentioned elsewhere in the present paper, that through Herbrand's theorem first-order theorem proving could be reduced to the propositional level [Rob57]. So propositional theorem proving became interesting not only because of its relevance for the logic of computer hardware but also in our more general context. The first propositional method of a formal kind was used in [DFS60], again programmed for an IBM 704 and also presented at the Paris conference. It used rules such as case splitting and pure literal reduction to rewrite the given expression until the truth value t or f was obtained.

In a sense this work may be seen as a reaction to earlier work by Newell, Shaw and Simon [NSS56] who took a heuristic rather than systematic approach to propositional theorem proving, relying on axioms, forward and backward implicational chaining and modus ponens (cf. [Cor96] for an account of such heuristic approaches). Their program was run on the JOHNNIAC computer from Rand Corp. Russell. Similarly Herb Gelernter relied on a heuristic approach in his realization of a rather successful geometry-theorem proving machine [Gel59] which became operative in 1959. Geometry with its long axiomatic tradition since Euclid's *Elements* was a natural mathematical subject to start with. Through Gilmore's paper [Gil70] Gelernter's geometry theorem prover came to influence modern theorem proving by its use of models to test the consistency of an hypothesis.

The final line of early AD research mentioned above consisted in the implementation of known decision procedures. Martin Davis implemented such a procedure for Presburger arithmetic already in 1954, using the JOHNNIAC of the Princeton Institute for Advanced Study [Dav57]. In effect this then was the very first operative system in AD, although a very restricted one in scope as in achievement. George Collins from the IBM Labs implemented on an IBM 704 parts of Tarski's decision procedure for elementary algebra in order to deal with a variety of problems that could be expressed in that language, a work presented at the same occasion as Davis' [Fef03].

The third and final point in this review of the beginning years in AD refers to the available infrastructure. It constrained the possible success much more than anything else. Prawitz had the luck to find the support of his father and of Voghera who did the extremely time-consuming job of programming and testing. Similarly a well-equipped environment like IBM Research, eg. for Gilmore and Gelernter, turned out to be very helpful.

It was John McCarthy who, on the basis of this experience, put the design and implementation of LISP on the top of his priorities since the availability of such a high-level language could reduce the amount of implementational work dramatically. For instance in [McC59] McCarthy says: "[The Wang algorithm for propositional logic] took about two hours to write the program and it ran on the fourth try." Unfortunately, even more than a decade later computing centers such as those in Germany typically had no implementation of LISP yet available

so that, for a few exceptions, the waste of time in implementational efforts in AI continued for many more years all over the world.

3 Important Historical Advances in AD

In the previous section we have described the very first attempts of automating deduction in first-order logic. Needless to mention that the theorems, which could be proved with those systems, were only rather trivial ones. This initiated half a century of research into improvements of these first procedures. In this section we want to describe some of the main early contributions in this vein.

An immediately obvious drawback of the first procedures was their treatment of substituting constants in a stupid systematic manner. Prawitz was the first in proposing a unificational method instead [Pra60] which used metavariables and substituted constants *by need* rather than according to some fixed sequence. The unification was computed by way of a system of resulting equations to be solved under certain restrictions. These restrictions derived from the well-known variable conditions in Gentzen-type systems. It is worth pointing out that later unification algorithms used a rather similar way of computation. In other words Prawitz deserves the credit for having introduced unification into proof procedures.

None of the procedures mentioned so far allowed function symbols other than constants as it was known from standard logic textbooks that these could be replaced in a certain way by predicates. The first paper [DP60] introducing Skolem functions, hence function symbols, and the Herbrand universe was by Martin Davis, a former student of Alonzo Church, and by Hilary Putnam, a mathematician-turned philosopher. It also proposed the clause form arrangement of the initial data to be refuted, i.e. proved by contradiction, which from there on has become a widely used standard. Unfortunately, this standard along with the weakness in Gilmore's procedure beared the ineradicable myth that the use of this standard has computational advantages over an analogue affirmative "clause" form, i.e. a representation of the formula in disjunctive normal form to be proved rather than refuted, although the difference is of course totally irrelevant [Bib87]. A further contribution was unit resolution which the authors called rule for the elimination of one-literal clauses.² In its treatment of terms it remained ignorant of Prawitz' unificational ideas while in the propositional part it used the independently discovered rules from [DFS60] already described above, all without any implementation though.

In 1960 a postdoc, J. Alan Robinson, at the University of Pittsburgh sent applications to several institutions including the Applied Mathematics Division of the Argonne National Laboratory at Chicago IL and received an offer for a summer research position from it. He eventually decided to rather accept a tenure track teaching offer for logic and philosophy of science from Rice University, but go to Argonne for a summer research position with the task assignment

² In [Cor96, Sect.3] the author erroneously attributes the introduction of unit resolution to [WCR64] as did others before him.

by the Division's director, William Miller, to implement on Argonne's IBM 704 the method of the Davis-Putnam paper just mentioned. From a four years employment at duPont he already had a solid experience in assembly language programming. So when he arrived at Argonne in May 1961 the programming was already done, yet in lack of a computer untested. Getting it run, reprogrammed in Fortran, and debugged took considerable efforts which were supported by George Robinson, the head of the division's Programming Development Section. Alan wrote an Argonne Report [Rob61] which in a polished version appeared eventually in 1963 in the Journal of the ACM [Rob63].

In the summer months of 1962 and 1963 Alan Robinson returned to Argonne where George Robinson had already become so excited from the earlier experiments that he started the transition to become a theorem proving researcher. Miller assigned one of his mathematicians, Larry Wos, to join the team consisting of the two Robinsons who first gave him a crash course in logic using Quine's *Methods of Logic*. During these two summer projects, Alan Robinson rediscovered unification and the resolution rule (in 1962) and introduced these two terms into the literature for the first time with the seminal paper [Rob65] (although the publication was delayed by some rumored referee until January 1965).

As to unification we already pointed out that Prawitz' first paper did contain the basic idea behind unification (as did a work by N.A. Shanin – see [SW83, p.30]) and Robinson was strongly influenced by it. He himself says: "I was absolutely inspired by Prawitz"³ However, already Herbrand's paper [Her30] contained a much more elegant version of it, which basically is the one Robinson published in [Rob65], expressed in recursive definitional rather than algorithmic terms though. Although Prawitz cited this Herbrand paper, he was not aware of this part of its contents (nor was Robinson). As to the resolution rule it was first discovered in [Bla37] already in 1937, coincidentally like Argonne also in Chicago, then rediscovered in [vOQ55b] as *consensus rule* and proposed for use in (propositional) theorem proving in [DN63], presented at Harvard University already in February 1962.

The beauty of Robinson's paper derives from his ability to rediscover these two powerful techniques and merge them with the solid platform for theorem proving which had been achieved by that time (including the purity and subsumption principles). Further, he did so in a mathematically clean and perfect way. This latter point is especially remarkable since Robinson by education was a philosopher with a Master's thesis on *Theories of Meaning Implicit in the British Empiricists Locke, Berkeley and Hume* and a PhD thesis on *Causality, Probability and Testimony*. He attributes the stimulation for his transformation towards a mathematically-oriented scientist especially to his teacher Arthur Pap,

³ Personal communication (e-mail message of 7 March 2007). – Prawitz' influence can also be seen in the worked example of Davis paper [Dav63, Section 6] which of course references Prawitz' work. This paper does however give not yet any specific hint to a unification-like method à la Robinson beyond Prawitz' equational system as suggested in [Dav83, p.18].

like Putnam a mathematician-turned philosopher. Pap also advised him for his PhD studies to go to Princeton where Alonzo Church educated a whole generation of excellent logicians (including John McCarthy, Marvin Minsky, and Dana Scott), although it was actually Hilary Putnam who acted as supervisor to his dissertation.

Once resolution was available the group at Argonne under the direction of Larry Wos and George Robinson set down to improve its performance by reducing the search space of generated resolvents. Most importantly they introduced factoring, the unit preference, the set of support strategy and implemented resolution with these additional features on a Control Data 3600 [WCR64]. At this point Argonne had become the undisputed world champion in theorem proving. Later, in response to a suggestion by Alan Robinson, they started to concentrate on dealing with equality in a special way, introducing demodulation and paramodulation. Generally, the publication of J.A. Robinson's paper spawned a flood of publications in theorem proving, ninety alone in the years 1967–1970, a period which is well covered by the article [WH83].

This tremendous influence extends up to this day. For instance, the winner of the recent CASC competitions in theorem proving was the resolution-based system Vampire by Andrei Voronkov. This dominance is however not undisputed, a topic which we further pursue in the final section of this paper.

So what is the lesson to be drawn for fertilizing the grounds for future discoveries in our or other fields? The obvious first conclusion is that a top education is the most important prerequisite for excellence. Prawitz, Gilmore, Davis, Robinson, McCarthy and many others are proof to this rule as we described. Of similar importance is the research environment comprising a wise leader like William Miller at Argonne, the right combination of people like Prawitz (logician) and Voghera (software engineer) or Robinson (logician), George Robinson (software engineer) and Wos (Mathematician), and adequate facilities. Argonne had won this competition because it featured both prerequisites in the best possible combination. There was nothing like Argonne in Europe in those days. For instance the Gesellschaft für Mathematik und Datenverarbeitung (GMD) was founded in Bonn not before 1968.

4 Gerd Veenker (1936–1996)

Gerd Veenker was born 9.12.1936 in Lüneburg. His father was a tailor which is worth noting because not only Gerd but also his brother Wolfgang later became university professors. After his school education in Lüneburg until 1957 he studied Mathematics and Physics in Hamburg, München and Tübingen.

Around 1960 he became interested in computers. He and his friend Frieder Schwenkel developed a particular interest in non-numeric computation such as game playing and theorem proving. They studied for instance the respective parts in the proceedings of the first IFIP conference in Paris 1959 which has already been mentioned several times in the preceding two sections.

No guidance by any professor in Tübingen could be expected to further this interest. Possibly not even an appropriate logic course was offered which could have introduced him into the underlying subject. However, Karl Zeller (28.12.1924–20.7.2006), a Mathematics professor with a speciality in limit theory and with experiences from several visits at US universities, in 1960 got a chair (Lehrstuhl Mathematik der Hochleistungsrechenanlagen, ie. mathematics of high-performance computers) which was at the same time responsible for the university’s computing center. It featured a Siemens 2002 also installed in 1960. Professor Zeller had a widely open mind and an unusually liberal attitude towards his students in terms of their subjects of interest. So when Veenker decided on his own to concentrate on theorem proving in his Diplomarbeit (Master’s thesis) and dissertation he would let him go in this direction and formally play the role of the supervising professor. He was supportive in that he allowed his students to make suggestions for invited colloquium talks (eg. Hermes from Freiburg) as well as for the topic and the literature of seminars officially run under his name. Additionally helpful was the friendly cooperative atmosphere among the members of the small group of students which as “Hiwis” (research assistants) gathered around the computing center.

In 1963 Veenker completed his Diplomarbeit (master’s thesis) entitled *Ein Entscheidungsverfahren für den Aussagenkalkül der Formalen Logik und seine Realisation in der Rechenmaschine* (A decision procedure for the propositional calculus of formal logic and its realisation on the computer). The list of its references demonstrates that in the meantime he had read most of the theorem proving literature available by 1962, in particular the papers discussed in the preceding two sections of the present paper. He gives a concise description of the related procedures of Hao Wang [Wan60b], Paul Gilmore [Gil60], Dunham et al. [DFS60], and Davis and Putnam [DP60]. In effect his procedure follows closely the one by Prawitz [PPV60], restricted to the ground level and allowing for the five logical operators $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$. It is programmed in the symbolic low-level programming language PROSA. For the first-order level he gives an outline of an envisaged but not yet implemented procedure.

For a master student this work is truly remarkable if one takes the lack of guidance and logical education as well as the limited computational infrastructure and resources into account. For instance, he used a number of tricks to fit formulas with up to 50 logical operators into the machine’s core memory consisting of 2048 machine words. In contrast to McCarthy’s two hours mentioned in Section 2 it took Veenker probably hundreds of hours to get the program to the point of success. He proudly states that for one example his program takes 84 seconds, for which Gilmore’s system could not find a proof after 21 minutes. The technical reason for this advantage is Gilmore’s costly transformation to disjunctive normal form which Prawitz had already avoided. The work was published shortly after completion [Vee63].

He also published a paper on a program for chess endgames which allow mate in two or three moves. But his focus remained on theorem proving. At the end of 1966 he had a dissertation ready with which he passed the rigorosum at the

beginning of 1967. Again, although working in full isolation, he was fully up-to-date with his references to the papers influencing the field during that time which in those days in lack of anything like a world-wide web required enormous efforts indeed. Apparently he learned of Robinson's 1965 resolution paper after much of his thesis and especially of his program was already completed. This can for instance be seen from his way of treating unification (in his algorithm GLS which is short for German >GLEichSetzung<) which follows the equational style of Prawitz' first paper but now extended to cover general terms since he used Skolem functions. So he may well have – once again – reinvented unification (called Verschmelzung) for general terms. Generally, he follows Davis' representational style [Dav63] and uses Davis' term "linked" in form of the German Verkettung.

On this basis he enumerates all paths through the matrix given by the set of clauses. However, once the procedure has located a connection it eliminates all paths through this connection in one step, focusing next on the two paths obtained from the current one by replacing each of the connected literals by another literal of the same clause. One of these two resulting paths is handled next while the other one is put on a stack for later treatment. So in summary the procedure is advanced in terms of the enumeration of the paths but in comparison with the later connection procedures [Bib87] does not yet restrict the search for a subsequent connection to those with literals in the previously connected clause which in resolution is known as the linearity restriction. This is unfortunate because he already mentions this possibility but only as a preference strategy; apparently he had not seen that this preference can be done without restriction of generality. He does use also the unit preference and the set of support strategy.

In addition to the complete and sound proof procedure just described, Veenker gives an incomplete procedure, called NEU (for new), which today we know as the unit-resulting (UR) strategy combined with unit resolution. The literature thus incorrectly attributes the discovery of this strategy to the authors of [MOW76], where it was introduced under this name, while Veenker invented it already ten years earlier. In lack of a better machine he programmed both procedures still for the Siemens 2002 which in comparison with the CD 3600 at Argonne was at least two orders of magnitude inferior. Taking this into account his running times were well competitive with the state of the art in 1966, a remarkable achievement in view of the lack of what we pointed out at the end of the previous section as the prerequisite for excellence, namely a top education in the subject and a stimulating research environment.

His PhD work was published in [Vee67] but totally ignored by the community. One reason of course was the publication being in German. Another was his total isolation as the only theorem proving specialist in the entire German-speaking area, if not in all of Europe at that time with the sole exception of Prawitz in Stockholm and of Bernard Meltzer at Edinburgh. In addition Veenker was a rather modest and reserved person. When the present author met him in 1969, discussing theorem proving issues, he apparently failed to point to his published work, then as well as in all encounters in later years, which I therefore read

carefully for the very first time not before the preparation of the present paper. This meeting in 1969 gathered Informatics researchers with DFG-funded projects at the Chiemsee and most likely was the very first event where Informatics-based German AI researchers from different institutions got together for presentations and discussions. According to my recollections he did not give a talk (nor did I).

Presumably he had just applied to the DFG for project funds to attack his next goal in theorem proving which was a special treatment of equality in his (incomplete) procedure NEU. This was achieved in the Diplomarbeit of Geerd-Rüdiger Hoffmann and published 1971 in [\[HV71\]](#) in English. It uses a kind of theory unification for equality. In the same year the author presented his first theorem proving paper at the GI Jahreskonferenz in München which in the discussion was heavily attacked by Mr. Hoffmann who pointed out that my Gentzen-type approach had already been shown not to be workable and thus waste of efforts. One might infer from this opinion of his student that also Veenker at that time had given up hope to pursue the line initially taken by him and rather opt for resolution as the winning technique. Perhaps it is for this reason that he also gave no talk about his work at the Oberwolfach meeting on automated theorem proving in 1976 which he attended.

Without having undergone the procedure of Habilitation Veenker received an (associate) professorship for Informatics and Applied Mathematics at the University of Bonn in 1972 where for 34 years he represented AI in his teaching. He never was promoted to a full professorship. In this position he took the initiative for the first official German AI meeting mentioned in the Introduction. His PhD students are Rainer Fröning, Joachim Hertzberg, Eberhard Klein, Knut Möller, Peter Schmidt, Volker Steinhage, and Erich Vorwerk, as far as I could find out. Two of these (Hertzberg and Möller) are now professors. As a professor he was popular with students because of his friendly and warm-hearted personality and hence he supervised a great number of Diplomarbeiten (master's theses). Some of these laid the foundation of academic careers of prolific scientists like Gerd Brewka, Dieter Fox, and also Sebastian Thrun whose autonomous vehicle Stanley in 2005 spectacularly won the DARPA Grand Challenge. Unfortunately, since 1976, Veenker suffered from very serious health problems which apparently kept him from staying scientifically as productive as during the first decade of his career. He died 23.6.1996 at the age of 59 shortly after the deaths of his wife and of his brother in the same year.

Are there any lessons to be learnt from the history of a man who was the first German scientist in the area of AD? In any case it raises a number of questions. One such question was already asked in 1969 by the logician Richard Büchi: "Why did the German logicians not engage in establishing the new field of Informatics?" He posed this question, which is to be seen within the context of Germany during Hilbert's time being the world-leader in logic, at the occasion of the inauguration of the Informatics buildings at the Technical University of München (TUM) to the internationally known German logician Kurt Schütte, a student of Hilbert. Büchi, then at Penn State, was invited to this occasion for a presentation. Schütte could not provide any reasonable answer to his question.

Later, Friedrich L. Bauer, one of the founders of Informatics in Germany joined Büchi and Schütte and, although he had not heard their prior discussion, shocked the two (as well as me) by harshly stating from nowhere: “Logic by now has no more than a peripheral significance for Informatics.”

In fact, Bauer’s statement provides an explanation for Veenker’s unfortunate situation. Establishing a new field like Informatics against the extremely rigid structures of the German academic world was not something to be achieved by decent and modest persons like Schütte who was exclusively devoted to his subject. It required clever and versatile power figures like Bauer who had all the required tricks at their disposal, even though they may have lacked the necessary education in the germane subjects. So Bauer was in fact right insofar as power influence was concerned.

Also it must be said that the German logics community did remain seated in its ivory tower. Besides organizing the International Logic Colloquium (held in Hannover) one of their major concerns in the sixties was the revision of the constitution of the German logic association (Deutsche Vereinigung für Mathematische Logik und Grundlagen der Wissenschaften, or DVMLG) which was bitterly debated for years. Bernays (Basel) was already too old to play a leading role, Specker (Zürich) as a Swiss kept himself at a distance, Büchi left to the US, Schütte did not even dare to respond to Bauer’s statement, and so forth. Academically they kept themselves in high regard as an elite which allegedly had good reason to look down to the academically and logically uninteresting computational problems of Informatics. Those who transformed from Logic to Informatics, like the author, became sort of banned, in any case kept in low regard. So people like Veenker, and to some extent also Prawitz who perhaps for those reasons later retreated back into logic and philosophy, academically found themselves sitting between the chairs, in stark contrast to the analog situation in the US where Computer Science was open-minded enough to appreciate topics like theorem proving and respected logicians like Davis did not feel like making their fingers dirty by pondering over the computational issues of proof procedures.

Since these historic frictions are still virulent in various ways, the lesson then is that attempts should be made to become consciously aware of, and overcome, them. In particular this means that Informatics should acknowledge computational logic as one of their fundamental and promising subareas, reflected also in the official characterizations of the field where it is rarely mentioned at all.

5 Perspectives for AD

The author has outlined his credo for the field of AD only recently in the article [\[Bib06\]](#). We will therefore not repeat these arguments here again except for a few additions especially with respect to the issues discussed in the previous sections.

Recall that at the end of Section 3 we reported of the success of resolution in the mid-sixties of the last century. But it turned out that resolution was no

panacea either in terms of efficiency of proof search. A lot of tricks have to be added and sometimes it is not clear why they work at all. The deeper reason for these problems lies in the fact that up to now the resolution rule has not been thoroughly understood. The present author thought he had achieved such a thorough understanding in his paper [BE97]. But Jörg Siekmann and Graham Wrightson pointed out that the result contradicts an example from [Eis91] so that there must still be some mistake in the obtained result which has not yet been discovered and corrected by anyone. Imagine that forty, in fact nearly 70 years after the discovery of resolution we are still struggling to understand it fully.

The situation is quite different for Gentzen-type theorem proving of the kind which was initiated by Prawitz as discussed in Section 2. With all the work which followed Andrews' matings method [And81] and Bibel's connection method [Bib83] which eliminated the original disadvantages pointed out in the literature discussed in Section 3 we know exactly what kind of improvements could still be made for a better performance of the systems. The difficulty lies in the enormous complexity of the task. In consequence many of the improvements which were worked out theoretically are not yet incorporated in one single system. The CASC-winning system SETHEO [LSBB92] featured many of them but by far not all. One particularly important example is the cut rule which has never been taken care of in any running system except for its very limited consideration in SETHEO. The author has stated a conjecture in [Bib06] which would open a way for its treatment. Another example concerns a refined treatment of variables as already discussed in [Bib87] which has just been worked out in more details in [AW07]. Like these two there are many more issues (heuristic guidance at the meta-level in special theories, learning of strategies, integration of models, etc.) let alone visions like Robinson's "science of proofs-as-explanations" [Rob00], which all are still waiting for being integrated into one single system along with all features scattered in various existing systems. Altogether this amounts to a formidable task.

One should even go a step further in broadening the perspective. Our field today features a great variety of different logics and logical calculi. In [Bib06, Sect.5] I already pointed out the importance of embedding the static logical space, under discussion so far in this paper, into the course of time in an appropriate way. We believe that transition logic [Bib04] achieves this aim in a more natural and effective way by focussing on local transitions rather than on a global transition from one world to another as in modal logics. But even within the static logical space something might be going wrong which could be rooted deeply in some historical decision made long ago. I mention the book [Brü96] which tries a restart of Aristotle's syllogisms in a modern and precise setting. It is just a very first little step in comparison to what modern logics offer. But it could be one of a more constructive nature, which, if followed by further ones, might possibly lead to a logic with better computational features than those which we know today. In fact it might be a good idea to start yet one step further back and abstract with modern AI technologies the logic underlying natural languages from large text corpora.

Achieving all these tremendously complex tasks according to the lesson from the end of Section 3 would require an excellent education of brilliant minds along with a research environment which I cannot spot anywhere in the world. Therefore I put forth the suggestion to *found sort of a Max-Planck Institute on the European level with such a broad basic research mission*. It would have to combine research excellence in a variety of related fields including semantics of natural language, logic and philosophy of logic, psychology of inferencing and proofs, cognitive science, knowledge representation and reasoning, and above all computation.

I want to conclude by pointing to the relevance of deduction in all kinds applications in virtually every area, independent of any of these future advances. This is because of the fundamental importance of reasoning in all human activities [Bib03]. Especially through the semantic web and through knowledge systems the importance of deduction will surely grow tremendously [Bib07]. So I completely share physicist Pierre Papon's conviction as expressed in his statement cited at the beginning of this paper [Pap06, p.10].

Acknowledgments. The text owes a lot to a number of people from whom the author got first-hand information about those early days. This includes a touching text about his personal developments by Alan Robinson and a delineation of his encounter with theorem proving by Paul Gilmore as well as helpful discussions with both. My picture about Gerd Veenker's early career derives from extensive material and support provided by Joachim Hertzberg, who was one of the initiators for this paper, and from lively discussions or exchange of letters with Margarete Zeller, Tübingen, Wilhelm Niethammer, Karlsruhe, Manfred Reimer, Dortmund, and Frieder and Trude Schwenkel, Hamburg/Winsen. Further information was received from Thomas Christaller, Martin Davis, Hans Langmaack and Sebastian Thrun. I am grateful to all of them. For any errors I take of course full responsibility.

References

- [And81] Andrews, P.B.: Theorem proving via general matings. *Journal of the ACM* 28, 193–214 (1981)
- [AW07] Antonsen, R., Waaler, A.: Liberalized variable splitting. *J. Automated Reasoning* (2007)
- [BE97] Bibel, W., Eder, E.: Decomposition of tautologies into regular formulas and strong completeness of connection-graph resolution. *Journal of the ACM* 44(2), 320–344 (1997)
- [Bet55] Beth, E.W.: Semantic entailment and formal derivability. *Mededlingen der Koninklijke Nederlandse Akademie van Wetenschappen* 18(13), 309–342 (1955)
- [Bet58] Beth, E.W.: On machines which prove theorems. *Simon Stevin Wis- en Naturkundig Tijdschrift* 32, 49–60, Reprinted in [SW83, 76–90] (1958)
- [Bib83] Bibel, W.: Matings in matrices. *Comm. ACM* 26, 844–852 (1983)
- [Bib87] Bibel, W.: *Automated Theorem Proving*, 2nd edn. Vieweg Verlag, Braunschweig (1987)

- [Bib03] Bibel, W.: Lehren vom Leben – Essays über Mensch und Gesellschaft. In: Sozialwissenschaft, Deutscher Universitäts-Verlag, Wiesbaden (2003)
- [Bib04] Bibel, W.: Transition logic revisited, 2004 (Submitted)
- [Bib06] Bibel, W.: Research perspectives for logic and deduction. In: Stock, O., Schaerf, M. (eds.) Reasoning, Action and Interaction in AI Theories and Systems. LNCS (LNAI), vol. 4155, pp. 25–43. Springer, Heidelberg (2006)
- [Bib07] Bibel, W.: Wissenssysteme und Komplexitätsbewältigung. In: Leiber, T. (ed.) Denken und Handeln in einer komplexen Welt – Festschrift zum 60. Geburtstag von Professor Klaus Mainzer, Hirzel Verlag, Stuttgart (2007)
- [Bla37] Blake, A.: Canonical Expressions in Boolean Algebra. PhD thesis, University of Chicago, Illinois (1937)
- [Brü96] Brüning, W.: Grundlagen der Strengen Logik. Königshausen und Neumann, Würzburg (1996)
- [Cor96] Cordeschi, R.: The role of heuristics in automated theorem proving – J.A. Robinson’s resolution principle. *Mathware & Soft Computing* 3, 281–293 (1996)
- [Dav57] Davis, M.: A computer program for Presburger’s algorithm. In: Summaries of talks presented at the Summer Institute for Symbolic Logic, Princeton NJ, pp. 215–233, Institute for Defense Analysis (1957), Also contained in [SW83, 41–48]
- [Dav63] Davis, M.: Eliminating the irrelevant from mechanical proofs. In: Proc. Symposium for Applied Mathematics XV, Providence, RI, pp. 15–30 (1963), Also contained in [SW83, 315–330]
- [Dav83] Davis, M.: The Prehistory and Early History of Automated Deduction. In: Siekmann, J., Wrightson, G. (eds.) Automation of Reasoning 1 – Classical Papers on Computational Logic 1957–1966, pp. 1–28. Springer, Berlin (1983)
- [DFS60] Dunham, B., Fridshal, R., Sward, G.L.: A non-heuristic program for proving elementary logical theorems. In: First International Conference on Information Processing, Paris, pp. 282–285. Unesco House (1960), Also contained in [SW83, 93–98]
- [DN63] Dunham, B., North, J.H.: Theorem testing by computer. In: Proc. Sympos, Brooklyn NY, pp. 173–177. Polytechnic Press (1963) Also contained in [SW83, 271–275]
- [DP60] Davis, M., Putnam, H.: A computing procedure for quantification theory. *Journal of ACM* 7, 201–215 (1960), Also contained in [SW83, 125–139]
- [Eis91] Eisinger, N.: Completeness, Confluence, and Related Properties of Clause Graph Resolution. Pitman, London (1991)
- [Fef03] Feferman, S.: Alfred tarski and a watershed meeting in logic: Cornell, 1957. In: Hintikka, J., et al. (eds.) Philosophy and Logic – In search of the Polish tradition. Synthese Library, vol. 323, pp. 151–162. Kluwer Acad. Publ., Dordrecht (2003), <http://math.stanford.edu/~feferman/papers/cornell.pdf>
- [Fre79] Frege, G.: Begriffsschrift. Louis Nebert, Halle (1879)
- [Gel59] Gelernter, H.: Realization of a geometry theorem-proving machine. In: Proc. First Intern. Conf. on Information Processing (IFIP), Paris, pp. 273–282. UNESCO House, (1959), Also contained in [SW83, 99–122]
- [Gil60] Gilmore, P.C.: A proof method for quantification theory: Its justification and realization. *IBM J. Research Develop.* 4, 28–35 (1960)
- [Gil70] Gilmore, P.C.: An examination of the geometry theory machine. *Artificial Intelligence* 1, 171–187 (1970)

- [Gol71] Goldfarb, W.D. (ed.): J. J. Herbrand — Logical writings. Reidel, Dordrecht (1971)
- [HA28] Hilbert, D., Ackermann, W.: Grundzüge der Theoretischen Logik. Springer, Heidelberg (1928)
- [Her30] Herbrand, J.J.: Recherches sur la théorie de la démonstration. In: Travaux Soc. Sciences et Lettres Varsovie, Cl. 3 (Mathem., Phys.) (1930), Engl. transl. in [Gol71]
- [Hin55] Hintikka, K.J.J.: Form and content in quantification theory. Acta Philosophica Fennica 8, 7–55 (1955)
- [HV71] Hoffmann, G.-R., Veenker, G.: The unit-clause proof procedure with equality. Computing 7(1-2), 91–105 (1971)
- [Kan57] Kanger, S.: Provability in Logic. PhD thesis, University of Stockholm (1957)
- [KK84] Kneale, W., Kneale, M.: The Development of Logic. Clarendon Press, Oxford (1984)
- [LSBB92] Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO — A high-performance theorem prover for first-order logic. Journal of Automated Reasoning 8(2), 183–212 (1992)
- [McC59] McCarthy, J.: The Wang algorithm for the propositional calculus programmed in LISP. In: McCarthy, J. (ed.) Symbol Manipulating Language Memo 14, Artificial Intelligence Project, MIT, Cambridge MA (1959), Quoted in [Wan60a, p.232]
- [MOW76] McCharen, J., Overbeek, R., Wos, L.: Problems and experiments for and with automated theorem proving programs. IEEE Transactions on Computers C-25, 773–782 (1976)
- [NSS56] Newell, A., Shaw, J.C., Simon, H.A.: The logic theory machine. IRE Trans. Information Theory IT-2, 61–79 (1956), Also contained in [SW83, 49-73]
- [Pap06] Papon, P.: Die Wissenschaft, Zeichen der Zeit. FTE Info – Magazin über europäische Forschung (An interview) 50, 9–11 (2006)
- [PPV60] Prawitz, D., Prawitz, H., Voghera, N.: A mechanical proof procedure and its realization in an electronic computer. J. ACM 7, 102–128 (1960)
- [Pra60] Prawitz, D.: An improved proof procedure. Theoria 26, 102–139 (1960), Also contained in [SW83, 159–199]
- [Rob57] Robinson, A.: Proving theorems (as done by man, logician, or machine). In: Summaries of Talks Presented at the Summer Institute for Symbolic Logic, Communic. Res. Div., Princeton, New Jersey, Institute for Defense Analysis (1957), Also contained in [SW83, 74–76]
- [Rob61] Alan Robinson, J.: Gamma I: A general theorem proving program for the IBM 704. Technical Report ANL-6447, Argonne National Laboratory, Chicago IL (1961)
- [Rob63] Alan Robinson, J.: Theorem proving on the computer. Journ. ACM 10(2), 163–174 (1963), Also contained in [SW83, 372–383]
- [Rob65] Alan Robinson, J.: A machine-oriented logic based on the resolution principle. Journal of ACM 12, 23–41 (1965), Also contained in [SW83, 397–415]
- [Rob00] Robinson, J.A.: PROOF=GUARANTEE+EXPLANATION. In: Hölldobler, S. (ed.) Intellectics and Computational Logic – Papers in Honor of Wolfgang Bibel. Applied Logic Series, vol. 19, pp. 277–294. Kluwer, Dordrecht (2000)
- [Sch56] Schütte, K.: Ein System des verknüpfenden Schließens. Archiv f. Mathematische Logik und Grundlagen der Wissenschaften 2, 55–67 (1956)

- [SW83] Siekmann, J., Wrightson, G. (eds.): Automation of Reasoning — Classical Papers on Computational Logic 1957-1966, vol. 1. Springer, Berlin (1983)
- [Vee63] Veenker, G.: Ein Entscheidungsverfahren für den Aussagenkalkül und seine Realisation in einem Rechenautomaten. *Grundl.stud. aus Kybernetik u. Geisteswiss* 4, 127–136 (1963)
- [Vee67] Veenker, G.: Beweisalgorithmen für die Prädikatenlogik. *Computing* 2(3), 263–283 (1967)
- [vOQ55a] van Orman Quine, W.: A proof procedure for quantification theory. *J. Symbolic Logic* 20, 141–149 (1955)
- [vOQ55b] van Orman Quine, W.: A way to simplify truth functions. *American Mathematical Monthly* 62, 627–631 (1955)
- [Wan60a] Wang, H.: Proving theorems by pattern recognition, Part I. *Comm. ACM* 3, 220–234 (1960), Also contained in [SW83, 229–243]
- [Wan60b] Wang, H.: Toward mechanical mathematics. *IBM Journ. Res. Develop.* 4, 2–22, Also contained in [SW83, 244–264] (1960)
- [WCR64] Wos, L., Carson, D., Robinson, G.A.: The unit preference strategy in theorem proving. In: *AFIPS Conf. Proc., Washington DC*, vol. 26, pp. 615–621. Spartan Books (1964)
- [WH83] Wos, L., Henschen, L.: Automated theorem proving 1965–1970. In: Siekmann, J., Wrightson, G. (eds.) *Automated Reasoning 2 – Classical Papers on Computational Logic 1967–1970*, vol. 2, pp. 1–24. Springer, Berlin (1983)

Cognitive Technical Systems — What Is the Role of Artificial Intelligence?

Michael Beetz¹, Martin Buss², and Dirk Wollherr²

¹ Institute of Automatic Control Engineering (LSR),
Faculty of Electrical Engineering and Information Technology

² Intelligent Autonomous Systems,
Department of Informatics
Technische Universität München
D-80290 München, Germany
www.cotesys.org

Abstract. The newly established cluster of excellence CoTESYS [1] investigates the realization of cognitive capabilities such as perception, learning, reasoning, planning, and execution for technical systems including humanoid robots, flexible manufacturing systems, and autonomous vehicles. In this paper we describe cognitive technical systems using a sensor-equipped kitchen with a robotic assistant as an example. We will particularly consider the role of Artificial Intelligence in the research enterprise.

Key research foci of Artificial Intelligence research in CoTESYS include (◦) symbolic representations grounded in perception and action, (◦) first-order probabilistic representations of actions, objects, and situations, (◦) reasoning about objects and situations in the context of everyday manipulation tasks, and (◦) the representation and revision of robot plans for everyday activity.

1 Introduction

The newly established cluster of excellence CoTESYS [1] (Cognition for Technical Systems) investigates the realization of cognitive capabilities such as perception, learning, reasoning, planning, and execution for technical systems including humanoid robots, flexible manufacturing systems, and autonomous vehicles (off-road vehicle and blimb). One of our ultimate goals is, as Brachman [2] puts it in the context of general cognitive systems, to turn technical systems into “ones that can reason using substantial amounts of appropriately represented knowledge, learn from its experience so that it performs better tomorrow than it did today, explain itself and be told what to do, be aware of its own capabilities and reflect on its own behavior, and respond robustly to surprise.” A technical system that is cognitive in this sense will be more reliable, flexible, adaptive, and robust. These kinds of systems ease interaction and cooperation with humans.

¹ CoTESYS is funded by the German Research Council DFG as a research cluster of excellence within the “excellence initiative” from 2006-2011. CoTESYS partner institutions are: Technische Universität München (TUM), Ludwig-Maximilians-Universität (LMU), Universität der Bundeswehr (UBM), Deutsches Zentrum für Luft- und Raumfahrt (DLR), and Max-Planck-Institute for Neurobiology (MPI), all in Munich.

Thus, we consider *Cognitive technical systems (CTS)* to be information processing systems equipped with artificial sensors and actuators, integrated and embedded into physical systems, and acting in a physical world. *CTSs* differ from other technical systems as they perform *cognitive control* and have *cognitive capabilities*. *Cognitive control* orchestrates reflexive and habitual behavior in accord with longterm intentions. *Cognitive capabilities* such as perception, reasoning, learning, and planning turn technical systems into systems that “*know what they are doing*” [2].

This paper sheds light on the role of Artificial Intelligence for the CoTESys cluster and cognitive technical systems in general, on methods of Artificial Intelligence that we believe to apply well to cognitive technical systems, and on the challenges that cognitive technical systems present to Artificial Intelligence research.

2 Why Cognitive Technical Systems Are Not Merely AI-Based Technical Systems

Looking at the notion of cognitive technical systems we have put forward in the introduction one might be tempted to simply view them as a subfield of Artificial Intelligence. This view, however, is neither valid nor useful. Let us explain why.

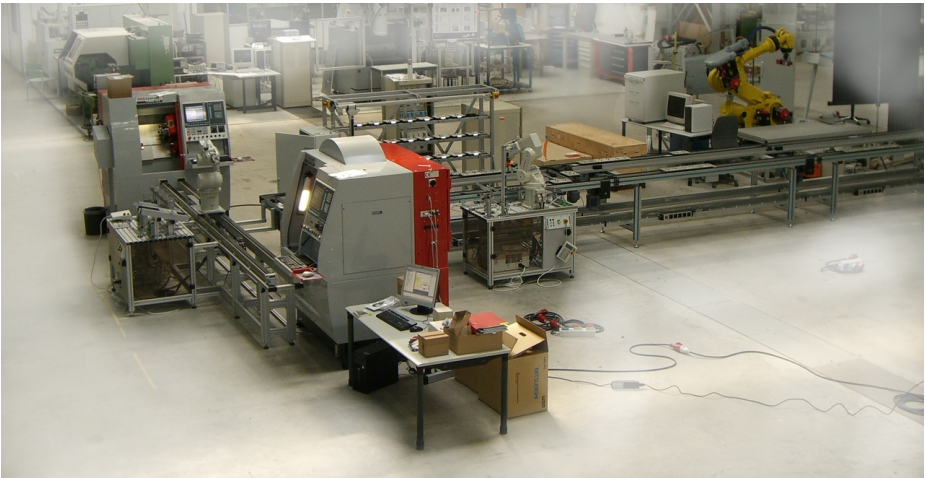


Fig. 1. Demonstrator Cognitive Factory: A flexible manufacturing system consisting of two CNC machines, an assembly robot, a material handling robot, an automatic storage unit, a computer controllable conveyor belt, and a quality measuring unit is further enhanced with sensor networks and high performance 3D laser sensors. Cognitive mechanisms enable the factory system to learn situation-specific models of production steps, to estimate the state of manufacturing processes, to dynamically reschedule individual production steps to react to perceived disturbances and respond to dynamically changed objective functions. The system also provides human process planners with much more informative models of production processes that are learned from experience.

2.1 Realizing Cognitive Technical Systems

A main focus of the COTESYS cluster is the realization of a cognitive factory [3] (see Figure 1), humanoid robots in sensor-equipped environments [4,5,6,7,8,9,10,11,12,13] (see Figure 2), and autonomous vehicles (see Figure 3) as the main demonstration platforms that are to exhibit cognitive capabilities. As successfully demonstrated in a number of other leading-edge research projects demonstration platforms and their accompanying demonstration scenarios can serve as the main driving forces for the research in the cluster (cf. [14,15,16,17]).

The COTESYS demonstrators and scenarios are designed to challenge fundamental as well as applied research in the areas of perception, knowledge and learning, reasoning and planning, interaction, and execution. Individual cognitive capabilities are to be integrated into complete control systems and embedded within the demonstrators. The demonstrator research ensures that the cluster is not producing isolated pieces of software but rather software components that function as part of an integrated cognitive system. The demonstrators thereby enforce researchers of different institutions and disciplines to cooperate in order to achieve the planned demonstration scenarios. The breadth of the demonstrators also reduce the risk of overfitting cognitive mechanisms to overly specific contexts because the demonstrators challenge the research along different dimensions.



Fig. 2. Demonstrator platforms humanoid robots. The COTESYS cluster investigates ways to equip the walking machine Johnnie (left, ©Prof. Ulbrich, TUM) and its currently developed successor Lola and Justin (right, ©Prof. Hirzinger, DLR), a humanoid upper body system for two handed manipulation, with cognitive capabilities. Intermediate research platforms are existing mobile robots with manipulators such as the B21 robot depicted in the middle acting as a kitchen assistant. Scenarios in which cognitive capabilities will be demonstrated include household work, party service, and shopping.

The realization of complex demonstration scenarios on these technical systems requires immense concerted efforts of researchers in the engineering and computational sciences. On the engineering side the systems need to be designed, modeled, and analyzed carefully. On the computational side we need comprehensive communication and computational infrastructure that combines the various hard- and software components for perception, reasoning and learning, and acting. It goes without saying that doing one without the other is doomed to fail.

The focus on demonstrators and integrated system research is also important as a research paradigm. The cognitive capabilities of CTSs enable them to reason about the use of their information processing mechanisms: they can check results, debug them, and apply better suited mechanisms if default methods fail. Therefore, their information processing mechanisms do not need to be hard coded completely. They should still be correct and complete but through dynamic adaptation rather than static coding. This is important because in all but the simplest cases completeness and correctness come at the cost of those problems becoming unsolvable — computationally intractable at best. For example, computing a scene description from a given camera image is an ill-posed problem [18], checking the validity of statements in first-order logical theories is undecidable, computing a plan for achieving a set of goals is intractable for all but the most trivial action representations.



Fig. 3. Demonstrator platforms vehicles. The autonomous automobile MuCAR-3 (left, ©Prof. Wünsche,) and the DLR blimb (right, ©Prof. Hirzinger, DLR) are planned to cooperate in specified rescue scenarios.

2.2 Motor Control in Natural Systems

A second aspect where the research area of technical cognitive systems goes beyond the scope of most AI research is that much of the inspiration of how technical systems are to be equipped with cognitive mechanisms is taken from the research in the cognitive sciences, in particular those areas that study computational models of perception and attention and motor control [2].

Industrial robots are faster, more accurate, and stronger than humans. Yet many manipulation tasks that are easily performed by humans as part of their everyday activities are well beyond the capabilities of such robots. The main reason for this superiority is that humans have a brain, an information and control mechanism tailored for flexible, reliable, and adaptive motion control. As we are working towards autonomous service robots operating and performing manipulation in the presence of humans and in human living and working environments, the robots must exhibit similar levels of flexibility, reliability, and adaptivity.

² See http://www.foresight.gov.uk/previous_projects/cognitive_systems/index.html for a comprehensive discussion of this and related subjects.

Animals and humans deal easily with everyday situations – an ability technical systems currently lack. Unlike artificial systems, they develop and learn how to extract and incorporate new information from the environment. Animals have survived in our complex world by developing brains and adequate information processing strategies [19]. Brains cannot compete with computers on tasks requiring raw computational power. However, they are extremely well-suited to deal with ill-structured problems that involve a high degree of unpredictability, uncertainty, and fuzziness. They can easily cope with an abundance of complex sensory stimuli that have to be transformed into appropriate sequences of motor actions.

Because brains of humans and non-human primates have successfully developed information processing mechanisms to overcome many of the limitations of technical systems, CoTeSys studies and analyzes cognition in natural systems and transfers the respective insights into the design and implementation of cognitive control systems for technical systems [20].

To this end, cognitive scientists study the neurobiological and neurocognitive foundations of cognition in humans and animals and develop computational models of cognitive capabilities that explain their empirical findings. These computational models will then be studied by the CoTeSys engineers and computer scientists with respect to their applicability to artificial cognitive systems and empirically evaluated in the context of the CoTeSys demonstrators.

2.3 The CoTeSys Approach

Recognizing the situation that the successful realization of CTSs require AI methods as well as solid grounding in physical systems and insights from the cognitive sciences, CoTeSys structures interdisciplinary research on cognition in three closely intertwined research threads, which perform fundamental research and empirically study and implement cognitive models in the context of the demonstration testbeds. The research threads are:

1. Systemic Neuroscience, Cognitive Science, and Neurocognitive Psychology which develop computational models of cognitive control, perception, and motor action based on experimental studies at the behavioral and brain level.
2. Information processing technology, which studies and develops algorithms and software systems for realizing cognitive capabilities. Particularly relevant are modern methods from Control and Information Theory, Artificial Intelligence including learning, perception, and symbolic reasoning.
3. Engineering technologies, which investigate research problems in the areas of mechatronics, sensing technology, sensor fusion, smart sensor networks, control rules, controllability, stability, model/knowledge representation, and reasoning needed to implement robust cognitive abilities in technical systems with guaranteed performance constraints.

In recent years, these disciplines studying cognitive systems have crossfertilized each other in various ways. Researchers studying human sensorimotor control have found convincing empirical evidence for the use of Bayes estimation and cost function enabled control mechanisms in natural movement control [19]. Bayesian networks and the

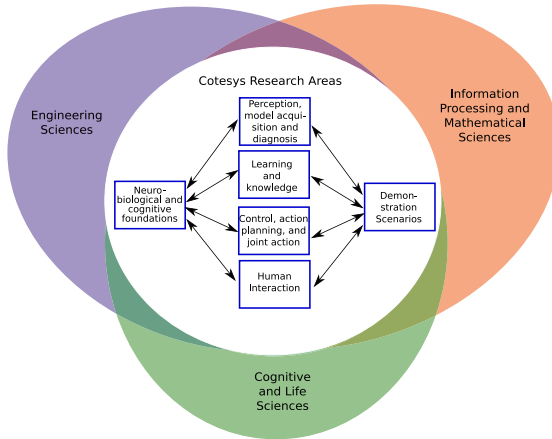


Fig. 4. CoTESYS research strategy: Three research disciplines (cognitive and life sciences, information processing and mathematical sciences, and engineering sciences) work synergetically together to explore cognition for technical systems. Research is structured into three groups of research areas: cognitive foundations, cognitive mechanisms, and demonstration scenarios. Cognitive mechanisms to be realized include perception, reasoning and learning, action selection and planning, and joint human/robot action.

associated reasoning and learning mechanisms have inspired research in cognitive psychology in particular the formation of causal theory with young children [21|22|23|24]. Functional MRI images of rat brains have shown neural activation patterns of place cells similar to multimodal probability distributions in robot localization using Bayesian filters [25].

The conclusions that CoTESYS draws from these examples are that (1) successful computational mechanisms in artificial cognitive systems tend to have counterparts with similar functionality in natural cognitive systems; and (2) new consolidated findings about the structure and functional organization of perception and motion control in natural cognitive systems show us much better ways of organizing and specifying computational tasks in artificial cognitive systems.

Cognition for technical systems is not the mere rational reconstruction of natural cognitive systems. Natural cognitive systems are impressively well adapted to the computational infrastructure and the perception and action capabilities of the systems they control. Technical cognitive systems have computational means, perception and action capabilities with very different characteristics. Learning and motor control for reaching and grasping provide a good case in point. While motor control in natural systems takes up to 100ms to receive motion feedback, high end industrial manipulators execute feedback loops at 1000Hz with a delay of 0.5ms. In contrast to robot arms, control signals for muscles are noisy and muscles take substantial amounts of time to produce the required force. On the other hand, antagonistic muscle groups support the achievement of equilibrium states. Thus, where in natural systems predictive models of motion are required because of the large delay of feedback signals, robot arms can perform the same kind of motions better by using fast feedback loops without resorting to prediction.

Because of these differences, we cannot expect that generally all information processing mechanisms optimized for the perceptual apparatus, the brain, and the limbs of humans or non-human primates will apply, without modification, to the control of CTSs.

3 Cognition in the Perception-Action Loop

COTESYS investigates the cognition in technical systems in terms of the cognition-based perception-action closed loop. Figure 5(left) depicts the system architecture of a cognitive system with multi-sensor perception, cognition (learning, knowledge, action planning), and action. COTESYS research is dedicated to real-time performance of this control loop. On the higher level, key components comprise environment models, learning and knowledge management, all in real-time and tightly connected to physical action.

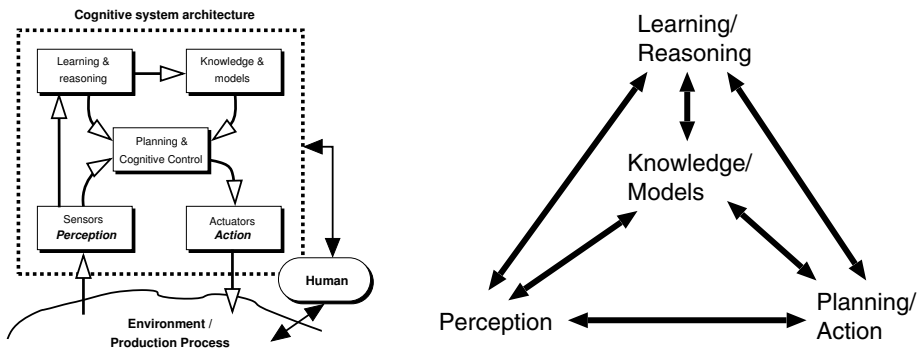


Fig. 5. The cognitive system architecture: The perception-action closed loop (left) and the interplay of the cognitive capabilities (right)

The mapping of the technical system operation onto the perception-action cycle depicted in Figure 5(left) might suggest that we functionally decompose cognition into modules where one module performs motor action, another one reasoning, and so on. In order to achieve the needed synergies, the coupling of the different cognitive capabilities must be much more intense and interconnected as depicted in Figure 5(right). For example, the system can *learn to plan* and *plan to learn*. It can *learn* to plan more reliably and efficiently and also *plan* in order to acquire informative experiences to learn from. Or, perception is integrated into action to perform tasks that require hand-eye coordination. Further, perception often requires action to obtain information that cannot be gathered passively.

COTESYS investigates the perception-action loop within a highly interdisciplinary research endeavor starting with discipline-specific views of the loop components in order to obtain a common understanding of key concepts, such as *perception*, (*motor*) *action*, *knowledge and models*, *learning*, *reasoning*, and *planning*.

Perception is the acquisition of information about the environment and the body of an actor. In cognitive science models, part of the information received by the receptors is

processed at higher levels in order to produce task-relevant information. This is done by recognizing, classifying, and locating objects, observing relevant events, recognizing the essence of scenes and intentional activities, retrieving context information, and recognizing and assessing situations [26,27]. In control theory, perception strongly correlates with the concept of observation — the identification of system states that are needed to generate the right control signals. Artificial intelligence, a subfield of computer science, is primarily concerned with perception and action; perception is often framed as a probabilistic estimation problem [28] and the estimated states are often transformed into symbolic representations that enable the systems to communicate and reason about what they perceive [29].

(Motor) Action is the process of generating behavior to change the world and to achieve some objectives of the acting entity [31]. To produce action, primate brains use a quasi-hierarchy ranging from elementary motor elements at lower cortical levels to complex “action” sequences and plans at higher levels. Natural cognitive systems use internal forward models to predict the consequences of motor signals to account for delays in the computation process and filtering out uninformative incoming sensory information [19]. This cognitive science view can be contrasted to control theory, where behavior is specified in terms of control rules. Control rules for feedback control are derived from accurate mathematical dynamical system models. The design of control rules aims at control systems that are controllable, stable, and robust and can thereby provably satisfy given performance requirements [32]. Action theories in Artificial Intelligence typically abstract from many dynamical aspects of actions and behavior in order to handle more complex tasks [33]. Powerful computational models have been developed to rationally select the best actions (based on decision theory criteria) [28], to learn skills and action selection strategies from experience [34], and to perform action aware control [35].

Knowledge (Models) in cognitive science is conceived to consist of both declarative and procedural knowledge [3]. Declarative knowledge is recognizing and understanding factual information known about objects, ideas, and events in the environment. It also contains the inter-relationships between objects, events, and entities in the environment. Procedural knowledge is information regarding how to execute a sequence of operations. In cognitive science various models have been proposed as part of computational models of motor control and learning to explain behavior of human and primate behavior in empirical studies [19,37]. Most prominent are the forward and backward models of actions for the prediction of the actions’ effects and sensory consequences and for the optimization of skills [38]. Graphical models have been proposed to explain the acquisition of causal knowledge with younger children [23]. In control systems, various mathematical models, such as differential equations or automata that capture the evolution of dynamical systems, are used [39]. Research in Artificial Intelligence has produced powerful representations for joint probability distributions and

³ Brown and Rosenbaum characterize *motor control* as “the ability of biological and artificial systems to plan, initiate, maintain, monitor, and correct movements to attain physically realizable goals. A *model* is a system or process that permits predictions.” [30].

⁴ Again we refer to the research review performed by the Foresight project [36] for a comprehensive introduction into the area of representation in the brain.

symbolic knowledge representation mechanisms. It has developed the mechanisms to endow *CTSs* with encyclopedic [40,41] and common sense knowledge [42].

Learning is the process of acquiring information, and, respectively, the reorganization of information that results in new knowledge. The learned knowledge can relate to skills, attitudes, and values and can be acquired through study, experience, or being taught — the cognitive science view. Learning causes a change of behavior that is persistent, measurable, and specified. It is a process that depends on experience and leads to long-term changes in behavior. In control theory, adaptive control investigates control algorithms in which one or more of the parameters varies in real time, to allow the controller to remain effective in varying process conditions. Another key learning mechanism is the identification of parameters in mathematical models. In Artificial Intelligence, a large variety of information processing methods for learning have been developed. These mechanisms include classification learners, such as decision tree learners or support vector machines, function approximators, such as artificial neural networks, sequence learning algorithms, and reinforcement learners that determine optimal action selection strategies for uncertain situations [43]. The learning algorithms are to be integrated into comprehensive systems that automatically collect the experience needed for self-improvement, that perform lifelong learning, and can improve system behavior even if very little experience is available⁵

Reasoning is a cognitive process by which an individual or system may infer a conclusion from an assortment of evidence, or from statements of principles. In the cognitive sciences reasoning processes are typically studied in the context of complex problem solving tasks, such as solving student problems, using protocol analysis methods (“think aloud”). In the engineering sciences specific reasoning mechanisms for prediction tasks, such as Bayesian filtering, are employed and studied [28]. Other reasoning tasks are solved in the system design phase by the system engineers, where control rules are proven to be stable. The resulting systems have no need for execution time reasoning because of their guaranteed behavior envelope. Artificial intelligence has developed a variety of reasoning mechanisms, including causal, temporal, spatial, and teleological reasoning, which enables *CTSs* to solve dynamically changing, interfering, and more complex tasks.

Planning is a process of generating (possibly partial) representations of future behavior, prior to the use of such plans, to constrain or control current behavior. It comprises reasoning about the future in order to generate, revise, or optimize the intended course of action. In Artificial Intelligence, we view plans as control programs that can be executed, be reasoned about, and be manipulated [44]. A key area where synergies between the engineering and information processing disciplines are to be expected is the tight coupling of symbolic action planning on the one side and manipulation with manipulation and path planning on the other one [45,46,47]. Another area where breakthroughs

⁵ The view of learning being integral parts of cognitive systems and requiring additional cognitive mechanisms is pushed by the DARPA IPTO office (<http://www.darpa.mil/ipto/>). Here we find funded research programs for *personalized assistants that learn, integrated learning, transfer learning, and learning applied to ground robots*.

are to be expected is the coupling of planning and learning. Here we expect plan-based robot control systems that learn to plan complex manipulation tasks both more reliably and efficiently and planning systems that help autonomous robots to acquire complex manipulation plans by incorporating planning mechanisms into the learning apparatus. The synergies to be expected from these combinations will be preconditions for robotic assistants that can competently perform everyday manipulation tasks in the presence of people and in human living environments [48].

4 The Assistive Kitchen as a Cognitive Technical System

Let us now look at the Assistive Kitchen as an example of the kind of cognitive technical systems we have in mind [13]. The *Assistive Kitchen* (Figure 6) is a ubiquitous computing, sensing, and actuation environment with a robotic assistant that aims at

- supporting and assisting people in their household chores through physical action;
- enhancing the cognitive capabilities of people by reminding them; and
- monitoring health and safety of the people.

To achieve these objectives, the Assistive Kitchen is to

- perceive, interpret, learn, and analyze models of household chore; and
- represent the acquired models such that the Assistive Kitchen can use them for activity and safety monitoring, health assessment, and for adapting itself to the needs and preferences of the people.

The Assistive Kitchen includes an autonomous robotic agent that is to learn and perform complex household chores. The robot must perform housework together with people or at least assist them in their activities. This requires safe operation in the presence of humans and behaving according to the preferences of the people they serve.

4.1 Infrastructure

We start with the hardware and software infrastructure of the kitchen, which consists of a mobile robot and networked sensing and actuation devices that are physically embedded into the environment.

Currently, an autonomous mobile robot with two arms with grippers acts as a robotic assistant in the Assistive Kitchen (see Figure 6). The robot is a RWI B21 robot equipped with a stereo CCD system and laser rangefinders as its primary sensors. One laser range sensor is integrated into the robot base to allow for estimating the robot's position within the environment. Small laser range sensors are mounted onto the robot's grippers to provide sensory feedback for reaching and grasping actions. The grippers are also equipped with RFID tag readers that support object detection and identification. Cameras allow for longer range object recognition and vision-based interaction with people.

The robot can manipulate objects and its environment using its two industrial strength Amtec Powercube arms with simple grippers. While the arms and grippers are not very dexterous they permit the execution of simple manipulation actions such as getting plates and glasses out of the cupboard and putting them onto the table.



Fig. 6. The Assistive Kitchen containing a robot and a variety of sensors. The sensors provided by the environment (left) in the form of a sensor network include laser range finders, RFID tag readers, force and acceleration sensors, and CCD cameras. The mobile manipulation platform is equipped with two robot arms with grippers, and various sensors, most notably a stereo camera unit and small laser range sensors mounted on the robot grippers.

The sensor-equipped kitchen environment (see Figure 6) disposes of global environment sensors, sensor-equipped furniture, web-enabled appliances, “smart” objects, and instrumentation for people acting in the environment [49]. In this section we will look at these components in more detail.

We have mounted a set of static off-the-shelf cameras positioned to cover the kitchen area with high resolution in critical working areas. With these cameras, actions of the people and robots can be tracked from different locations to allow for more accurate positioning and pose estimation. In addition, laser range sensors are mounted at the walls for covering large parts of the kitchen. They provide accurate and valuable position data for the people present in the environment and their movements within the kitchen.

The furniture is also equipped with various sensors. For example, we have cupboards with long-range RFID tag readers that enable the cupboards to “know” the identities of the RFID tagged objects that are inside of them. They also have magnetic contact sensors that sense whether the cupboard doors are open or closed. The table contains several integrated capacitive sensors that report the presence of objects based on local capacitance, while the RFID readers provide exact information on what object was placed there.

Web-enabled kitchen appliances such as the refrigerator, the oven, the microwave, and the faucet, allow for remote and wireless monitoring and control.

In addition, kitchen utensils, tools and small appliances are equipped with integrated sensors. For example, we use a knife (see Figure 7) instrumented with a 6DOF force/torque sensor that allows us to record the force trajectories over extended periods of time. Because the shapes of the force trajectories are characteristic for the physical properties of the objects, we can learn object specific force profiles and use them to classify the objects being cut.

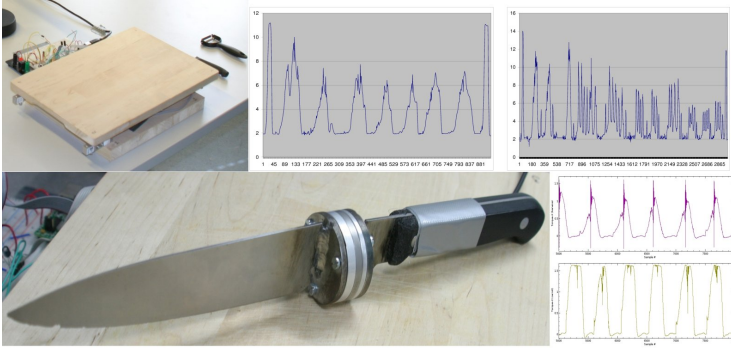


Fig. 7. Knife with embedded force sensors networked within the wireless sensor network. The graphs show force profiles characteristic for cutting objects of particular classes such as bananas or apples.

Another smart object is a sensor-instrumented coffee machine. The integrated sensors provide information whether the filter unit is open and about whether a coffee filter is installed, whether the machine is switched on or off, the amount of water in the water container, etc.

Small ubiquitous devices offer the possibility to instrument people with additional sensors. In our case, we have built a glove equipped with an RFID tag reader (see Figure 8) that enables us to identify the objects that are manipulated by the person wearing it. In addition, the person is equipped with tiny inertial measurement units that provide us with detailed information about the person's limb motions.

The sensors in the Assistive Kitchen are connected into distributed sensor networks (see Figure 8), which are enhanced with cognitive capabilities. To this end the sensors are wirelessly connected to small ubiquitous computing devices (like Gumstix) and to personal computers that perform state estimation and data-mining tasks. This way, activity data can be collected and abstracted into models in a distributed manner. Cognitive capabilities of the network include the estimation of meaningful states from sensor data, the continual acquisition, update, and use of activity models, and the estimation of context conditions that allow for the simplification of recognition tasks. Because of these capabilities cognitive sensor networks always have up-to-date models of objects, situations, and activities, which enable the networks to provide a continual service for answering queries about the environment and the activities that take place in it.

Cognitive sensor networks can estimate states and recognize events that are meaningful in the application domain but must be obtained by combining, interpreting, and abstracting the sensor data of different sensors over time. For example, mounting RFID tags and acceleration sensors to kitchen utensils allows the environment to recognize *force-dynamic states* such as an object being picked up, carried, or put down. The recognition of force-dynamic states is essential for segmenting activities into meaningful subactions. The networks can also learn about places that play particular roles in the activities that are monitored. The system can learn where the people prepare food or where food is stored, etc.

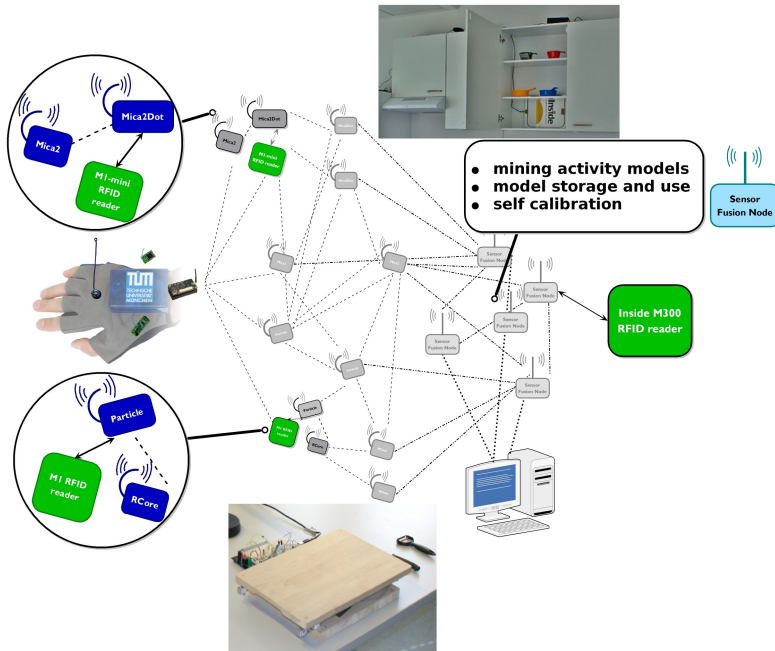


Fig. 8. Sensor networks in the Assistive Kitchen

4.2 Perception

There is a large number of perception tasks that need to be accomplished by the sensor equipped kitchen and the robotic assistant. The sensor-equipped environment has to recognize, classify, and estimate the parameters of everyday manipulation activities, such as setting the table, cooking, and cleaning up. It has to determine the objects relevant for tasks — even in cluttered scenes, classify them, determine their positions and orientation, and grasp points to manipulate them. The robot has to perceive possibly using the sensor network as an additional resource the semantically meaningful components of the environment both very accurately and abstractly. Finally, a variety of perceptual capabilities with respect to people are needed including the detection and recognition of people, understanding human body language (mimics, gestures), etc.

In the remainder of the Section we focus on two perceptual tasks that are particular interesting from the point of view of Artificial Intelligence: the perception of human actions and activities and the perception of human living environments.

The Perception of Human Actions and Activities. The challenge here is that the sensor-equipped kitchen has to recognize and interpret human activities over extended periods of time, meaning days or even weeks [50]. While being switched on the system has to recognize high-level activities such as setting the table, having a meal, cooking, and cleaning up. Activities can be performed concurrently such as cleaning and cooking and can be interrupted, for example when a visitor comes. Activities have to be decomposed into discrete actions such as putting a plate on the table in order to recognize how

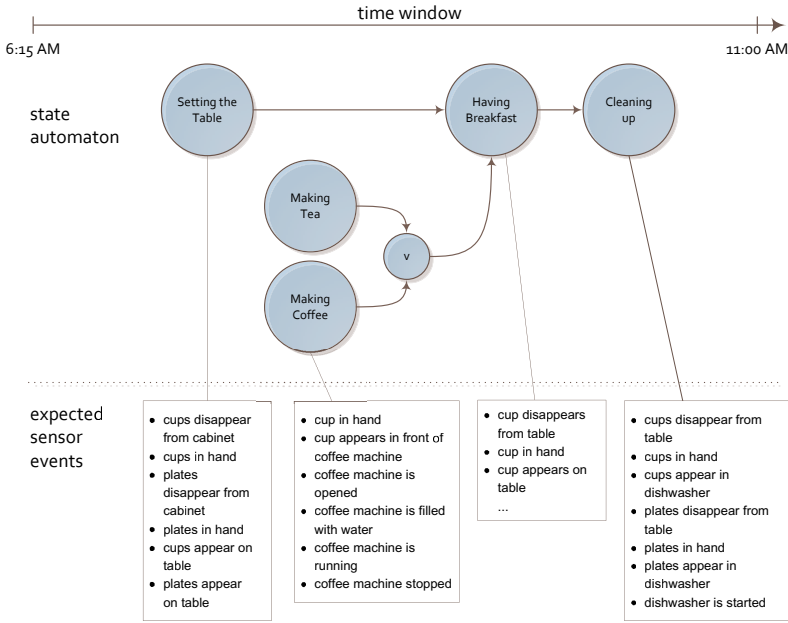


Fig. 9. RFID based recognition of everyday activities based on the disappearance of plates and cups in the cabinet, on the table, and in the dishwasher

activities are performed. In particular, the perception system has to recognize that people carry stacks of plates to set the table instead of carrying the plates one by one. Finally, to enable effective imitation learning by the robotic assistant the perception system has to determine the precise parameters of motion trajectories of manipulation tasks.

For the reliable longterm observation of high-level everyday activities it is in our view essential that we realize a perception system that generates streams of high-level events that are strongly correlated with the activities to be recognized [51]. We believe that use of vision algorithms that operate on image streams would be brittle and demand very high computational resources, because the amount of sensor data interpreted is immense and the pixel information is only weakly correlated with the activities to be recognized [52].

Therefore, we build the high-level activity recognition system based on RFID tag readers mounted in the environment and extend them such that they signal objects of particular types appearing and disappearing in the sensing range of the respective RFID tag readers. Based on these perceptual events we can characterize setting the breakfast table as plates, cups, and other utensils disappearing from cabinets and drawers and appearing on the breakfast table. Researchers from the Intel Lab in Seattle and the University of Washington have pioneered this type of activity recognition by pointing out the strong correlations between the activities of daily life and the objects used [53].

Another important perceptual task is the precise segmentation and decomposition of activities into meaningful actions, such as picking up an object and putting it on a table. Cognitive models suggest that this segmentation could be done by physically interpreting activity data using the concepts of force-dynamic states and events. The basic idea of force-dynamic models of actions is that in these models we can use the state *holding* as the invariant that discriminates putting objects from many other actions that might be similar with respect to features. Evidence that force-dynamic states are appropriate to classify everyday manipulation actions can also be found in the human language that is typically particularly rich with respect to force-dynamic variants of action executions (e.g., push, pull, draw, ...). In this model we can represent a pickup event as follows:

$$\text{pickup}(\text{hand}, \text{object}, \text{support}) \iff$$

$$\text{supports}(\text{support}, \text{object}) \wedge \text{contacts}(\text{support}, \text{object})$$

followed by

$$\text{supports}(\text{hand}, \text{object}) \wedge \text{attached}(\text{hand}, \text{object})$$

A big advantage of using force-dynamic states and events as our basic concepts for action recognition is that using distributed sensor networks we can build relatively simple special purpose sensors that can detect force dynamic states reliably and with high accuracy. To do this we mount RFID tags and acceleration sensors to every object relevant for the activities. We also use a glove with an integrated RFID tag reader and acceleration sensor. Using this sensor setting we can recognize the force-dynamic event of picking up object *o* with the right hand *h* by looking at similar acceleration profiles of *h* and *o* and the low-range RFID tag reader of *h* reporting the RFID of *o*.

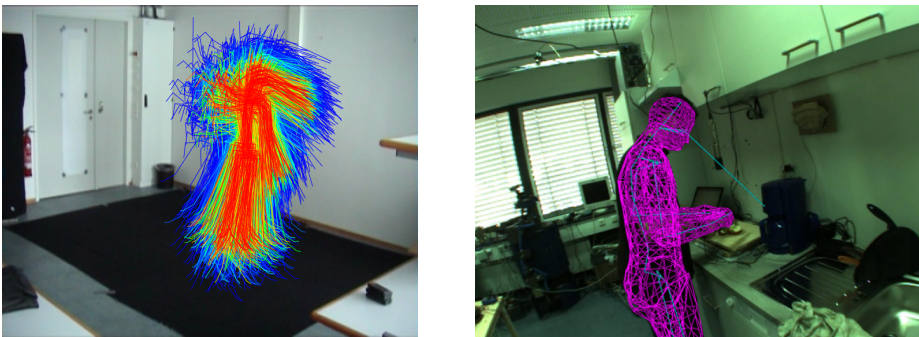


Fig. 10. 3D vision-based full-body motion tracker. Particles representing full body poses are depicted in the left subfigure. A digital human model with the parameters of the most probable particle is depicted on the right.

Finally, accurate motion and trajectory data are obtained by tracking a digital human model in multiple image streams recorded by the cameras installed in the sensor-equipped kitchen (see Figure 10). This accurate motion tracking requires a concerted

effort from AI technology. Learning components acquire appearance models of the person to be tracked and if possible of body parts. These appearance models are then used by the visual segmentation algorithms and the sensing evidence is probabilistically combined using an efficient, high-dimensional particle filter for tracking full-body motion.

The Perception of Kitchen Environments. The perception of living environments needed for the assistive kitchen presents other key challenges. Here, the perception system has to recognize the objects that the robotic assistant has to manipulate. At an abstract task-oriented level a household robot could consider the environment to consist of containers with front doors. Some of them have specific purposes such as the fridge or the dishwasher. Other task-relevant objects are tables, such as the working plate, drawers, and shelves.

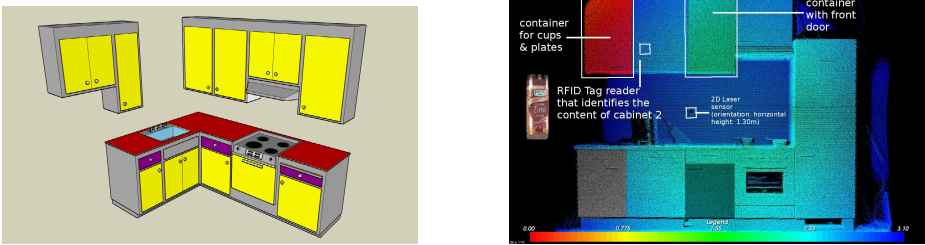


Fig. 11. Point cloud based 3D object model of the kitchen environment of our demonstration scenario. Objects of the classes cabinet (container with front door), tables, and drawers are shown in different colors. A 3D object model inferred from laser scans is shown on the right.

We are developing a perceptual system for this task that uses a precise laser range scanner on a robot arm, CCD cameras, and the wireless sensor network as its sensing resources. The key features of this perception system are that it very reliably removes noise and outliers in point clouds generated by 3D laser scans [54]. The mechanisms are also capable of appropriately filling holes in the object models that are typically caused by occlusions. This accuracy is needed to segment the front of an integrated kitchen into doors of the individual cabinets. Another key feature of this perceptual apparatus is that it can use object states to correctly classify objects. For example, to classify that an object is a cabinet the perceptual component must have seen the cabinet in an open and a closed state.

4.3 Model Acquisition

COTESYS approaches the acquisition, management, and use of activity models using the combination of first-order statements and probabilistic interpretation of statements as the basic representational apparatus.

Thus, the sensor data acquired by the sensor network in the course of observing kitchen activities are abstracted into learning task-specific experiences and stored into a relational database. The relational database together with causal links between domain

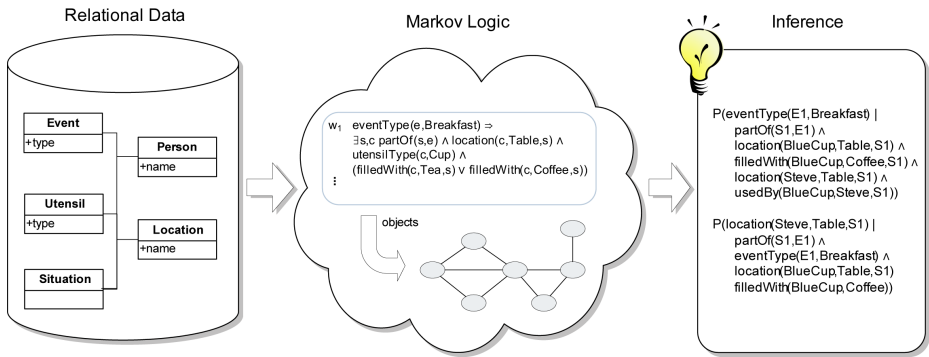


Fig. 12. Acquisition and use of a Markov Logic-based model of kitchen activities. The sensor data acquired by the activity recognition system are stored in a relational database system. The data in the database together with the causal structure on domain relations imply a joint probability distribution over relations in the activity domain. This distribution is represented using Markov logic, which allows for inferring the conditional probability of first-order statements.

relations then imply a probability distribution over everyday activities, their parameterized execution, and characteristics. These models of activities are then acquired by applying relational learning mechanisms to the given data and causal structure (see Figure 12) (cf. [55][56]).

Environment models are resources of the household robot that enable it to better perform its jobs. As such they must enable the robot to better infer where things are that it needs, how to manipulate objects (e.g., open doors), what kinds of objects are stored in which cabinets, etc.

To this end, a cabinet identified by the perception module must be associated with additional knowledge including what kinds of objects are stored in it, which sensor of the sensor network reports the content of this cabinet, how is the content of the cabinet organized. Another aspect of knowledge that the identified object models must be associated with are the roles that objects play in activities. For example, the robot might need to know the cup that Martin uses for breakfast, or the table where food is prepared. This knowledge is represented using description logics based representations in conjunction with datamining and learning mechanisms [57]. It is evident that models that are effective resources for the robot's activities must combine environment and activity models.

4.4 Execution

One of the hardest challenges for execution is the question of how an autonomous robot can perform everyday activities such as setting the table, preparing meals, cleaning up as flexibly, reliably, and efficiently as it is required in doing household work over extended periods of time. The performance of the robotic assistant critically depends on the design and implementation of its activity plans.

A key issue of plan execution is the design of plan languages and plans that do not abstract away too much important information as it is typically done in STRIPS-like [58] and PDDL-like plan languages [59]. In order to achieve high performance robot behavior plan execution mechanisms must address optimal parameterizations of control routines and the reliable execution of tasks [60]. Consider an office delivery robot navigating through its environment. If we look at the state of the art, in particular when the robots use sonar sensors, then the robots are capable of navigating through the hallway quickly, but have problems with traversing doorways. Consequently, looking ahead and preparing for smooth and optimal doorway traversals has more impact on performance than tour optimization. However, these improvements cannot be achieved when considering door traversal as a blackbox independent of its parameterization and situational context.

Making plans tolerant of sensor error, execution failures, and changing environments requires them to specify how to respond to asynchronously arriving sensory data and other events. They must also prescribe concurrent and synchronized actions. Many control patterns other than those provided by common plan representations have been proven to be necessary for flexible and reliable robot control. Plans cannot abstract away from the fact that they generate concurrent, event-driven control processes without the robot losing the capability to predict and forestall many kinds of plan execution failures.

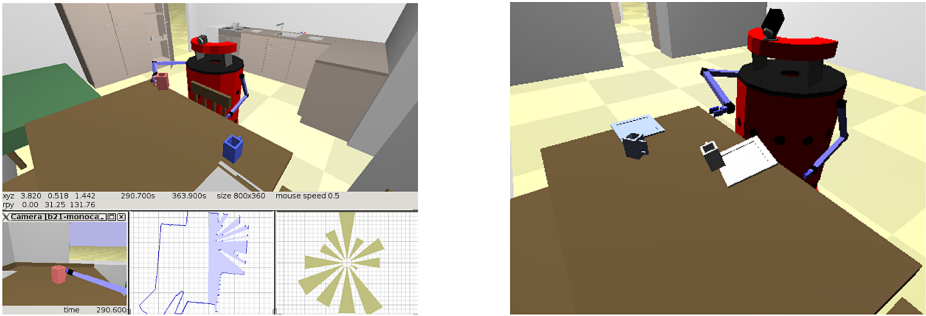


Fig. 13. Simulated robot acting in the kitchen (left) and a typical execution failure that the robot has to correct and to recover from (right)

4.5 Reasoning and Planning

Robotic agents can not be fully programmed for every application. Thus, in this demonstration scenario we realize robot control programs that specialize to their respective robot platform, work space, and tasks (see Figure 14).

Specifically we realize a high-level control program for setting the table. The program learns from experience where to stand when taking a glass out of the cupboard, how to best grasp particular kitchen utensils, where to look for particular cutlery, etc. This requires the control system to know the parameters of control routines and to have models of how the parameters change the behavior [60]. Also, the robots are required to perform their tasks over extended periods of time, which asks for very robust control.

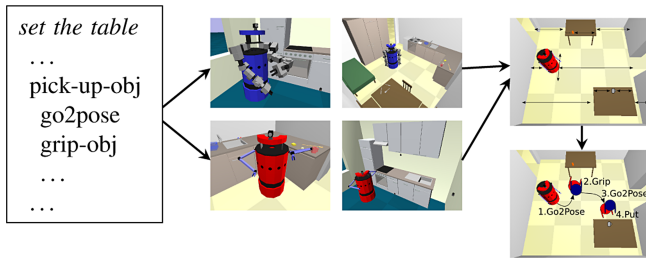


Fig. 14. Self-adaptation of different robots in different kitchens

Let us consider the setting of a table as an illustrative example (see Figure 15) for the automatic acquisition of new high-level skills. Upon receiving “set the table” the robot retrieves instructions from webpages such as *ehow.com*. These instructions are typically sequences of steps to be executed in order to carry out the activities successfully. The challenges of this execution scenario are: (1) translate the abstract instructions into an executable robot control program, (2) supplement missing information through observations of kitchen activities, (3) transform the action sequence into an activity structure that can be carried out more reliably, efficiently, and flexibly. Instructions typically abstract away from these aspects of activity specification.

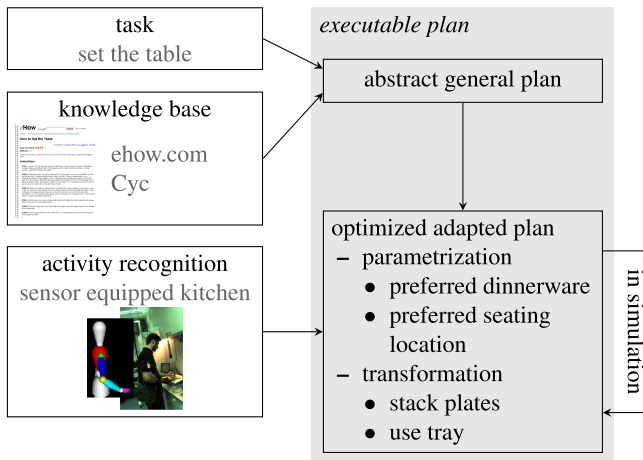


Fig. 15. Learning to set the table

Let us now look at some of the hard challenges in this scenario. First, translating abstract instructions into a working robot control program requires answers to the following research questions: (1) How can the plan libraries of autonomous household robots be specified so generally, reliably, transparently, and modularly that a robot can compose (almost) working plans from abstract instructions? [61] For newly composed sequences of plan steps to work it helps if the individual plan steps are specified as

“universal plans”, that is they achieve – if necessary – all preconditions needed for producing the desired effects. (2) Debugging newly created plans from instructions requires the robot to predict what will happen if it executes the new plan, to identify the flaws of the plan with respect to its desired behavior, and to revise the plan in order to avoid the predicted flaws. (3) Optimizing tasks like table setting also requires the technical cognitive system to observe people setting the table, to infer the structure of the activity and reason about why people do not follow the abstract instructions like a robot but perform the task the way they do. This way the robot would learn that people stack plates when carrying them in order to minimize the distance they have to walk. The robot would then transform its plan analogously and test whether this change of activity structure would result in improved performance.

5 Conclusions

In this paper we have described cognitive technical systems as an application area of Artificial Intelligence technology using the Assistive Kitchen as an example. In this context we have identified various tasks where AI technology can advance the state-of-the-art in cognitive technical systems technology substantially. However, this requires that the respective AI methods scale towards high performance control of robotic systems in the context of sensor-equipped, embedded robot applications.

We have pointed out a number of research areas where we expect breakthrough results into this direction. Most notably these areas include:

- symbolic representations grounded in perception and action,
- first-order probabilistic representations of actions, objects, and situations,
- reasoning about objects and situations in the context of everyday manipulation tasks in human living environments,
- the representation and revision of robot plans for everyday activity.

to name only a few.

It is also necessary that AI researchers closely study the research in neighboring research disciplines. For example, research on manipulation and motion planning [46] has already outperformed AI planning technology in a number of aspects. Also, research in computational motor control [19] can tell us nowadays how prediction, model learning, and probabilistic inference work together to produce smooth and high-performance robot behavior.

In this view, cognitive technical systems are an ideal application area to drive AI research and prove its relevance to the realization of high-performance autonomous robotic systems.

References

1. Buss, M., Beetz, M., Wollherr, D.: CoTeSys — cognition for technical systems. In: Proceedings of the 4th COE Workshop on Human Adaptive Mechatronics (HAM) (2007)
2. Brachman, R.: Systems that know what they’re doing. IEEE Intelligent Systems, 67–71 (November/December 2002)

3. Zäh, M.F., Lau, C., Wiesbeck, M., Ostgathe, M., Vogl, W.: Towards the cognitive factory. In: Proceedings of the 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2007) (2007)
4. Ott, C., Eiberger, O., Friedl, W., Bauml, B., Hillenbrand, U., Borst, C., Albu-Schaffer, A., Brunner, B., Hirschmüller, H., Kielhofer, S., Konietzschke, R., Suppa, M., Wimbock, T., Zacharias, F., Hirzinger, G.: A humanoid two-arm system for dexterous manipulation. In: Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots, IEEE Computer Society Press, Los Alamitos (2006)
5. Ott, C., Eiberger, O., Friedl, W., Bäuml, B., Hillenbrand, U., Borst, C., Albu-Schäffer, A., Brunner, B., Hirschmüller, H., Kielhöfer, S., Konietzschke, R., Suppa, M., Wimbock, T., Zacharias, F., Hirzinger, G.: A humanoid two-arm system for dexterous manipulation. In: IEEE-RAS International Conference on Humanoid Robots, pp. 276–283. IEEE Computer Society Press, Los Alamitos (2006)
6. Zacharias, F., Borst, C., Hirzinger, G.: Bridging the gap between task planning and path planning. In: Proceedings of IROS'06, the IEEE International Conference on Intelligent Robots and Systems, Beijing, China, pp. 4490–4495. IEEE Computer Society Press, Los Alamitos (2006)
7. Ulbrich, H., Buschmann, T., Lohmeier, S.: Development of the humanoid robot LOLA. *Journal of Applied Mechanics and Materials* 5(6), 529–539 (2006)
8. Lohmeier, S., Buschmann, T., Ulbrich, H., Pfeiffer, F.: Modular joint design for a performance enhanced humanoid robot. In: Proceedings of the 2006 IEEE Int. Conf. Rob. Aut (ICRA), Orlando, USA, pp. 88–93. IEEE Computer Society Press, Los Alamitos (2006)
9. Gienger, M., Löffler, K., Pfeiffer, F.: Design and realization of jogging Johnnie. In: CISM Courses and Lectures (2002)
10. Buss, M., Hardt, M., Kiener, J., Sobotka, M., Stelzer, M., von Stryk, O., Wollherr, D.: Towards an autonomous, humanoid, and dynamically walking robot: Modelling, optimal trajectory planning, hardware architecture, and experiments. In: Proceedings of the IEEE/RAS International Conference on Humanoid Robots, Karlsruhe, Germany (2003)
11. Wollherr, D., Buss, M., Hardt, M., von Stryk, O.: Research and development towards an autonomous biped walking robot. In: Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics AIM2003, Kobe, Japan, pp. 968–973 (2003)
12. Sobotka, M., Wollherr, D., Buss, M.: A jacobian method for online modification of precalculated gait trajectories. In: Proceedings of the 6th International Conference on Climbing and Walking Robots, Catania, Italy, pp. 435–442 (2003)
13. Beetz, M., Bandouch, J., Kirsch, A., Maldonado, A., Müller, A., Rusu, R.B.: The assistive kitchen — a demonstration scenario for cognitive technical systems. In: Proceedings of the 4th COE Workshop on Human Adaptive Mechatronics (HAM) (2007)
14. Doherty, P., Granlund, G., Kuchcinski, K., Sandewall, E., Nordberg, K., Skarman, E., Wiklund, J.: The WITAS unmanned aerial vehicle project. In: Proc. of the European Conference on Artificial Intelligence (ECAI 2000) (2000)
15. Muscettola, N., Nayak, P.P., Pell, B., Williams, B.: Remote Agent: To boldly go where no AI system has gone before. *Artificial Intelligence* 103, 5–48 (1998)
16. Thrun, S., Beetz, M., Bennewitz, M., Cremers, A., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., Schulz, D.: Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *International Journal of Robotics Research* (2000)

17. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., Van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P.: Winning the darpa grand challenge. *Journal of Field Robotics*, 2006 (accepted for publication)
18. Bertero, M., Poggio, T., Torre, V.: Ill-posed problems in early vision. *Proceedings of the IEEE* 76(8), 869–889 (1988)
19. Koerding, K.P., Wolpert, D.M.: Bayesian decision theory in sensorimotor control. *Trends in Cognitive Sciences* 10 , 319–326 (2006)
20. Schaal, S., Schweighofer, N.: computational motor control in humans and robots. *Current Opinion in Neurobiology* (6), 675–682 (2005)
21. Chater, N., Tenenbaum, J.B., Yuille, A.: Probabilistic models of cognition: Conceptual foundations. *Trends in Cognitive Sciences* 10(7) (2006)
22. Chater, N., Tenenbaum, J.B., Yuille, A.: Probabilistic models of cognition: where next? *Trends in Cognitive Sciences* 10(7) (2006)
23. Gopnik, A., Glymour, C., Sobel, D., Schulz, L.: A theory of causal learning in children: causal maps and bayes nets. *Psychological review* (2004)
24. Gopnik, A., Glymour, C., Sobel, D., Schulz, L.: Causal learning mechanisms in very young children: two- three-, and four-year-olds infer causal relations from patterns of variation and covariation. *Developmental Psychology* 37 (2001)
25. Skaggs, W., McNaughton, B.: Spatial firing properties of hippocampal ca1 populations in an environment containing two visually identical regions. *Journal of Neuroscience* 18 (1998)
26. Dickmanns, E.: *Dynamic Vision for Perception and Control of Motion*. Springer, Heidelberg (2007)
27. A research roadmap of cognitive vision, Tech. Rep. v5, P Auer et al, *ECVISION* (2005), <http://www.ecvision.org>
28. Thrun, S., Burgard, W., Fox, D.: *Probabilistic Robotics*. MIT Press, Cambridge (2005)
29. Beetz, M., Kirchlechner, B., Lames, M.: Computerized real-time analysis of football games. *IEEE Pervasive Computing* 4(3), 33–39 (2005)
30. Brown, L.E., Rosenbaum, D.A.: *Encyclopedia of Cognitive Science*. In: *Motor Control: Models*, Macmillan, London (2002)
31. Barnard, P., Dayan, P., Redgrave, P.: Foresight cognitive systems project research review: Action. tech. rep
32. Buss, M.: Hybrid control of mechatronic systems. *Systems, Control and Information* 46(3), 129–137 (2002)
33. Reiter, R.: *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge (2001)
34. Sutton, R., Barto, A.: *Reinforcement Learning: an Introduction*. MIT Press, Cambridge (1998)
35. Stulp, F., Beetz, M.: Optimized execution of action chains using learned performance models of abstract actions. In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)* (2005)
36. Morris, R., Hitch, G., Graham, K., Bussey, T.: Foresight cognitive systems project research review: Learning and memory. tech. rep.
37. Peters, J., Mistry, M., Udawadia, F., Cory, R., Nakanishi, J., Schaal, S.: A unifying methodology for the control of robotic systems. In: *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)* (2005)
38. Kawato, M.: Feedback-error-learning neural network for supervised motor learning. In: *Advanced Neural Computers*, pp. 365–472. Elsevier Science Publishers, Amsterdam (1990)

39. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.-H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Grossman, R.L., Ravn, A.P., Rischel, H., Nerode, A. (eds.) *Hybrid Systems*. LNCS, vol. 736, pp. 209–229. Springer, Heidelberg (1993)
40. Lenat, D.B.: Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11) (1995)
41. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. *Scientific American Magazine* (May 2001)
42. Davis, E.: *Representations of Commonsense Knowledge*. Kaufmann Publishers Inc., San Francisco (1990)
43. Mitchell, T.M.: *The Discipline of Machine Learning*. Carnegie Mellon University, White Paper (2006)
44. Beetz, M.: A roadmap for research in robot planning. tech. rep., PLANET: European Network of Excellence in AI Planning (2004), <http://www9.cs.tum.edu/research/tcu/>
45. Cambon, S., Gravot, F., Alami, R.: A robot task planner that merges symbolic and geometric reasoning. In: *16th European Conference on Artificial Intelligence (ECAI'2004)*, pp. 895–899 (2004)
46. Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., Inoue, H.: Motion planning for humanoid robots (2003)
47. Kuffner, J., LaValle, S.: RRT-connect: An efficient approach to single-query path planning. In: *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000)*, San Francisco, CA, April 2000, IEEE Computer Society Press, Los Alamitos (2000)
48. Kemp, C., Edsinger, A., Torres-Jara, E.: Challenges for robot manipulation in human environments. *IEEE Robotics & Automation Magazine* 14, 20–29 (2007)
49. Rusu, R.B., Maldonado, A., Beetz, M., Gerkey, B.: Extending Player/Stage/Gazebo towards cognitive robots acting in ubiquitous sensor-equipped environments. In: *Accepted for the IEEE International Conference on Robotics and Automation (ICRA) Workshop for Network Robot System*, Rome, Italy, April 14, 2007, IEEE Computer Society Press, Los Alamitos (2007)
50. Patterson, D., Fox, D., Kautz, H., Philipose, M.: Fine-grained activity recognition by aggregating abstract object usage. In: *Proceedings of the IEEE International Symposium on Wearable Computers*, Osaka, Japan, October 2005, IEEE Computer Society Press, Los Alamitos (2005)
51. Philipose, M., Fishkin, K., Perkowitz, M., Patterson, D., Fox, D., Kautz, H., Hahnel, D.: Inferring activities from interactions with objects. *Pervasive Computing, IEEE* (2004)
52. Wren, C., Azarbayejani, A., Darrell, T., Pentland, A.: Pfindex: real-time tracking of the human body. Tech. Rep. 353, MIT Media Lab (1996)
53. Smith, J., Fishkin, K., Jiang, B., Mamishev, A., Philipose, M., Rea, A., Roy, S., Sundara-Rajan, K.: Rfid-based techniques for human activity recognition. *Communications of the ACM* (September 2005)
54. Rusu, R.B., Blodow, N., Marton, Z., Soos, A., Beetz, M.: Towards 3d object maps for autonomous household robots. In: *submitted to Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)* (2007)
55. Domingos, P., Richardson, M.: Markov Logic: A Unifying Framework for Statistical Relational Learning. In: *Proceedings of the ICML 2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*, pp. 49–54 (2004)
56. Domingos, P.: What's Missing in AI: The Interface Layer. In: Cohen, P. (ed.) *Artificial Intelligence: The First Hundred Years*, AAAI Press (2006)

57. Hoyningen-Huene, N.v., Kirchlechner, B., Beetz, M.: GrAM: Reasoning with grounded action models by combining knowledge representation and data mining. In: *Towards Affordance-based Robot Control*. LNCS (LNAI), Springer, Heidelberg (to appear, 2007)
58. Bylander, T.: The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2), 165–204 (1994)
59. Fox, M., Long, D.: PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20, 61–124 (2003)
60. Beetz, M.: Plan representation for robotic agents. In: *Proceedings of the Sixth International Conference on AI Planning and Scheduling*, Menlo Park, CA, pp. 223–232. AAAI Press (2002)
61. Firby, R.J., Prokopowicz, P., Swain, M.: Plan representations for picking up trash. In: *Proceedings of the Int. Joint Conf. on Artificial Intelligence* (1995)

Artificial Intelligence Is Engineering Intelligence – Why Should We Care About Natural Intelligence?

Thomas Christaller

Fraunhofer IAIS, Sankt Augustin
thomas.christaller@iais.fraunhofer.de

Artificial Intelligence is about designing and constructing artefacts, normally not about explaining human intelligence. So, why should we care about natural intelligence when talking about AI? There are several important more or less recent findings in brain science as well as ethology which require a deeper rethinking on the AI side. Based on them, the hypothesis in this talk is: The rising complexity of the behaviour system and of personalized social relationships was one of the major reasons for developing intelligence – contrary to the huge resource consumption that intelligence costs an individual. The most important result of this development was the capability of forecasting the behaviour of conspecifics for survival in a complex social environment. This capability was also useful for other purposes, including forecasting behaviour of individuals of other species and nature itself.

A second focus in the talk will be language and the hypothesized reasons or causes for its evolution and its primary usages. This will lead to the concept of imitation and its neural basis. Some plausible speculations will be given, why all these findings fit into a relatively consistent picture of natural intelligence. The conclusion will be some examples on how these findings can inspire AI research and the construction of AI systems.

Applying Machine Learning Techniques for Detection of Malicious Code in Network Traffic

Yuval Elovici, Asaf Shabtai, Robert Moskovitch, Gil Tahan, and Chanan Glezer

Deutsche Telekom Laboratories at Ben-Gurion University,
Be'er Sheva, 84105 Israel

{elovici, shabtaia, robertmo, gilta, chanan}@bgu.ac.il

Abstract. The Early Detection, Alert and Response (eDare) system is aimed at purifying Web traffic propagating via the premises of Network Service Providers (NSP) from malicious code. To achieve this goal, the system employs powerful network traffic scanners capable of cleaning traffic from known malicious code. The remaining traffic is monitored and Machine Learning (ML) algorithms are invoked in an attempt to pinpoint unknown malicious code exhibiting suspicious morphological patterns. Decision trees, Neural Networks and Bayesian Networks are used for static code analysis in order to determine whether a suspicious executable file actually inhabits malicious code. These algorithms are being evaluated and preliminary results are encouraging.

Keywords: Malicious Code, Machine Learning, Network Service Provider (NSP), Feature Selection.

1 Introduction

In a recent online safety survey conducted by America Online and the National Cyber Security Alliance (NCSA), 81% of the respondents were found to be lacking recently-updated anti-virus software, a properly-configured firewall, and/or spyware protection. Nevertheless, 74% of the respondents use the Internet for “sensitive” transactions from their home computers, including among others banking, stock trading, and reviewing personal medical information [1-4]. One way to prevent users from being infected by malicious code is to purify the network traffic by the Network Service Provider (NSP). The Early Detection, Alert and Response (eDare) system was designed to accomplish this task [5]. The proposed system employs powerful network traffic scanners to remove known malicious code from network traffic. The remaining suspicious traffic is monitored and ML techniques, such as classification algorithms, are invoked for identifying unknown malicious code. Each ML technique is implemented as a modular plug-in appended to the core system. *Decision Trees*, *Bayesian Networks* and *Artificial Neural Network* plug-ins are all used in this study for static code analysis in order to determine whether a file contains malicious code.

Detection of malicious employing content-based features operationalized by ML learning algorithms is not a new concept. In [6], ML techniques were applied for detection of unknown malicious code based on binary code content. They used three

different feature extraction methods: Program Header, Strings features and Byte Sequences features, on which they applied four classifiers: Signature-based method (anti-virus), Ripper – a rule based learner, Naïve Bayes and Multi Naïve Bayes. Their study showed that all ML methods had better accuracy than the signature-based algorithm. Other studies used various features extracted from the suspected binary code, feature selection methods and ML techniques in order to detect unknown malicious code [6-8].

The major advantages of eDare when compared to the aforementioned approaches are: first, its flexibility in using plug-ins (without recompiling the system when adding or updating plug-ins); and second, its ability to weigh and integrate the results of the multiple plug-ins into one final detection decision with superior accuracy than each of the individual plug-ins. In this paper we present a short review on eDare and its plug-ins and will show some preliminary evaluation results that demonstrate the system ability to detect unknown malicious code.

The rest of the paper is structured as follows: Section 2 describes the architecture of the eDare system; section 3 presents the ML algorithms employed for detection of unknown malicious code; section 4 illustrates the preliminary results emanating from an empirical evaluation of eDare. Finally, section 6 concludes the paper with a summary and an overview of future research avenues.

2 The eDare Framework

This research adopts a distributed, network-based approach, in which NSPs constantly monitor traffic flowing via their infrastructure in an attempt to detect and remove malicious code (hereafter: eThreats). eDare is designed to provide maximum automation in the cycle of intercepting, analyzing, alerting and overcoming instances of eThreats. The system aims to provide very low false positive by integrating multiple sources of information and multiple eThreat detection techniques. Finally, the system easily accommodates external plug-ins, expert consultation and risk assessment.

The conceptual architecture of eDare is depicted in Figure 1. The group of eThreats faced by eDare can be classified into the following two types:

- **Known** eThreats for which eDare has already generated a distinct signature and are intercepted by the **Known eThreat handling Module**; and
- **Unknown (New)** eThreats which eDare is yet to encounter and classify, and for which eDare needs to generate a distinct signature via the **New eThreat handling Module**. In order to enable flexibility in employing various new eThreat detection algorithms, modular plug-ins are used. All plug-ins have a similar interface, that is, a suspicious file to be examined as input, and a threat rank as output. Examples of plug-ins incorporated by eDare include: a statistical plug-in, static analysis (morphological) plug-in, ML plug-in and a collaborative plug-in. Finally, a **Risk-weighing** function collects all ranks from the relevant plug-ins and calculates an integrated rank according to some weighing scheme. In case the final rank of a file is above a devisable threshold, the file will be transferred to the **Signature builder Module** which will construct a unique signature.

When encountering new eThreats or suspicious behaviors, the response time and effectiveness of the system is substantially expedited by sharing observations and warnings between users and the system via a **Collaborative Module**, which further propagates relevant alerts across the protected network.

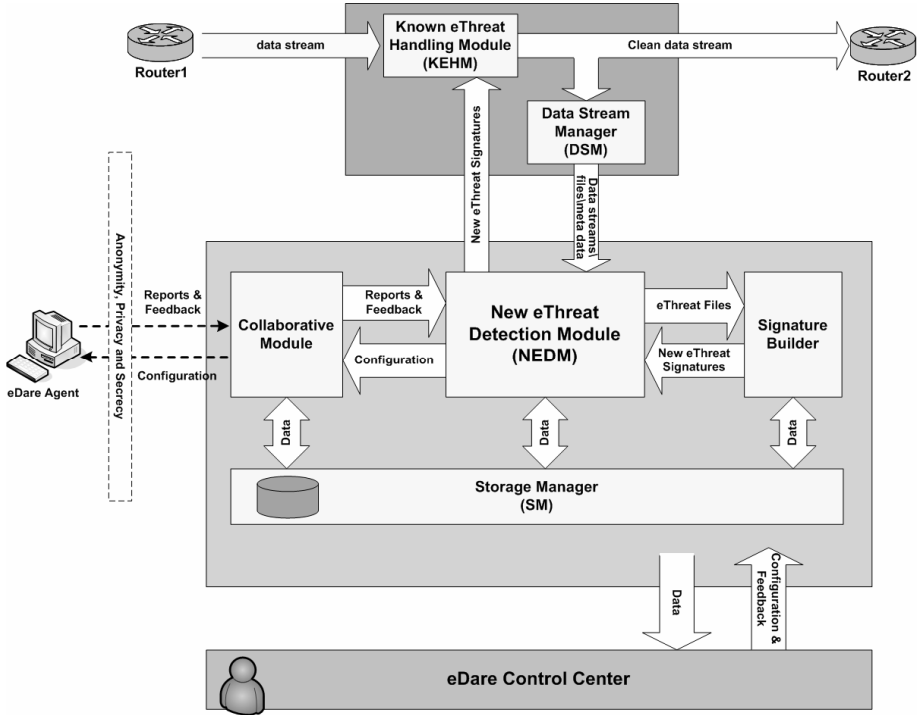


Fig. 1. Architecture of the eDare System

The role of the **Data Stream Manager Module** is to obtain a "clean" network data stream from the Known eThreat Handler Module. It then extracts and assembles files from the continuous network data stream and forwards them to the New eThreat Detection Module. The overall configuration and operation of the system is controlled by a human administrator via a console termed **eDare Control Center**.

3 Detecting Malicious Code Using ML Techniques

In this paper we present how eDare performed when using five plug-ins. The plug-ins were based on three ML techniques (described in subsections 3.1-3.3) and on two types of inputs (described in subsection 3.4).

3.1 Decision Trees

Decision tree learners [9] are a well-established family of ML algorithms. Classifiers are represented as trees whose internal nodes are tests on individual features and

leaves are classification decisions. Typically, a greedy heuristic search method is used to find a compact decision tree that correctly classifies the training data. The decision tree is induced from the dataset by splitting the variables based on the *expected information gain*. Modern implementations include pruning which avoids over-fitting. In this study we evaluated J48, the Weka [10] version of the commonly used C4.5 algorithm [9]. An important characteristic of Decision Trees is the explicit form of their knowledge which can be easily represented as a set of rules.

3.2 Bayesian Networks

Bayesian networks are a form of probabilistic graphical modeling [11]. Specifically, a Bayesian network is a directed acyclic graph of nodes with variables and arcs representing dependencies among the variables. Bayesian networks are based on Bayes' theorem, and they are known for their ability to represent conditional probabilities which capture the internal relationships between the studied variables. A Bayesian network can thus be considered a mechanism for automatically constructing extensions of Bayes' theorem to more complex problems. We evaluated the Bayesian Network standard version which comes with WEKA [10].

3.3 Artificial Neural Network

An Artificial Neural Network (ANN) [12] is an information processing paradigm inspired by information processing mechanisms of biological nervous systems (i.e., the brain). The key element of this paradigm is the structure of the information processing system modeled as a network comprising many highly interconnected Processing Elements (PEs) (also termed Neurons). Neurons work together in order to approximate a specific transformation function. An ANN is configured for a specific application, such as pattern recognition or data classification, through a *learning process* during which the weights of the inputs in each neuron are updated. The weights are updated by a *training algorithm*, such as back-propagation, according to examples the network receives in order to reduce the value of an *error function*. The power and usefulness of ANN have been demonstrated in numerous applications including speech-synthesis, medicine, finance and many other pattern-recognition problems. Neural models show more promise in achieving human-like performance when dealing with unstructured application domains where traditional AI knowledge representation techniques (i.e., rule/frame bases) are cumbersome. All ANN manipulations in this study have been performed within the MATLAB(r) environment using the Neural Network Toolbox [13].

3.4 Feature Selection

In order to determine whether a suspected file is malicious or not, using ML techniques, we extracted two types of static features: n-grams and Win32 executables Portable Executable header. We implemented a tool that extracts 5-grams from the binary representation of a file. Out of all 5-grams that were extracted we chose the top 5,500 which are the most frequent in the file set. Next, for each 5-gram we calculated its *tf-idf* score. The *tf-idf* score is mainly used as a text retrieval method. *tf_i* is the term frequency (i.e., n-gram) in the file *i* and *idf* is the log value of the number of files in

whole repository, divided by the number of files that include that term. We used Fisher’s Score [14] for feature selection in order to choose the top 300 5-grams out of the 5,500 5-grams, to be used for the evaluation. Next, PE features were extracted from Win32 executables using the PE Feature Extractor tool which parses an EXE/DLL file according to PE Format rules. We statically extracted different PE format features representing information contained within each Win32 PE binary (EXE or DLL). Examples of the information extracted are: creation\modification time, machine type, file size, linker version, section alignment, code size, imported DLLs, exported functions, etc.

4 Preliminary Evaluation Results

eDare was deployed and tested in a network-security lab with distinct “clean” and “infected” environments. We collected a repository of 7694 malicious files representing a variety of eThreats and 22736 benign files. This collection was used to train and test the effectiveness of each of eDare’s plug-ins and their effect on the overall system’s performance. After conducting a rigorous research, we chose to use the following plug-ins for classification of suspected files:

- ANN (5grams; top300; Fisher): Artificial Neural Network (ANN) classifier trained on the top 300 5-grams, selected using Fisher score.
- BN (5grams; top300; Fisher): Bayesian Network (BN) classifier trained on the top 300 5-grams, selected using Fisher score
- DT (5grams; top300; Fisher): Decision Tree classifier trained on the top 300 5-grams, selected using Fisher score.
- BN (PE): Bayesian Network (BN) classifier trained on the PE (Portable Executable) features.
- DT (PE): Decision Tree (DT) classifier trained on the PE (Portable Executable) features.

A simple voting scheme [15] was applied as an ensemble algorithm in order to generate a final recommendation from the classifications of the individual plug-ins. In order to compare the various plug-ins we used the following common evaluation measures: *True Positive Rate (TPR)*, *False Positive Rate (FPR)* and *Accuracy*. TPR is the proportion of malware files classified correctly; FPR is the proportion of benign files misclassified; and Accuracy is the number of correctly classified instances (either malware or benign), divided by the entire number of instances. We also used *Receiver Operating Characteristics (ROC)* curves. A ROC curve is a graphical representation of the trade-off between the true positive (TPR) and false positive rates (FPR) for every possible cut-off.

Each plug-in was trained using 30% of the dataset and tested using the rest of the dataset (70%). Table 1 describes the evaluation results when the detection threshold was set to 0.5, where FPR stands for the false positive rate, TPR stands for the true positive rate and the accuracy stands for the detection accuracy. It is easy to see in Table 1 that, among individual plug-ins, DT (PE) had the highest accuracy with lowest false positive. The results also show that simple voting improves the overall accuracy, reduces the false positive rate and improves the true positive rate compared to all of the individual plug-ins except the BN (PE).

The graph in Fig. 2 presents the ROCs of all individual plug-ins and their combination using a standard weighing mechanism. The weighing mechanism combines the individual plug-in decision into a final single decision based on the simple voting algorithm. It is easy to see in Fig. 2 that the Risk Weighting line provides the best true positive rate with a minimum false positive rate and thus should be preferred over each individual ML technique. This observation is also captured quantitatively by the Area Under Curve (AUC) measure which reaches its peak (0.983) in the case of the weighted line.

Table 1. Evaluation results for detection threshold = 0.5

| | FPR | TPR | Accuracy |
|------------------------------|--------------|--------------|-----------------|
| ANN (5grams; top300; Fisher) | 0.038 | 0.893 | 0.945 |
| BN (5grams; top300; Fisher) | 0.206 | 0.885 | 0.816 |
| BN (PE) | 0.058 | 0.933 | 0.94 |
| DT (5grams; top300; Fisher) | 0.039 | 0.87 | 0.938 |
| DT (PE) | 0.035 | 0.925 | 0.955 |
| Risk Weighing(Voting) | 0.032 | 0.928 | 0.958 |

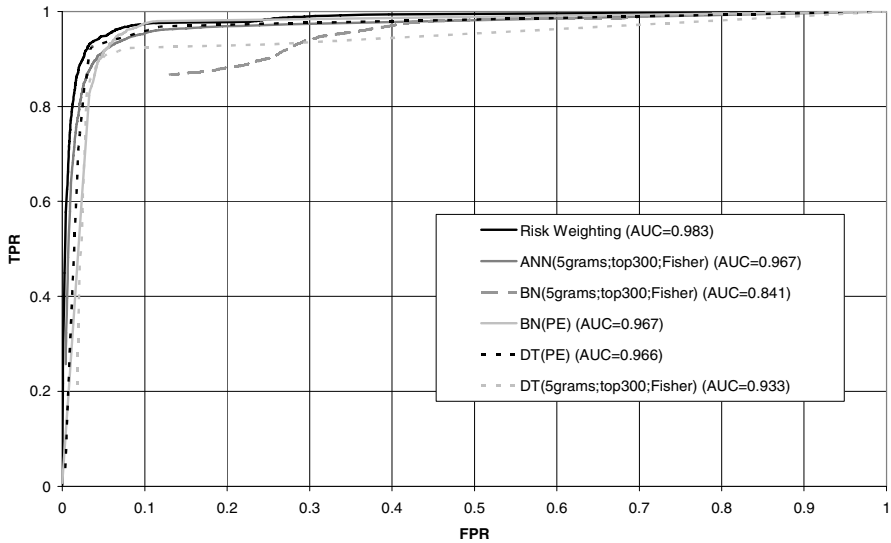


Fig. 2. Detection Plug-ins ROC

5 Summary

In this paper we presented a new system aimed at protecting users from various eThreats. eDare protects end-users by purifying NSPs' traffic from known eThreats without a mandatory requirement to install any software on end-users' computers. Sanitation of Web traffic from malicious code is performed by dedicated commercial

cleaning devices capable of removing threats based on prespecified signatures that eDare generates.

eDare can also detect unknown threats by employing, among others, ML algorithms as plug-ins in a flexible, open manner. Once a new threat is detected, eDare generates a signature and update the above cleaning devices deployed over NSPs' network.

Our empirical experimental findings suggest that a new eThreat risk-weighting scheme, weighing several ML algorithms (i.e., ANN, BN, DT) in various configurations (n-grams, PE) outperforms each of individual algorithms in terms of prediction accuracy. This important finding suggests that eDare's new eThreat detection module should be designed in a flexible, plug-in mode accommodating a heterogeneous collection of ML algorithms in order to facilitate better prediction accuracy of new eThreats.

References

1. NCSA Study, http://www.staysafeonline.info/pdf/safety_study_2005.pdf
2. Symantec Internet Security Threat Report (January-June 2004), www.symantec.com
3. The Danger of Spyware, Symantec Security Response (June 2003), <http://www.symantec.com>
4. Symantec 2006 Security Report, http://www.symantec.com/specprog/threatreport/entwhitepaper_symantec_internet_security_threat_report_x_09_2006.en-us.pdf
5. Tahan, G., Glezer, C., Elovici, Y.: eDare- Early Detection Alert and Response to Electronic Threats, Working Paper, Deutsche Telekom Labs at Ben Gurion University
6. Schultz, M., Eskin, E., Zadok, E., Stolfo, S.: Data Mining Methods for Detection of New Malicious Executables. In: Proc. of the IEEE Symposium on Security and Privacy, pp. 178–184 (2001)
7. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidan, R.: N-gram based Detection of New Malicious Code. In: Proc. of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04) (2004)
8. Kolter, J.Z., Maloof, M.A.: Learning to detect malicious executables in the wild. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470–478. ACM Press, New York, NY (2004)
9. Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
10. Weka software, <http://www.cs.waikato.ac.nz/ml/weka/>
11. Pearl, J.: Fusion, propagation, and structuring in belief networks. *Artificial Intelligence* 29(3), 241–288 (1986)
12. Bishop, C.: *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford (1995)
13. Demuth, H., Beale, M.: *Neural Network toolbox for use with Matlab*. The Mathworks Inc., Natick, MA (1998)
14. Golub, T., Slonim, D., Tamaya, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., Lander, E.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286, 531–537 (1999)
15. Bauer, E., Kohavi, R.: An empirical comparison of voting classification Algorithms. Bagging, Boosting, and Variants. *Machine Learning* 35, 1–38 (1999)

Location-Based Activity Recognition

Dieter Fox

University of Washington, Seattle, USA
fox@cs.washington.edu

Knowledge of a person's location provides important context information for many applications, ranging from services such as E911 to personal guidance systems that help cognitively-impaired individuals move safely through their community. Location information is also extremely helpful for estimating a person's high-level activities. In this talk we show how Bayesian filtering and conditional random fields can be applied to estimate the location and activity of a person using sensors such as GPS or WiFi. The techniques track a person on graph structures that represent a street map or a skeleton of the free space in a building. We also show how to learn a user's significant places and daily movements through the community. Our models use multiple levels of abstraction so as to bridge the gap between raw GPS measurements and high level information such as a user's mode of transportation, her current goal, and her significant places (e.g. home or work place). Finally, we will discuss recent work on using a multi-sensor board so as to better estimate a person's activities.

Pinpointing in the Description Logic \mathcal{EL}^+

Franz Baader¹, Rafael Peñaloza^{2,*}, and Boontawee Suntisrivaraporn^{1,**}

¹ Theoretical Computer Science, TU Dresden, Germany

{baader,meng}@tcs.inf.tu-dresden.de

² Intelligent Systems, University of Leipzig, Germany

penaloza@informatik.uni-leipzig.de

Abstract. Axiom pinpointing has been introduced in description logics (DLs) to help the user understand the reasons why consequences hold by computing minimal subsets of the knowledge base that have the consequence in question. Until now, the pinpointing approach has only been applied to the DL \mathcal{ALC} and some of its extensions. This paper considers axiom pinpointing in the less expressive DL \mathcal{EL}^+ , for which subsumption can be decided in polynomial time. More precisely, we consider an extension of the pinpointing problem where the knowledge base is divided into a *static* part, which is always present, and a *refutable* part, of which subsets are taken. We describe an extension of the subsumption algorithm for \mathcal{EL}^+ that can be used to compute all minimal subsets of (the refutable part of) a given TBox that imply a certain subsumption relationship. The worst-case complexity of this algorithm turns out to be exponential. This is not surprising since we can show that a given TBox may have exponentially many such minimal subsets. However, we can also show that the problem is not even output polynomial, i.e., unless $P=NP$, there cannot be an algorithm computing all such minimal sets that is polynomial in the size of its input *and output*. In addition, we show that finding out whether there is such a minimal subset within a given cardinality bound is an NP-complete problem. In contrast to these negative results, we also show that one such minimal subset can be computed in polynomial time. Finally, we provide some encouraging experimental results regarding the performance of a practical algorithm that computes one (small, but not necessarily minimal) subset that has a given subsumption relation as consequence.

1 Introduction

Description logics (DLs) [2] are a successful family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, databases, and bio-medical ontologies, but their most notable success so far is the adoption of the DL-based language OWL [15] as standard

* Funded by the German Research Foundation (DFG) under grant GRK 446.

** Funded by the German Research Foundation (DFG) under grant BA 1122/11-1.

ontology language for the semantic web. For a developer or user of a DL-based ontology, it is often quite hard to understand why a certain consequence holds, and even harder to decide how to change the ontology in case the consequence is unwanted. For example, in the current version of the medical ontology SNOMED [24], the concept *Amputation-of-Finger* is unintendedly classified as a subconcept of *Amputation-of-Arm*. Finding the axioms that are responsible for this among the more than 350,000 terminological axioms of SNOMED without support by an automated reasoning tool is not easy.

As a first step towards providing such support, Schlobach and Cornet [22] describe an algorithm for computing all the *minimal subsets* of a given knowledge base that *have* a given *consequence*. In the following, we call such a set a *minimal axiom set (MinA)*. It helps the user to comprehend why a certain consequence holds. The knowledge bases considered in [22] are so-called unfoldable \mathcal{ALC} -terminologies, and the unwanted consequences are the unsatisfiability of concepts. The algorithm is an extension of the known tableau-based satisfiability algorithm for \mathcal{ALC} [23], where labels keep track of which axioms are responsible for an assertion to be generated during the run of the algorithm. The authors also coin the name “axiom pinpointing” for the task of computing these minimal subsets.

The problem of computing MinAs of a DL knowledge base was actually considered earlier in the context of extending DLs by default rules. In [3], Baader and Hollunder solve this problem by introducing a labeled extension of the tableau-based consistency algorithm for \mathcal{ALC} -ABoxes [14], which is very similar to the one described later in [22]. The main difference is that the algorithm described in [3] does not directly compute minimal subsets that have a consequence, but rather a monotone Boolean formula whose variables correspond to the axioms of the knowledge bases and whose minimal satisfying valuations correspond to the MinAs. Another difference between [3] and [22] is that in the former paper the ABox is divided into a static and a refutable part, where the elements of the static part are assumed to be always present, and subsets are built only of the refutable part of the ABox.

The approach of Schlobach and Cornet [22] was extended by Parsia et al. [20] to more expressive DLs, and the one of Baader and Hollunder [3] was extended by Meyer et al. [19] to the case of \mathcal{ALC} -terminologies with general concept inclusions (GCIs), which are no longer unfoldable. Axiom pinpointing has also been considered in other research areas, though usually not under this name. For example, in the SAT community, people have considered the problem of computing minimally unsatisfiable (and maximally satisfiable) subsets of a set of propositional formulae. The approaches for computing these sets developed there include special purpose algorithms that call a SAT solver as a black box [18, 7], but also algorithms that extend a resolution-based SAT solver directly [9, 26].

Whereas the previous work on pinpointing in DLs considered fairly expressive DLs that contain at least \mathcal{ALC} , this work is concerned with pinpointing in the inexpressive DL \mathcal{EL}^+ , which allows for conjunction, existential restrictions, and

complex role inclusion axioms. There are several good reasons for considering \mathcal{EL}^+ . First, several bio-medical ontologies such as SNOMED [24], the Gene Ontology [25], and large parts of Galen [21] can be expressed in \mathcal{EL}^+ . In particular, both SNOMED and Galen require role inclusion axioms. Second, reasoning in \mathcal{EL}^+ and some of its extensions remains polynomial even in the presence of GCI [1], which are required by Galen. Third, \mathcal{EL}^+ is the DL currently handled by our reasoner CEL [5, 4], which behaves quite well on very large ontologies such as SNOMED.

Although the polynomial-time subsumption algorithm for \mathcal{EL}^+ described in [1, 5] is not tableau-based, the ideas for extending tableau-based algorithms to pinpointing algorithms employed in [3, 22] can also be applied to this algorithm. However, we will see that the normalization phase employed by this algorithm introduces an additional problem. We will also consider the complexity of pinpointing in \mathcal{EL}^+ . In contrast to the case of \mathcal{ALC} , where the subsumption problem is already quite complex (PSPACE-complete), subsumption in \mathcal{EL}^+ is polynomial, which makes it easier to analyze in how far pinpointing is a source of additional complexity. Not surprisingly, it turns out that there may be exponentially many MinAs, which shows that an algorithm for computing all MinAs needs exponential time in the size of the input TBox. Even worse, we can show that it is not even possible to obtain an algorithm that is polynomial in the size of the input *and the output* (unless P=NP). In addition, even testing whether there is a MinA of cardinality $\leq n$ for a given natural number n is an NP-complete problem. On the positive side, one MinA can always be computed in polynomial time. Finally, we will provide some experimental results regarding the performance of a practical algorithm that computes one (not necessarily minimal) set that has a given consequence.

2 The Description Logic \mathcal{EL}^+

In DLs, *concept descriptions* are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. \mathcal{EL}^+ concept descriptions are formed using the three constructors shown in the upper part of Table 1. An \mathcal{EL}^+ *ontology* or *TBox* is a finite set of *general concept inclusion (GCI)* and *role inclusion (RI)* axioms, whose syntax is shown in the lower part of Table 1. The sublanguage of \mathcal{EL}^+ that does not allow for RIs is called \mathcal{EL} . We will also use the name \mathcal{HL} for the sublanguage of \mathcal{EL} that does not allow for existential restrictions. This name is motivated by the fact that GCIs involving \mathcal{HL} concepts are basically propositional Horn clauses.

The semantics of \mathcal{EL}^+ is defined in terms of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the domain $\Delta^{\mathcal{I}}$ is a non-empty set of individuals, and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the semantics column of Table 1. An interpretation \mathcal{I} is a *model* of a TBox \mathcal{T} if, for each inclusion axiom in \mathcal{T} , the conditions given in the semantics column of Table 1 are satisfied.

Table 1. Syntax and semantics of \mathcal{EL}^+

| Name | Syntax | Semantics |
|--------------------|---|--|
| top | \top | $\Delta^{\mathcal{I}}$ |
| conjunction | $C \sqcap D$ | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| exists restriction | $\exists r.C$ | $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ |
| GCI | $C \sqsubseteq D$ | $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ |
| RI | $r_1 \circ \dots \circ r_n \sqsubseteq s$ | $r_1^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ |

Since there is no constructor in \mathcal{EL}^+ that can cause logical inconsistencies, satisfiability of concepts or consistency of the TBox are not interesting inference problems. The main inference problem for \mathcal{EL}^+ is the subsumption problem:

Definition 1 (concept subsumption). *Given two \mathcal{EL}^+ concept descriptions C, D and an \mathcal{EL}^+ TBox \mathcal{T} , C is subsumed by D w.r.t. \mathcal{T} (written $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model \mathcal{I} of \mathcal{T} .*

In the following, we will restrict the attention to subsumption between concept names. This is justified by the fact that subsumption between concept descriptions can be reduced to subsumption between concept names: we have $C \sqsubseteq_{\mathcal{T}} D$ iff $A \sqsubseteq_{\mathcal{T} \cup \{A \sqsubseteq C, D \sqsubseteq B\}} B$ where A, B are new concept names not occurring in C, D and \mathcal{T} .

In order to describe our pinpointing algorithm for subsumption in \mathcal{EL}^+ , we must briefly recall the known polynomial-time subsumption algorithm for \mathcal{EL}^+ [1, 5, 4]. First, this algorithm transforms a given TBox into a *normal form* where all GCIs have one of the following forms: $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B$, $A \sqsubseteq \exists r.B$, $\exists r.A \sqsubseteq B$, and where all RIs are of the form $r \sqsubseteq s$ or $r \circ r' \sqsubseteq s$, where $r, r', s \in \mathbf{N}_R$, $n \geq 1$, and A, A_1, \dots, A_n, B are elements of \mathbf{N}_C^{\top} , i.e. concepts names or the top concept \top . This transformation can be achieved in linear time using simple transformation rules, which basically break down complex GCIs into simpler ones (see [1] for details).

Given a TBox \mathcal{T} in normal form over the concept names \mathbf{N}_C and role names \mathbf{N}_R , the *subsumption algorithm for \mathcal{EL}^+* employs completion rules to extend an initial set of assertions until no more assertions can be added. Assertions are of the form (A, B) or (A, r, B) where $A, B \in \mathbf{N}_C^{\top}$, and $r \in \mathbf{N}_R$. Intuitively, the assertion (A, B) expresses that $A \sqsubseteq_{\mathcal{T}} B$ holds and (A, r, B) expresses that $A \sqsubseteq_{\mathcal{T}} \exists r.B$ holds. The algorithm starts with a set of assertions \mathcal{A} that contains (A, \top) and (A, A) for every concept name A , and then uses the rules shown in Fig. 1 to extend \mathcal{A} . Note that such a rule is only applied if it really extends \mathcal{A} , i.e., if the assertion added by the rule is not yet contained in \mathcal{A} . The following theorem, which is shown in [1], summarizes the important properties of this algorithm.

¹ A polynomial-time subsumption algorithm for \mathcal{EL} with GCIs was first presented in [8], and subsumption in \mathcal{HL} with GCIs is basically the implication problem for propositional Horn clauses, which is known to be solvable in linear time [10].

If $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B \in \mathcal{T}$ **and** $\{(X, A_1), \dots, (X, A_n)\} \subseteq \mathcal{A}$ **then** add (X, B) to \mathcal{A} .
If $A \sqsubseteq \exists r.B \in \mathcal{T}$ **and** $(X, A) \in \mathcal{A}$ **then** add (X, r, B) to \mathcal{A} .
If $\exists r.A \sqsubseteq B \in \mathcal{T}$ **and** $\{(X, r, Y), (Y, A)\} \subseteq \mathcal{A}$ **then** add (X, B) to \mathcal{A} .
If $r \sqsubseteq s \in \mathcal{T}$ **and** $(X, r, Y) \in \mathcal{A}$ **then** add (X, s, Y) to \mathcal{A} .
If $r \circ r' \sqsubseteq s \in \mathcal{T}$ **and** $\{(X, r, Y), (Y, r', Z)\} \subseteq \mathcal{A}$ **then** add (X, s, Z) to \mathcal{A} .

Fig. 1. Completion rules for subsumption in \mathcal{EL}^+

Theorem 1. *Given an \mathcal{EL}^+ ontology \mathcal{T} in normal form, the subsumption algorithm terminates in time polynomial in the size of \mathcal{T} . After termination, the resulting set of assertions \mathcal{A} satisfies $A \sqsubseteq_{\mathcal{T}} B$ iff $(A, B) \in \mathcal{A}$, for all concept names A, B occurring in \mathcal{T} .*

3 A Pinpointing Algorithm for \mathcal{EL}^+

In many applications, it makes sense to distinguish two kinds of axioms in an ontology: trusted ones whose correctness is no longer doubted, and refutable ones for which the designer or user of the ontology is not yet sure whether they are correct. For example, if an already well-established ontology is extended, one might view the newly added GCIs as refutable, but trust the GCIs of the existing ontology. From now on, we assume that TBoxes are of the form $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$, i.e., they are a disjoint union of a *static* TBox \mathcal{T}_s (whose axioms are irrefutable) and a *refutable* TBox \mathcal{T}_r .

Definition 2 (MinA). *Let $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ be an \mathcal{EL}^+ TBox and A, B concept names occurring in it such that $A \sqsubseteq_{\mathcal{T}} B$. Then, a minimal axiom set (MinA) for \mathcal{T} w.r.t. $A \sqsubseteq B$ is a subset \mathcal{S} of \mathcal{T}_r such that $A \sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}} B$, but $A \not\sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}'} B$ for all strict subsets $\mathcal{S}' \subset \mathcal{S}$.*

If $\mathcal{T}_s = \emptyset$, then all axioms are assumed to be refutable. This is the case considered, e.g., in [22]. As an example, consider the TBox $\mathcal{T} = (\emptyset \uplus \mathcal{T}_r)$ consisting of the (refutable) GCIs

$$\text{ax}_1: A \sqsubseteq \exists r.A, \quad \text{ax}_2: A \sqsubseteq Y, \quad \text{ax}_3: \exists r.Y \sqsubseteq B, \quad \text{ax}_4: Y \sqsubseteq B. \quad (1)$$

We have $A \sqsubseteq_{\mathcal{T}} B$, and it is easy to see that $\{\text{ax}_2, \text{ax}_4\}$ and $\{\text{ax}_1, \text{ax}_2, \text{ax}_3\}$ are the MinAs for \mathcal{T} w.r.t. $A \sqsubseteq B$.

In the following, we show how the polynomial-time subsumption algorithm presented in the previous section can be modified to a pinpointing algorithm. However, instead of computing the MinAs directly, we follow the approach introduced in [3] that computes a monotone Boolean formula from which the MinAs can be derived. To define this formula, which we will call pinpointing formula in the following, we assume that every refutable axiom $t \in \mathcal{T}_r$ is labeled with a unique propositional variable, $\text{lab}(t)$; axioms $t \in \mathcal{T}_s$ are labeled with $\text{lab}(t) := \mathbf{t}$

(for truth). Let $lab(\mathcal{T})$ be the set of all propositional variables labeling axioms in \mathcal{T}_r . A *monotone Boolean formula* over $lab(\mathcal{T})$ is a Boolean formula using (some of) the variables in $lab(\mathcal{T})$ and only the binary connectives conjunction and disjunction and the nullary connective \mathbf{t} (for truth). As usual, we identify a propositional *valuation* with the set of propositional variables made true by this valuation. For a valuation $\mathcal{V} \subseteq lab(\mathcal{T})$, let $\mathcal{T}_{\mathcal{V}} := \{t \in \mathcal{T}_r \mid lab(t) \in \mathcal{V}\}$.

Definition 3 (pinpointing formula). *Given an \mathcal{EL}^+ TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ and concept names A, B occurring in it, the monotone Boolean formula ϕ over $lab(\mathcal{T})$ is a pinpointing formula for \mathcal{T} w.r.t. $A \sqsubseteq B$ if the following holds for every valuation $\mathcal{V} \subseteq lab(\mathcal{T})$: $A \sqsubseteq_{\mathcal{T}_s \cup \mathcal{T}_{\mathcal{V}}} B$ iff \mathcal{V} satisfies ϕ .*

In our example, we can take $lab(\mathcal{T}) = \{ax_1, \dots, ax_4\}$ as set of propositional variables. It is easy to see that $ax_2 \wedge (ax_4 \vee (ax_1 \wedge ax_3))$ is a pinpointing formula for \mathcal{T} w.r.t. $A \sqsubseteq B$.

If we order valuations by set inclusion, then we obviously have the following relation between MinAs and minimal satisfying valuations of the pinpointing formula.

Proposition 1. *Let ϕ be a pinpointing formula for \mathcal{T} w.r.t. $A \sqsubseteq B$. Then*

$$\{\mathcal{T}_{\mathcal{V}} \mid \mathcal{V} \text{ is a minimal valuation satisfying } \phi\}$$

is the set of all MinAs for \mathcal{T} w.r.t. $A \sqsubseteq B$.

This shows that it is enough to design an algorithm for computing a pinpointing formula to obtain all MinAs. For example, one possibility is to bring ϕ into disjunctive normal form and then remove disjuncts implying other disjuncts. Note that this may cause an exponential blowup, which means that, in some cases, the pinpointing formula provides us with a compact representation of the set of all MinAs. Also note that this blowup is not really in the size of the pinpointing formula but rather in the number of variables. Thus, if the size of the pinpointing formula is already exponential in the size of the TBox \mathcal{T} (which may well happen), computing all MinAs from it is still “only” exponential in the size of \mathcal{T} . More about the complexity of computing all MinAs from a given pinpointing formula can be found in Section 4.

For the moment, let us assume that the TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ is already in normal form. Recall that the axioms in \mathcal{T}_r have a unique propositional variable as label, whereas the axioms in \mathcal{T}_s have label \mathbf{t} . In the *pinpointing extension* of the subsumption algorithm for \mathcal{EL}^+ , assertions a are labeled with monotone Boolean formulae $lab(a)$. The initial assertions (A, \top) and (A, A) receive label \mathbf{t} . The definition of rule application is modified as follows. Assume that the preconditions of a rule from Fig. 1 are satisfied for the set of assertions \mathcal{A} w.r.t. the TBox \mathcal{T} . Let ϕ be the conjunction of the labels of the GCIs from \mathcal{T} and the assertions from \mathcal{A} occurring in the precondition. If the assertion in the consequence of the rule does not yet belong to \mathcal{A} , then it is added with label ϕ . If the assertion is already there with label ψ , then its label is changed to $\psi \vee \phi$ if

this formula is not equivalent to ψ ; otherwise (i.e., if ϕ implies ψ) the rule is not applied.

It is easy to see that this modified algorithm always terminates, though not necessarily in polynomial time. In fact, there are polynomially many assertions that can be added to \mathcal{A} . If the label of an assertion is changed, then the new label is a more general monotone Boolean formula, i.e., it has more models than the original label. Since there are only exponentially many models, the label of a given assertion can be changed only exponentially often. Accordingly, the size of the label of an assertion can grow only exponentially. Equivalence of monotone Boolean formulae is an NP-complete problem. However, given formulae over n propositional variables whose size is exponential in n , equivalence can be tested in time exponential in n . Thus, there are at most exponentially many rule applications, each of which takes at most exponential time. This yields an exponential time bound for the execution of the pinpointing algorithm.

Regarding correctness of the pinpointing algorithm, it is easy to see that the set of assertions \mathcal{A} obtained after termination is identical to the one obtained by the unmodified algorithm. In addition, we can show that, for all assertions $(A, B) \in \mathcal{A}$, the formula $lab((A, B))$ is a pinpointing formula for \mathcal{T} w.r.t $A \sqsubseteq B$.²

Theorem 2. *Given an \mathcal{EL}^+ TBox $\mathcal{T} = (\mathcal{I}_s \uplus \mathcal{I}_r)$ in normal form, the pinpointing algorithm terminates in time exponential in the size of \mathcal{T} . After termination, the resulting set of assertions \mathcal{A} satisfies the following two properties for all concept names A, B occurring in \mathcal{T} :*

1. $A \sqsubseteq_{\mathcal{T}} B$ iff $(A, B) \in \mathcal{A}$, and
2. $lab((A, B))$ is a pinpointing formula for \mathcal{T} w.r.t $A \sqsubseteq B$.

As an example, consider the TBox \mathcal{T} consisting of the refutable GCIs given in (II) and of no irrefutable axioms. The pinpointing algorithm proceeds as follows. Since $ax_2 : A \sqsubseteq Y \in \mathcal{T}$ and $(A, A) \in \mathcal{A}$ with label \mathfrak{t} , the assertion (A, Y) is added to \mathcal{A} with label ax_2 (actually with label $ax_2 \wedge \mathfrak{t}$, which is equivalent to ax_2). Since $ax_1 : A \sqsubseteq \exists r.A \in \mathcal{T}$ and $(A, A) \in \mathcal{A}$ with label \mathfrak{t} , (A, r, A) is added to \mathcal{A} with label ax_1 . Since $ax_4 : Y \sqsubseteq B \in \mathcal{T}$ and $(A, Y) \in \mathcal{A}$ with label ax_2 , (A, B) is added to \mathcal{A} with label $ax_2 \wedge ax_4$. Finally, since $ax_3 : \exists r.Y \sqsubseteq B \in \mathcal{T}$, $(A, Y) \in \mathcal{A}$ with label ax_2 , and $(A, r, A) \in \mathcal{A}$ with label ax_1 , the label of $(A, B) \in \mathcal{A}$ is modified from $ax_2 \wedge ax_4$ to $(ax_2 \wedge ax_4) \vee (ax_1 \wedge ax_2 \wedge ax_3)$. This final label of (A, B) is a pinpointing formula for \mathcal{T} w.r.t. $A \sqsubseteq B$.

As described until now, our pinpointing algorithm for \mathcal{EL}^+ can only deal with normalized TBoxes, i.e., the pinpointing formula ϕ it yields contains propositional variables corresponding to the normalized GCIs. We now show that the algorithm for computing pinpointing formulae can easily be extended to one dealing also with non-normalized TBoxes. First, note that the relationship between original axioms and normalized axioms is many to many: one axiom in the original TBox can give rise to several axioms in the normalized one, and one axiom in the normalized TBox can come from several axioms in the original

² A proof of this fact in a more general setting can be found in [6].

TBox. For example, consider the GCIs $A \sqsubseteq B_1 \sqcap B_2, A \sqsubseteq B_2 \sqcap B_3$, which are normalized to $A \sqsubseteq B_1, A \sqsubseteq B_2, A \sqsubseteq B_3$. Each original GCI gives rise to two normalized ones, and the normalized GCI $A \sqsubseteq B_2$ has two sources, i.e., it is present in the normalized TBox if the first *or* the second original GCI is present in the input TBox.

Now, assume that $\widehat{\mathcal{T}}$ is an unnormalized input TBox, and that \mathcal{T} is the corresponding normalized TBox where we view all axioms in \mathcal{T} as being refutable. Let ϕ be a pinpointing formula for \mathcal{T} w.r.t. $A \sqsubseteq B$, where A, B are concept names occurring in $\widehat{\mathcal{T}}$ (and thus also in \mathcal{T}). We can now modify ϕ to a *pinpointing formula for the original TBox* $\widehat{\mathcal{T}}$ as follows. Assume that the refutable axioms in $\widehat{\mathcal{T}}$ are associated with unique propositional variables, and the irrefutable ones in $\widehat{\mathcal{T}}$ with \mathbf{t} . Each normalized axiom in \mathcal{T} has a finite number of original axioms as sources. We modify ϕ by replacing the propositional variable for each normalized axiom by the disjunction of the labels of its sources. Note, in particular, that the propositional variable of a normalized axiom that has an irrefutable axiom as source is replaced by a formula that is equivalent to \mathbf{t} .

4 The Complexity of Computing All MinAs

In this section we will show several hardness results regarding the computation of all MinAs. We can actually show all of them already for the sublanguage \mathcal{HL} of \mathcal{EL}^+ . Of course, these results then also hold for \mathcal{EL} and \mathcal{EL}^+ .

If we want to compute all MinAs, then in the worst case an exponential runtime cannot be avoided since there may be *exponentially many MinAs* for a given TBox. The following example shows that this is already the case for \mathcal{HL} TBoxes.

Example 1. For all $n \geq 1$, the size of the \mathcal{HL} TBox

$$\mathcal{T}_n := \{B_{i-1} \sqsubseteq P_i \sqcap Q_i, P_i \sqsubseteq B_i, Q_i \sqsubseteq B_i \mid 1 \leq i \leq n\}$$

is linear in n , and we have $B_0 \sqsubseteq_{\mathcal{T}_n} B_n$. Assume that all axioms in \mathcal{T}_n are refutable. Then, there are 2^n MinAs for \mathcal{T}_n w.r.t. $B_0 \sqsubseteq B_n$ since, for each $i, 1 \leq i \leq n$, it is enough to have $P_i \sqsubseteq B_i$ or $Q_i \sqsubseteq B_i$ in the set.

In Section 5 we will show that a single MinA can be computed in polynomial time. However, as soon as we want to know more about the properties of the set of *all* MinAs, this cannot be achieved in polynomial time (unless $P=NP$). For example, determining whether there is a MinA whose cardinality is bounded by a given natural number n is NP-hard.

Theorem 3. *Given an \mathcal{HL} TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$, concept names A, B occurring in \mathcal{T} , and a natural number n , it is NP-complete to decide whether or not there is a MinA for \mathcal{T} w.r.t. $A \sqsubseteq B$ of cardinality $\leq n$. This already holds in the case where $\mathcal{T}_s = \emptyset$.*

Proof. The problem is in NP since one can simply guess a subset \mathcal{S} of \mathcal{T}_r with cardinality n , and then check in polynomial time whether $A \sqsubseteq_{\mathcal{T}_s \cup \mathcal{S}} B$. Clearly, such a set exists iff there is a MinA of cardinality $\leq n$.

NP-hardness can be shown by a reduction of the NP-hard *hitting set problem* [13]: given a collection S_1, \dots, S_k of sets and a natural number n , is there a set S of cardinality $\leq n$ such that $S \cap S_i \neq \emptyset$ for $i = 1, \dots, k$. Such a set S is called a *hitting set*. In the reduction, we use a concept name P for every element $p \in S_1 \cup \dots \cup S_n$ as well as the additional concept names A, B, Q_1, \dots, Q_k . Given $S_1 = \{p_{11}, \dots, p_{1\ell_1}\}, \dots, S_k = \{p_{k1}, \dots, p_{k\ell_k}\}$, we define the TBox $\mathcal{T} = (\emptyset \uplus \mathcal{T}_r)$ with

$$\mathcal{T}_r := \{P_{ij} \sqsubseteq Q_i \mid 1 \leq i \leq k, 1 \leq j \leq \ell_i\} \cup \{A \sqsubseteq P_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq \ell_i\} \cup \{Q_1 \sqcap \dots \sqcap Q_k \sqsubseteq B\}.$$

It is easy to see that S_1, \dots, S_k has a hitting set of cardinality $\leq n$ iff there is a MinA for \mathcal{T} w.r.t. $A \sqsubseteq B$ of cardinality $\leq n + k + 1$. \square

Given the fact that a TBox may have exponentially many MinAs, it is clear that it is not possible to enumerate all MinAs in time polynomial in the size of the input. However, in complexity theory one also considers other kinds of complexity measures for the complexity of enumeration problems [16]. One possibility is to ask whether there is an algorithm that enumerates all MinAs in time polynomial in the size of the input *and the output*, i.e., in the size of the TBox and the number of MinAs. We will call such an algorithm *output polynomial*. One advantage of an output polynomial algorithm is that it runs in polynomial time in case there are only polynomially many outputs.

The pinpointing algorithm for \mathcal{EL}^+ described in Section 3 uses as a subprocedure the enumeration of all minimal valuations satisfying a given monotone Boolean formula. Unfortunately, already this problem is known not to have an output polynomial solution (unless $P=NP$). A proof of this fact can be found in the technical report [11]; since this result is not included in the corresponding journal paper [12], we provide our own proof for the sake of completeness.

Theorem 4. *There is no output polynomial algorithm for computing all minimal satisfying valuations of monotone Boolean formulae, unless $P=NP$.*

To prove this theorem, it is enough to show (see [17]) that the following decision problem is NP-hard:

Lemma 1. *Given a monotone Boolean formula ϕ and a set \mathcal{M} of minimal valuations satisfying ϕ , deciding whether there exists a minimal valuation $\mathcal{V} \notin \mathcal{M}$ satisfying ϕ is NP-hard in the size of ϕ and \mathcal{M} .*

Proof. The proof is by reduction of the NP-hard *hypergraph 2-coloring problem* [13]: given a collection $H = \{E_1, \dots, E_m\}$ of subsets of a set of vertices V , each of them of size 3, is there a set C such that $C \cap E_i \neq \emptyset$ and $(V \setminus C) \cap E_i \neq \emptyset$ for $i = 1, \dots, m$? \square

³ In other words, both C and its complement must be hitting sets for E_1, \dots, E_m .

Let $V = \{v_1, \dots, v_n\}$ and $E_i = \{v_{i1}, v_{i2}, v_{i3}\}$ for all $i = 1, \dots, m$. We represent every $v_i \in V$ by a propositional variable p_i , and construct the monotone Boolean formula $\phi := \psi \vee \bigvee_{i=1}^m \psi_i$, where

$$\psi = \bigwedge_{i=1}^m p_{i1} \vee p_{i2} \vee p_{i3} \quad \text{and} \quad \psi_i = p_{i1} \wedge p_{i2} \wedge p_{i3}$$

and the set $\mathcal{M} := \{\mathcal{V}_i := \{p_{i1}, p_{i2}, p_{i3}\} \mid 1 \leq i \leq m \text{ and no strict subset of } \mathcal{V}_i \text{ satisfies } \psi\}$.

It is easy to see that the formula ϕ as well as the set \mathcal{M} can be constructed in time polynomial in the size of V and H . Moreover, every valuation $\mathcal{V}_i \in \mathcal{M}$ satisfies the formula ψ_i , and hence also ϕ . It is minimal since no strict subset of \mathcal{V}_i satisfies (i) any of the ψ_j (which require valuations of size at least 3 to be satisfied) nor (ii) ψ since otherwise the condition in the definition of \mathcal{M} would be violated. This shows that ϕ and \mathcal{M} indeed form an instance of the problem considered in the lemma.

To complete the proof of NP-hardness of this problem, it remains to be shown that there is a minimal valuation $\mathcal{V} \notin \mathcal{M}$ satisfying ϕ iff there is a set $C \subseteq V$ such that $C \cap E_i \neq \emptyset$ and $(V \setminus C) \cap E_i \neq \emptyset$ for all $1 \leq i \leq m$.

For the *if direction*, let C be such a set, which we assume without loss of generality to be minimal with respect to set inclusion. We define the valuation $\mathcal{V}_C := \{p_i \mid v_i \in C\}$ and claim that it is the minimal valuation we are looking for. For every $1 \leq i \leq m$, $C \cap E_i \neq \emptyset$ implies that there is a $1 \leq j \leq 3$ such that $v_{ij} \in C$, which means that $p_{ij} \in \mathcal{V}_C$. This shows that \mathcal{V}_C satisfies ψ and thus also ϕ . In addition, since $(V \setminus C) \cap E_i \neq \emptyset$, there is a $1 \leq k \leq 3$ such that $v_{ik} \notin C$. Thus, \mathcal{V}_C is different from all the valuations $\mathcal{V}_i \in \mathcal{M}$, and it does not satisfy any of the formulae ψ_i .

To show that \mathcal{V}_C is minimal, assume that $\mathcal{V}' \subset \mathcal{V}_C$. Since C is minimal, the set $C' := \{v_i \mid p_i \in \mathcal{V}'\} \subset C$ is such that there is a $1 \leq i \leq m$ with $C' \cap E_i = \emptyset$. This implies that \mathcal{V}' does not satisfy $p_{i1} \vee p_{i2} \vee p_{i3}$, and hence it does not satisfy ψ . As a subset of \mathcal{V}_C , it also does not satisfy any of the formulae ψ_i , and thus it does not satisfy ϕ . This shows that \mathcal{V}_C is a minimal valuation satisfying ϕ that does not belong to \mathcal{M} .

For the *only-if direction*, assume that there is a minimal valuation $\mathcal{V} \notin \mathcal{M}$ satisfying ϕ . This valuation cannot satisfy any of the formulae ψ_i . Indeed, (i) for $\mathcal{V}_i \in \mathcal{M}$ this would imply that \mathcal{V} is a superset of one of the valuations in \mathcal{M} , which contradicts either the minimality of \mathcal{V} or the fact that it does not belong to \mathcal{M} ; (ii) for $\mathcal{V}_i \notin \mathcal{M}$ there would be a smaller valuation satisfying ψ , which contradicts the minimality of \mathcal{V} .

Since \mathcal{V} is a model of ϕ , it must thus satisfy ψ . Define the set $C_{\mathcal{V}} := \{v_i \mid p_i \in \mathcal{V}\}$. Since \mathcal{V} satisfies ψ , for every $1 \leq i \leq m$ there is a $1 \leq j \leq 3$ such that $p_{ij} \in \mathcal{V}$, and thus $v_{ij} \in C_{\mathcal{V}} \cap E_i$. On the other hand, since \mathcal{V} does not satisfy any of the formulae ψ_i , for every $1 \leq i \leq m$ there must also be a $1 \leq l \leq 3$ such that $p_{il} \notin \mathcal{V}$, which means that $E_i \not\subseteq C_{\mathcal{V}}$ and hence $(V \setminus C_{\mathcal{V}}) \cap E_i \neq \emptyset$. \square

Theorem [4](#) follows from this lemma since an output polynomial algorithm whose runtime is bounded by the polynomial $P(|\phi|, |\mathcal{M}|)$ (where ϕ is the input and \mathcal{M}

Algorithm 1. Compute one MinA for $\mathcal{T} = (\emptyset \uplus \{t_1, \dots, t_n\})$ w.r.t. $A \sqsubseteq B$.

```

1: if  $A \not\sqsubseteq_{\mathcal{T}} B$  then
2:   return no MinA
3:  $\mathcal{S} := \{t_1, \dots, t_n\}$ 
4: for  $1 \leq i \leq n$  do
5:   if  $A \sqsubseteq_{\mathcal{S} \setminus \{t_i\}} B$  then
6:      $\mathcal{S} := \mathcal{S} \setminus \{t_i\}$ 
7: return  $\mathcal{S}$ 

```

the output) could be used to decide the problem introduced in the lemma in polynomial time as follows: given ϕ and \mathcal{M} , run the algorithm for time at most $P(|\phi|, |\mathcal{M}|)$ and check whether the generated valuations are exactly those in \mathcal{M} .

Theorem 4 shows that an algorithm for computing all MinAs based on computing the pinpointing formula and then producing its minimal satisfying valuations cannot be output polynomial. However, we can also use Theorem 4 to show that there cannot be any algorithm for computing MinAs that is output polynomial.

Theorem 5. *There is no output polynomial algorithm that computes, for a given \mathcal{HL} TBox $\mathcal{T} = (\mathcal{T}_s \uplus \mathcal{T}_r)$ and concept names A, B occurring in \mathcal{T} , all MinAs for \mathcal{T} w.r.t. $A \sqsubseteq B$, unless $P=NP$.*

Proof. We show that the problem of computing minimal valuations of monotone Boolean formulae can be reduced in polynomial time to the problem of computing MinAs of an \mathcal{HL} TBox. Given a monotone Boolean formula ϕ , we introduce one concept name B_ψ for every subformula ψ of ϕ , and one additional concept name A . We define TBoxes \mathcal{T}_ψ for the subformulae ψ of ϕ by induction: if $\psi = p$ is a propositional variable, then $\mathcal{T}_\psi := \{A \sqsubseteq B_p\}$; if $\psi = \psi_1 \wedge \psi_2$, then $\mathcal{T}_\psi := \{B_{\psi_1} \sqcap B_{\psi_2} \sqsubseteq B_\psi\}$; if $\psi = \psi_1 \vee \psi_2$, then $\mathcal{T}_\psi := \{B_{\psi_1} \sqsubseteq B_\psi, B_{\psi_2} \sqsubseteq B_\psi\}$.

Obviously, the size of \mathcal{T}_ϕ is linear in the size of ϕ . In \mathcal{T}_ϕ , we declare the GCIs $A \sqsubseteq B_p$ with p a propositional variable to be refutable, and the other GCIs to be irrefutable. With this division of \mathcal{T}_ϕ into a static and a refutable part, it is easy to see that there is a 1–1-correspondence between the minimal satisfying valuations of ϕ and the MinAs for \mathcal{T}_ϕ w.r.t. $A \sqsubseteq B_\phi$. In particular, given a MinA \mathcal{S} , the corresponding valuation $\mathcal{V}_{\mathcal{S}}$ consists of all p such that $A \sqsubseteq B_p \in \mathcal{S}$. Thus, if we could compute all MinAs with an output polynomial algorithm, we could do the same for all minimal satisfying valuations. \square

5 Computing One MinA

For the sake of simplicity, we restrict the attention in this section to the case where all axioms in the TBox are assumed to be refutable. Note, however, the results could easily be extended to the general case.

A single MinA can be computed in polynomial time by the simple Algorithm 1, which goes through all axioms (in a given fixed order) and throws away those

that are not needed to obtain the desired subsumption relationship. Since the algorithm performs $n + 1$ subsumption tests (where n is the cardinality of \mathcal{T}), and each such test takes only polynomial time, the overall complexity of this algorithm is polynomial. It is easy to see that its output (in case $A \sqsubseteq_{\mathcal{T}} B$) is indeed a MinA for \mathcal{T} w.r.t. $A \sqsubseteq B$.

Theorem 6. *Given an \mathcal{EL}^+ TBox $\mathcal{T} = (\emptyset \uplus \mathcal{T}_r)$, Algorithm 1 terminates in time polynomial in the size of \mathcal{T} , and yields a MinA for \mathcal{T} w.r.t. $A \sqsubseteq B$ if $A \sqsubseteq_{\mathcal{T}} B$.*

Although it requires only polynomial time, computing one MinA using Algorithm 1 may still be impractical for very large TBoxes like SNOMED. In fact, the algorithm has to make as many calls of the subsumption algorithm as there are axioms in the TBox (in the case of SNOMED, more than 350,000). Here we propose an *improved algorithm* that proceeds in two steps: (i) first compute a small (though not necessarily minimal) subset of the TBox from which the subsumption relationship follows; (ii) then minimize this set using Algorithm 2. Of course, this approach makes sense only if the algorithm used in step (i) is efficient and produces fairly small sets. It wouldn't help to use the trivial algorithm that always produces the whole TBox. In the following, we denote by nMinA such a (not necessarily minimal) subset obtained by step (i).

An algorithm that realize step (i) and runs in polynomial time can easily be obtained from the pinpointing algorithm sketched in Section 3 by strengthening the preconditions of rule applicability. The only modification is the following: if an assertion in the consequence of a rule already belongs to the current set of assertions, then this rule is not applied, i.e., once an assertion is there with some label, the label remains unchanged. Thus, every assertion (A, B) in the final set has a conjunction of propositional variables as its label, which clearly corresponds to a subset of the TBox from which the subsumption relationship $A \sqsubseteq B$ follows. In general, this subset is not minimal, however. (Because of the space constraints, we cannot give an example demonstrating this.)

As described until now, this modified algorithm works on normalized TBoxes. To get an appropriate subset of the original axioms, one can use a greedy strategy for producing a set of original axioms that covers a given set \mathcal{S} of normalized axioms in the following sense. For each original axiom t , let \mathcal{S}_t be the set of normalized axioms t gives rise to. The set \mathcal{T}' of original axioms *covers* \mathcal{S} if $\mathcal{S} \subseteq \bigcup_{t \in \mathcal{T}'} \mathcal{S}_t$. The use of a greedy strategy adds another possible source of non-minimality. (We use a non-optimal greedy strategy to keep the algorithm polynomial. In fact, even determining whether there is a cover set of size $\leq n$ is another NP-complete problem [13].)

Our preliminary experimental results confirm that this algorithm is indeed more practical than Algorithm 1. Based on the *refined algorithm* underlying the CEL reasoner [5, 4], we have implemented the practical algorithm described above for computing *exactly one* MinA for each subsumption relationship in \mathcal{EL}^+ . The experiments were run on a variant of the Galen Medical Knowledge

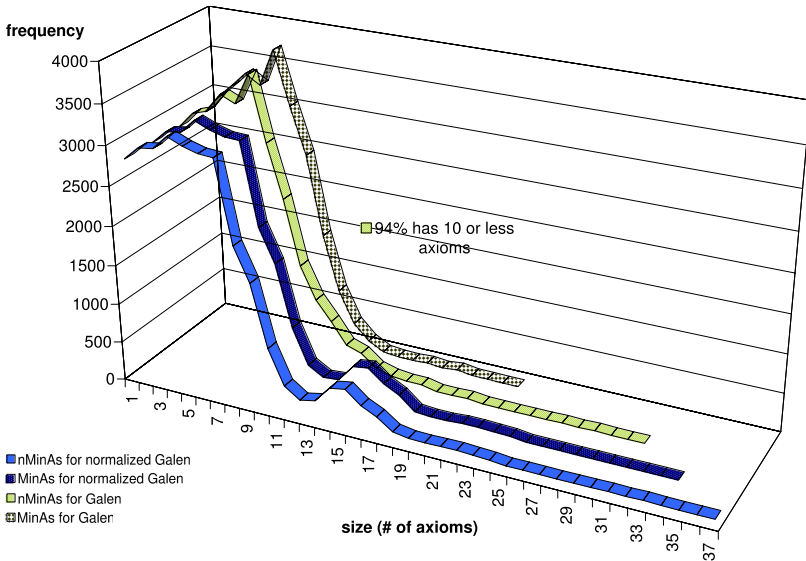


Fig. 2. Statistical data on the sizes of all computed axiom sets

Base [21]⁴ which is a TBox consisting of more than 4,000 axioms. On the normalized version of this TBox, CEL needs about 14 sec to compute all subsumption relationships between concept names occurring in this TBox. Overall, over 27,000 subsumption relationships are computed. The overhead for computing for all of these subsumption relationships (possibly non-minimal) subsets from which they already follow was a bit more than 50%: the modified pinpointing algorithm described above needed about 23 sec. Going from the nMinAs for the normalized TBox to the corresponding nMinAs for the original Galen TBox with the greedy strategy took 0.27 sec. Finally, the overall time required for minimizing these sets using Algorithm 1 (with CEL [4] as the subsumption reasoner) was 9:45 min. For these last two numbers one should take into account, however, that these involved treating more than 27,000 such sets. For a single such set, the average post-processing time was negligible (on average 21 milliseconds). Also note that applying Algorithm 1 directly to the whole TBox for just one subsumption relationship (between *Renal-Artery* and *Artery-Which-Has-Laterality*) took more than 7 hours.

Thus, from the point of view of runtime, our practical algorithm behaves quite well on Galen. The same can be said about the quality of its results. Figure 2 displays the distribution graphs of the sizes of all computed nMinAs and their corresponding MinAs. The average size of an axiom set computed by the algorithm before using Algorithm 1 to minimize it was 5 (with maximum size 31), which is quite small and thus means that this set can directly be given to the user

⁴ Since Galen uses expressivity not available in \mathcal{EL}^+ , we have simplified it by removing inverse role axioms and treating functional roles as ordinary ones.

as an explanation for the subsumption relationship. Also, the computed nMinAs were almost minimal: on average, the possibly non-minimal sets computed by the algorithm were only 2.59% larger than the minimal ones. When considering the normalized TBox (i.e., without translating back to the original TBox), this number was even better (0.1%). This means that in most cases it is probably not necessary to further minimize the sets using Algorithm 1. If demanded by the user for a specific subsumption relationship it can still be done without taking much time.

6 Additional and Future Work on Pinpointing

The pinpointing extension of the subsumption algorithm for \mathcal{EL} described in Section 3 as well as the pinpointing algorithm for \mathcal{ALC} described in 3 are instances of a general approach for modifying “tableau-like” reasoning procedures to pinpointing procedures 6.

Instead of computing minimal subsets that have a given consequence, one sometimes also wants to compute maximal subsets that do not have a given consequence. Given the pinpointing formula ϕ , these sets correspond to maximal valuations that do not satisfy ϕ . The complexity results from Section 4 hold accordingly for such maximal sets. However, we currently do not know how to obtain a practical algorithm computing one such set (i.e., the results of Section 5 cannot be transferred to the case of maximal sets). Another open problem is the question of whether Theorem 5 also holds in the special case where the static TBox is empty.

Finally, space optimizations shall be studied to cater for large ontologies such as SNOMED with up to ten million subsumptions, and thus nMinAs.

References

- [1] Baader, F., Brandt, S., Lutz, C.: Pushing the \mathcal{EL} envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Edinburgh (UK), pp. 364–369. Morgan Kaufmann, Los Altos (2005)
- [2] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
- [3] Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. *J. of Automated Reasoning* 14, 149–180 (1995)
- [4] Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 287–291. Springer, Heidelberg (2006)
- [5] Baader, F., Lutz, C., Suntisrivaraporn, B.: Is tractable reasoning in extensions of the description logic \mathcal{EL} useful in practice. *Journal of Logic, Language and Information*, Special Issue on Method for Modality on M4M (to appear, 2007)
- [6] Baader, F., Penaloza, R.: Axiom pinpointing in general tableaux. LTCS-Report LTCS-07-01, Germany, See (2006), <http://lat.inf.tu-dresden.de/research/reports.html>

- [7] Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Hermenegildo, M.V., Cabeza, D. (eds.) *Practical Aspects of Declarative Languages*. LNCS, vol. 3350, pp. 174–186. Springer, Heidelberg (2005)
- [8] Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: de Mántaras, R.L., Saitta, L. (eds.) *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, pp. 298–302 (2004)
- [9] Davydov, G., Davydova, I., Büning, H.K.: An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. *Ann. of Mathematics and Artificial Intelligence* 23(3–4), 229–245 (1998)
- [10] Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming* 1(3), 267–284 (1984)
- [11] Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. Technical Report CD-TR 91/16, Christian Doppler Labor für Expertensysteme, TU-Wien (1991)
- [12] Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.* 24(6), 1278–1304 (1995)
- [13] Garey, M.R., Johnson, D.S.: *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (1979)
- [14] Hollunder, B.: Hybrid inferences in KL-ONE-based knowledge representation systems. In: *Proc. of the German Workshop on Artificial Intelligence*, pp. 38–47. Springer, Heidelberg (1990)
- [15] Horrocks, I., Patel-Schneider, P.F., Van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics* 1(1), 7–26 (2003)
- [16] Johnson, D.S., Yannakakis, M., Papadimitriou, C.H.: On generating all maximal independent sets. *Inf. Process. Lett.* (1988)
- [17] Kavvadias, D.J., Sideri, M., Stavropoulos, E.C.: Generating all maximal models of a Boolean expression. *Inf. Process. Lett.* (2000)
- [18] Liffiton, M.H., Sakallah, K.A.: On finding all minimally unsatisfiable subformulas. In: Bacchus, F., Walsh, T. (eds.) *SAT 2005*. LNCS, vol. 3569, pp. 173–186. Springer, Heidelberg (2005)
- [19] Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic \mathcal{ALC} . In: *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, AAAI Press/The MIT Press (2006)
- [20] Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Ellis, A., Hagino, T. (eds.) *Proc. of the 14th International Conference on World Wide Web (WWW'05)*, pp. 633–640. ACM Press, New York (2005)
- [21] Rector, A., Horrocks, I.: Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In: *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, AAAI Press (1997)
- [22] Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, Acapulco, Mexico, pp. 355–362. Morgan Kaufmann, Los Altos (2003)
- [23] Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. *Artificial Intelligence* 48(1), 1–26 (1991)

- [24] Spackman, K.A., Campbell, K.E., Cote, R.A.: SNOMED RT: A reference terminology for health care. *J. of the American Medical Informatics Association (Fall Symposium Supplement)*, 640–644 (1997)
- [25] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25, 25–29 (2000)
- [26] Zhang, L., Malik, S.: Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In: *Proc. of the Conference on Design, Automation and Test in Europe (DATE'03)*, pp. 10880–10885. IEEE Computer Society Press, Los Alamitos (2003)

Integrating Action Calculi and Description Logics

Conrad Drescher and Michael Thielscher

Department of Computer Science,
Dresden University of Technology
Nöthnitzer Str. 46, 01187 Dresden, Germany

Abstract. General action languages, like e.g. the Situation Calculus, use full classical logic to represent knowledge of actions and their effects in dynamic domains. Description Logics, on the other hand, have been developed to represent static knowledge with the help of decidable subsets of first order logic. In this paper, we show how to use Description Logic as the basis for a decidable yet still expressive action formalism. To this end, we use ABoxes as decidable state descriptions in the basic Fluent Calculus. As a second contribution, we thus obtain an independent semantics – based on a general action formalism – for a recent method for ABox-Update.

1 Introduction

General action languages like the Situation Calculus [1] or the Fluent Calculus [2] are highly expressive formalisms for representing knowledge of actions and effects in dynamic domains. In this way, they provide the formal foundations for programming languages and systems for the design of logically reasoning agents who can execute high-level strategies and solve planning problems [3]. However, the use of full classical logic as the basis for these calculi implies, in general, undecidability even of static questions such as whether the current state knowledge entails that a specific action is executable. The existing solutions to this problem often restrict the action calculi to being essentially propositional and/or employing the closed-world assumption. Description Logics, on the other hand, provide expressive but decidable languages for the representation of static knowledge. In particular, they are of far greater expressivity than propositional logic. Efficient decision procedures have been developed and implemented for a variety of such logics [4].

In this paper, we show how to integrate Description Logics into a general action formalism. Our motivation is two-fold: On the one hand, the integration allows to restrict the expressiveness of general reasoning about actions to expressive yet decidable fragments of first order logic. This also provides the formal foundations for integrating decision procedures for Description Logics into action programming languages and systems, which will allow agents to resort to these algorithms whenever they have to verify conditions against their state

knowledge. On the other hand, the integration of Description Logics into an action language provides a semantics for a recent definition of ABox-Update [5], which is thus embedded into a general formalism for reasoning about actions and change.

The specific contributions of this paper are the following:

1. We show how ABoxes can be used as expressive, decidable state descriptions in the basic Fluent Calculus.
2. We provide semantics for ABox-Update by capturing them with Fluent Calculus state update axioms.
3. We lay the theoretical foundations for a practical action programming language built on top of Description Logic reasoners.

The rest of the paper is organized as follows: In Section 2 we recall the basics of the Fluent Calculus and give a brief introduction to Description Logics. In Section 3 we show how ABoxes can be used as state descriptions in the Fluent Calculus, and we prove that state update axioms provide a correct characterization of ABox-Update. Furthermore, we show how to integrate simple TBox reasoning and discuss some of the problems that arise in the general case. After a discussion of related work, we conclude with a summary and outlook.

2 Preliminaries

In this section, we introduce the general action formalism Fluent Calculus; we assume familiarity with the classical Situation Calculus. We then recall the very essentials of Description Logics.

2.1 Fluent Calculus

The Fluent Calculus is a general action formalism: it enables the axiomatization of dynamic domains, i.e. of initial knowledge about the world, action preconditions and action effects. As running example of a dynamic domain we will use the following simplistic online-store scenario; we will give a Fluent Calculus axiomatization of this scenario at the end of this section.

Example 1. Initially, all that is known is that customer John has ordered the item *NiceBook*. An order cancellation can be processed only if the order is known. If the order already has been paid for, the customer is entitled to a refund.

We refer to the mutable properties of a dynamic domain as the *fluents*. In the Situation Calculus fluents are modelled as first order atoms, extended by an additional argument for a point in time, e.g. $\text{Ordered}(\text{John}, \text{NiceBook}, S_0)$. The Fluent Calculus extends the Situation Calculus with an explicit notion of a *state* associated with a situation, denoted $\text{State}(S_0)$. Intuitively, a state may be identified with the set of all the fluents that hold at any one time. To this end *reification* is employed: both fluents and states are modelled as terms; cf. $\text{Holds}(\text{Ordered}(\text{John}, \text{NiceBook}), \text{State}(S_0))$. This allows to apply first-order

quantification to fluents and states, which in turn is helpful for devising a solution to the famous Frame Problem. In the following we give a compact, formal introduction to the technical basics of Fluent Calculus and the axiom schemes employed to encode dynamic domains.

Basics of Fluent Calculus. Fluent Calculus is based on many-sorted classical logic with equality. The standard sorts are OBJECT, ACTION, SITUATION, FLUENT and STATE, with FLUENT a sub-sort of STATE.¹ A term of sort FLUENT is a *fluent* – analogously we speak of *states*, *situations*, *actions* and *objects*. Situations are sequences of actions rooted in an initial situation S_0 – e.g. $\text{Do}(\text{Order}(\text{NiceBook}), S_0)$. Just as in the classical Situation Calculus, they provide a branching time structure for Fluent Calculus. At the heart of Fluent Calculus is an axiomatization of states representing combinations of fluents.

Definition 1 (basic signature). *The signature of Fluent Calculus contains:*

- A countable infinity of function symbols into sort OBJECT and FLUENT – but only a finite number thereof into sort ACTION.²
- Two symbols for functions into sort situation:
 - S_0 : SITUATION — the initial situation.
 - Do : ACTION \times SITUATION \rightarrow SITUATION — mapping a situation to its successor, as the result of executing an action.
- Three symbols for functions into states:
 - \emptyset : STATE — the empty state.
 - \circ : STATE \times STATE \rightarrow STATE — for conjoining fluents into states and states into bigger states.
 - $State$: SITUATION \rightarrow STATE — denoting the state of a situation.
- A binary predicate symbol $Poss$: ACTION \times SITUATION — relating action preconditions to situations.

To gain an intuition for the role played by \circ , compare Situation Calculus’

$$\text{Ordered}(\text{John}, \text{NiceBook}, S_0) \wedge \text{Ordered}(\text{Mary}, \text{EvenNicerBook}, S_0)$$

with Fluent Calculus’

$$(\exists z)\text{State}(S_0) = \text{Ordered}(\text{John}, \text{NiceBook}) \circ \text{Ordered}(\text{Mary}, \text{EvenNicerBook}) \circ z.$$

Definition 2 (holds macro). *A fluent f is said to hold in a state z if the latter is composed of f and some other state z' via \circ ; a fluent holds in a situation if it holds in the state of the situation:*

$$\begin{aligned} \text{Holds}(f, z) &\stackrel{def}{=} (\exists z')z = f \circ z' \text{ and} \\ \text{Holds}(f, s) &\stackrel{def}{=} \text{Holds}(f, \text{State}(s)). \end{aligned}$$

¹ By convention, variables x, a, s, f and z are used for objects, actions, situations, fluents and states, respectively.

² Each with arguments of sort OBJECT only.

The foundational axioms Σ_{state} of the Fluent Calculus govern the behavior of states.

Definition 3 (foundational axioms). ³

| | |
|--|----------------------------|
| $(z_1 \circ z_2) \circ z_3 = z_1 \circ (z_2 \circ z_3)$ | (<i>Associativity</i>) |
| $z_1 \circ z_2 = z_2 \circ z_1$ | (<i>Commutativity</i>) |
| $\neg Holds(f, \emptyset)$ | (<i>Empty state</i>) |
| $Holds(f_1, f_2) \supset f_1 = f_2$ | (<i>Irreducibility</i>) |
| $Holds(f, z_1 \circ z_2) \supset (Holds(f, z_1) \vee Holds(f, z_2))$ | (<i>Decomposition</i>) |
| $(\forall f)(Holds(f, z_1) \equiv Holds(f, z_2)) \supset z_1 = z_2$ | (<i>State equality</i>) |
| $(\forall P)(\exists z)(\forall f)(Holds(f, z) \equiv P(f))$ | (<i>State existence</i>) |

where P is a unary predicate variable of sort FLUENT.

The last axiom ensures the existence of a state for every combination of fluents. For a detailed introduction to this and the other axioms the interested reader is referred to [6].

Definition 4 (finite state). A finite state ϑ is a term $f_1 \circ \dots \circ f_n$ such that each f_i ($1 \leq i \leq n$) is a fluent. If $n = 0$, then $\vartheta = \emptyset$.

Definition 5 (fluent addition/subtraction). The following macros provide an intuitive notation for describing relations between different states:

- $z_1 + f \stackrel{def}{=} z_1 \circ f$
- $z_1 - f \stackrel{def}{=} (z_2 = z_1 \vee z_2 \circ f = z_1) \wedge \neg Holds(f, z_2)$

These definitions are recursively extended to addition and subtraction of finite states ϑ^+ and ϑ^- : these will consist of the positive and negative effects of actions.

We next introduce formulas capable of expressing which (fluent or non-fluent) properties hold in a state and in a situation, respectively.

Definition 6 (state/situation formula). A state formula $\Delta(z)$ is a first order formula with free state variable z and without any occurrences of states other than in expressions of the form $Holds(f, z)$, and without actions or situations. Replacing every occurrence of z by $State(s)$ in a state formula $\Delta(z)$, we obtain a situation formula $\Delta(s)$.

Definition 7 (unique name axioms). Every Fluent Calculus instance includes a set Σ_{una} of unique-name axioms that contains a formula of the form

$$f_i(\mathbf{x}) \neq f_j(\mathbf{y})$$

³ Variables not within the scope of any quantifier are to be read as universally quantified throughout this paper unless otherwise stated.

for each pair of distinct function symbols of sort FLUENT as well as for each pair of distinct function symbols of sort ACTION and a formula of the form

$$(f_i(\mathbf{x}) = f_i(\mathbf{y}) \supset \mathbf{x} = \mathbf{y}),$$

for each function symbol of sort FLUENT or ACTION.

Domain Specifications. In order to specify a dynamic domain we need to axiomatize knowledge about the initial state, action preconditions and the effects resulting from action execution. Formally, a domain is specified as a set of axioms $\Sigma = \Sigma_{\text{state}} \cup \Sigma_{\text{una}} \cup \Sigma_{\text{init}} \cup \Sigma_{\text{poss}} \cup \Sigma_{\text{sua}}$, where :

- $\Sigma_{\text{init}} = \{(\exists z)\text{State}(S_0) = z \wedge \Delta(z)\}$, with $\Delta(z)$ a state formula,
- Σ_{poss} is a set of precondition axioms, one for each action, and
- Σ_{sua} is a set of state update axioms, one for each action.

Definition 8 (precondition axiom). A precondition axiom for action $A(\mathbf{x})$ is a formula $\text{Poss}(A(\mathbf{x}), s) \equiv \Delta(s)$, where $\Delta(s)$ is a situation formula with free variables among \mathbf{x} and s .

Definition 9 (state update axiom). A state update axiom is a formula of the form

$$\begin{aligned} \text{Poss}(A(\mathbf{x}), s) \supset \\ (\exists \mathbf{y}_1)(\Delta_1(s) \wedge \text{State}(\text{Do}(A(\mathbf{x}), s))' = \text{State}(s) - \vartheta_1^- + \vartheta_1^+) \\ \vee \dots \vee \\ (\exists \mathbf{y}_n)(\Delta_n(s) \wedge \text{State}(\text{Do}(A(\mathbf{x}), s))' = \text{State}(s) - \vartheta_n^- + \vartheta_n^+). \end{aligned}$$

The finite states ϑ_i^- and ϑ_i^+ with free variables among \mathbf{x}, \mathbf{y}_i are the negative and positive effects of $A(\mathbf{x})$ under condition $\Delta(s)$. $\Delta(s)$ itself is a situation formula with free variables among \mathbf{x}, \mathbf{y}_i and s .

Example 1. (continued) The online-store scenario from example 1 is axiomatized in Fluent Calculus as follows, illustrating each type of axiom:

$$(\exists z)\text{State}(S_0) = z \wedge \text{Holds}(\text{Ordered}(\text{John}, \text{NiceBook}), z),$$

$$\text{Poss}(\text{CancelOrder}(\text{customer}, \text{item}), s) \equiv \text{Holds}(\text{Ordered}(\text{customer}, \text{item}), s),$$

$$\begin{aligned} \text{Poss}(\text{CancelOrder}(\text{customer}, \text{item}), s) \supset \\ (\text{Holds}(\text{Paid}(\text{item}), s) \wedge \text{State}(\text{Do}(\text{CancelOrder}(\text{customer}, \text{item}), s)) = \\ \text{State}(s) - \text{Ordered}(\text{customer}, \text{item}) + \text{Refund}(\text{customer}, \text{item})) \\ \vee \\ (\neg \text{Holds}(\text{Paid}(\text{item}), s) \wedge \text{State}(\text{Do}(\text{CancelOrder}(\text{customer}, \text{item}), s)) = \\ \text{State}(s) - \text{Ordered}(\text{customer}, \text{item})). \end{aligned}$$

| Name | Syntax | Semantics |
|-------------------------|---------------|--|
| negation | $\neg C$ | $\mathfrak{D}^{\mathcal{I}} \setminus C^{\mathcal{I}}$ |
| conjunction | $C \sqcap D$ | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| disjunction | $C \sqcup D$ | $C^{\mathcal{I}} \cup D^{\mathcal{I}}$ |
| existential restriction | $\exists R.C$ | $\{x \mid \exists y(x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ |
| universal restriction | $\forall R.C$ | $\{x \mid \forall y(x, y) \in R^{\mathcal{I}} \supset y \in C^{\mathcal{I}}\}$ |

Fig. 1. Syntax and semantics of \mathcal{ALC}

This axiomatization entails

$$\neg \text{Holds}(\text{Ordered}(\text{John}, \text{NiceBook}), \text{Do}(\text{CancelOrder}(\text{John}, \text{NiceBook}), S_0))$$

and

$$\text{Holds}(\text{Paid}(\text{NiceBook}), \text{Do}(\text{CancelOrder}(\text{John}, \text{NiceBook}), S_0) \supset$$

$$\text{Holds}(\text{Refund}(\text{John}, \text{NiceBook}), \text{Do}(\text{CancelOrder}(\text{John}, \text{NiceBook}), S_0)).$$

2.2 Description Logics

In this section, we recall those facts about Description Logics (DLs) that are essential to the ensuing discussion. A gentle introduction can be found in [4]. Description Logics are a family of Knowledge Representation formalisms; typically, they are decidable fragments of classical first order logic. In the following we employ the term Description Logic solely for such fragments.

A particular DL is based on a set of concept names N_C (unary predicates), a set of role names N_R (binary predicates), a set of individual names N_I (constants), and a number of constructors for inductively defining complex concepts and roles.

The semantics of Description Logics is defined via interpretations $\mathcal{I} = (\mathfrak{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\mathfrak{D}^{\mathcal{I}}$ is a non-empty set of individuals. The interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $C \in N_C$ to a subset $C^{\mathcal{I}}$ of $\mathfrak{D}^{\mathcal{I}}$, each role name $R \in N_R$ to a binary relation $R^{\mathcal{I}}$ on $\mathfrak{D}^{\mathcal{I}}$, and each individual name $I \in N_I$ to an individual $I^{\mathcal{I}} \in \mathfrak{D}^{\mathcal{I}}$. The semantics is extended inductively to complex concepts and roles. Figure 1 introduces the syntax and semantics of the core DL \mathcal{ALC} .

Definition 10 (ABox). An assertional box (ABox) is a finite, non-empty set of concept assertions $C(I)$ and role assertions $R(I_1, I_2)$ and $\neg R(I_1, I_2)$, where C and R may be complex concepts and roles, respectively.

For example, $\{\text{Outbound} \sqcup \text{Delivered}(\text{Package})\}$ is an ABox expressing uncertainty over the whereabouts of a particular package.

A number of highly-optimized tableau-based reasoners for effectively deciding even very expressive DLs are available [7,8,9].

3 Integration

We will now lay the theoretical foundations for an integration of Description Logics into Fluent Calculus. We will first show how the latter can use DL ABoxes as structured and decidable world descriptions. We then turn our attention to a recently proposed method for ABox-Update: After recalling the essential definitions we establish a Fluent Calculus semantics for these updates, thus relating them to standard AI action calculi. Furthermore, these findings will enable us to identify fragments of Fluent Calculus where questions of action applicability and effects resulting from action execution can effectively be computed.

3.1 ABoxes and State Formulas

We now establish a connection between Description Logic ABoxes and Fluent Calculus state formulas. This connection will be a consequence of a more general result on the relation between first order sentences and state formulas.

Consider an arbitrary countable first order language \mathcal{L} . We can then define a Fluent Calculus instance that contains exactly one function symbol F_i of sort FLUENT for every predicate symbol $P_i \in \mathcal{L}$ (except equality). Moreover, its terms \mathbf{t} of sort OBJECT are precisely the terms of \mathcal{L} .

Definition 11. *The mapping τ_z takes first order sentences in \mathcal{L} to state formulas $\Delta(z)$:*

$$\begin{aligned} \tau_z(P_i(\mathbf{t})) &= \text{Holds}(F_i(\mathbf{t}), z) \\ \tau_z(t_1 = t_2) &= (t_1 = t_2) \\ \tau_z(\varphi \wedge \psi) &= \tau_z(\varphi) \wedge \tau_z(\psi) \\ \tau_z(\neg\varphi) &= \neg\tau_z(\varphi) \\ \tau_z(\exists x\varphi) &= \exists x\tau_z(\varphi). \end{aligned}$$

Theorem 1 (first order sentences and state formulas). *A first order sentence φ in a countable language \mathcal{L} has a model iff $\{\tau_z(\varphi)\} \cup \Sigma_{\text{state}}$ has a model.*

Proof. (\Rightarrow)

First, observe that we can restrict our attention to certain models of φ , namely the term models obtained via the standard Henkin construction [10]. The domain \mathcal{D} of these models consists of equivalence classes on all the terms of \mathcal{L} . Let $\mathcal{M}_1 = (\mathcal{D}_{\mathcal{M}_1}, \cdot^{\mathcal{M}_1}) \models \varphi$ be such a model. Let \mathfrak{F} be the set of all fluents built from terms occurring in an equivalence class in $\mathcal{D}_{\mathcal{M}_1}$.

Then $\mathcal{M}_2 = (\mathcal{D}_{\text{object}}, \mathcal{D}_{\text{fluent}}, \mathcal{D}_{\text{state}}, \cdot^{\mathcal{M}_2}) \models \{\tau_z(\varphi)\} \cup \Sigma_{\text{state}}$ where

- $\mathcal{D}_{\text{object}} = \mathcal{D}_{\mathcal{M}_1}$,
- $\mathcal{D}_{\text{fluent}} = \{\{f\} \mid f \in \mathfrak{F}\}$,
- $\mathcal{D}_{\text{state}} = \mathcal{P}(\mathfrak{F})$, the power set of \mathfrak{F} ,

⁴ In the following we assume without loss of generality that \mathcal{L} contains equality.

- $t^{\mathcal{M}_2} = t^{\mathcal{M}_1}$ for objects t ,
- $F(\mathbf{t})^{\mathcal{M}_2} = \{F(\mathbf{t}^{\mathcal{M}_2})\}$ for each fluent $F(\mathbf{t})$,
- $\emptyset^{\mathcal{M}_2} = \{\}$,
- $(z_1 \circ z_2)^{\mathcal{M}_2} = (z_1)^{\mathcal{M}_2} \cup (z_2)^{\mathcal{M}_2}$, and
- $z^{\mathcal{M}_2} = \{F_i(\mathbf{t}^{\mathcal{M}_2}) \mid \mathcal{M}_1 \models P_i(\mathbf{t})\}$.

Interpreting \emptyset as empty-set, \circ as set-union and states as sets of fluents is a model of the foundational axioms Σ_{state} of Fluent Calculus [6]. The proof is completed by structural induction on φ .

(\Leftarrow) In this case simply let $\mathcal{M}_3 = (\mathfrak{D}_{\text{object}}, \mathfrak{D}_{\text{fluent}}, \mathfrak{D}_{\text{state}}, \cdot^{\mathcal{M}_3}) \models \{\tau_z(\varphi)\} \cup \Sigma_{\text{state}}$. Then $\mathcal{M}_4 = (\mathfrak{D}_{\text{object}}, \cdot^{\mathcal{M}_4}) \models \phi$ where

- $t^{\mathcal{M}_4} = t^{\mathcal{M}_3}$ for terms t of sort *object* and
- $P_i^{\mathcal{M}_4} = \{\mathbf{t}^{\mathcal{M}_4} \mid \mathcal{M}_3 \models \text{Holds}(F_i(\mathbf{t}), z)\}$.

This proof, too, is completed by structural induction on φ . □

This result justifies an intuitive identification of state formulas with the more familiar first order sentences. Moreover, it enables us to transfer known decidability or complexity results for fragments of first order logic to instances of the Fluent Calculus, where state formulas are restricted accordingly. In particular this applies to Description Logic ABoxes. Using ABoxes as state formulas, in an actual implementation we can resort to DL reasoners in order to decide static state knowledge, e.g. action preconditions. Researchers in DL have investigated a great number of DLs of varying strength; from these we can choose a logic that we deem appropriate for the task under consideration.

3.2 Updated ABoxes and State Update Axioms

In a recent paper, a method for updating Description Logic ABoxes has been proposed. Next we will briefly recall essential definitions and results; for in-depth coverage, the interested reader is referred to [5]. Subsequently, we will provide a Fluent Calculus semantics for ABox-Update, and thus relate the latter to a standard AI formalism.

ABox-Update. After introducing the syntactic objects describing an ABox-Update, we restate the semantic considerations underlying the whole approach.

Definition 12 (conditional ABox update). *A conditional update \mathcal{U} is a finite, non-empty set of expressions φ/ψ , where the condition φ is an ABox assertion and the postcondition ψ is a concept/role literal. Consistency of the condition part φ_i for a number of expressions φ_i/ψ_i implies the consistency of their postconditions ψ_i . The condition part may be omitted by writing \top/ψ , where \top abbreviates a tautology.*

The semantics of ABox-Update is defined using the possible models approach of Winslett [11]; that is, for every interpretation \mathcal{I} we define an updated interpretation \mathcal{I}' . E.g., if $\mathcal{U} = \{\varphi_1/C(I_1), \varphi_2/\neg C(I_2)\}$ and \mathcal{I} entails both φ_1 and φ_2 , then

\mathcal{I}' should interpret C as \mathcal{I} , but include the individual I_1 into the interpretation of C and exclude the individual I_2 from it. These should be the only changes to occur. The following definition captures this minimal change policy.

Definition 13 (conditional interpretation update). *Let \mathcal{U} be a conditional update and $\mathcal{I}, \mathcal{I}'$ interpretations such that $\mathfrak{D}^{\mathcal{I}} = \mathfrak{D}^{\mathcal{I}'}$ and \mathcal{I} and \mathcal{I}' agree on the interpretation of individual names. Then \mathcal{I}' is the result of updating \mathcal{I} with \mathcal{U} , written $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$, if the following holds for all concept names $C \in N_C$ and role names $R \in N_R$:*

$$\begin{aligned} C^{\mathcal{I}'} = & (C^{\mathcal{I}} \cup \{ I^{\mathcal{I}} \mid \varphi / C(I) \in \mathcal{U} \wedge \mathcal{I} \models \varphi \}) \\ & \setminus \{ I^{\mathcal{I}} \mid \varphi / \neg C(I) \in \mathcal{U} \wedge \mathcal{I} \models \varphi \} \text{ and} \\ R^{\mathcal{I}'} = & (R^{\mathcal{I}} \cup \{ (I_1^{\mathcal{I}}, I_2^{\mathcal{I}}) \mid \varphi / R(I_1, I_2) \in \mathcal{U} \wedge \mathcal{I} \models \varphi \}) \\ & \setminus \{ (I_1^{\mathcal{I}}, I_2^{\mathcal{I}}) \mid \varphi / \neg R(I_1, I_2) \in \mathcal{U} \wedge \mathcal{I} \models \varphi \}. \end{aligned}$$

Let $\mathcal{M}(\mathcal{A})$ denote the set of all models of an ABox \mathcal{A} .

Definition 14 (updated ABox). *For an ABox \mathcal{A} and a conditional update \mathcal{U} the updated ABox \mathcal{A}' is defined model-theoretically such that:*

$$\mathcal{M}(\mathcal{A}') = \{ \mathcal{I}' \mid \mathcal{I} \in \mathcal{M}(\mathcal{A}) \wedge \mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}' \}.$$

For applying a conditional update \mathcal{U} to an ABox \mathcal{A} resulting in ABox \mathcal{A}' we also write $\mathcal{A}' = \mathcal{A} * \mathcal{U}$. In spite of some negative results in [5] it has been established for a whole range of DLs that these admit ABox-Update; i.e. for arbitrary \mathcal{A} and \mathcal{U} the updated ABox $\mathcal{A}' = \mathcal{A} * \mathcal{U}$ always exists. For the DLs ranging from $\mathcal{ALCCO}^{\textcircled{a}}$ to $\mathcal{ALCQIO}^{\textcircled{a}}$ – which are closely related to the familiar $\mathcal{SHOIN}(D)$ underlying the Ontology Web Language (OWL) – algorithms for computing updated ABoxes have been presented. For an updated ABox $\mathcal{A}' = \mathcal{A} * \mathcal{U}$ there are polynomials p_1, p_2 and q such that

- $|\mathcal{A}'| \leq 2^{p_1(|\mathcal{A}|)} \cdot 2^{p_2(|\mathcal{U}|)}$ and
- \mathcal{A}' is computed in time $q(|\mathcal{A}'|)$.

For repeated updates the final ABox can be exponential only in the size of the original ABox and the total size of all updates.

The authors of [5] also propose two mechanisms for obtaining smaller updated ABoxes; in both cases the result of updating is exponential only in the size of the update. One is based on introducing abbreviations for some complex concepts. The other eliminates the asymmetry between concepts and roles typically found in DLs: it introduces powerful operators on roles. Update algorithms for such DLs are also given; the strongest DL under consideration is as expressive as the two variable fragment of first order logic with counting quantifiers [12].

Fluent Calculus Semantics for ABox-Update. We will now establish a Fluent Calculus semantics for any DL that is both embeddable into first order logic and closed under the above definition of update. To do so, for a given DL, ABoxes $\mathcal{A}, \mathcal{A}'$ and update \mathcal{U} with $\mathcal{A}' = \mathcal{A} * \mathcal{U}$, we will define a corresponding

domain axiomatization Σ in a suitable Fluent Calculus instance. We will then prove that for every model of Σ there are models \mathcal{I} and \mathcal{I}' of \mathcal{A} and \mathcal{A}' satisfying $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$ and vice versa.

First, we associate with \mathcal{U} the name Update. The Fluent Calculus instance is defined such that

- it contains exactly one action, namely Update, and
- there is a bijection between
 - the objects and the individual names N_I , and
 - the function symbols of sort FLUENT and the union of the concept and role names, $N_C \cup N_R$.

Next, since we consider only first order embeddable DLs, we can clearly define a mapping τ_z from ABoxes to state formulas $\Delta(z)$, analogously to the mapping from Definition 11; similarly, τ_s maps ABoxes to situation formulas. In the domain axiomatization Σ to be constructed, let

$$\Sigma_{\text{init}} = \{(\exists z)\text{State}(S_0) = z \wedge \tau_z(\mathcal{A})\}.$$

We now turn to the construction of a state update axiom corresponding to the update $\mathcal{U} = \{\varphi_1/\psi_1, \dots, \varphi_n/\psi_n\}$. Define the set $\mathcal{E}_1 = \{\varphi_i/\psi_i \mid \varphi_i/\psi_i \in \mathcal{U}\} \cup \{\neg\varphi_i/\text{nil} \mid \varphi_i/\psi_i \in \mathcal{U}\}$ and let \mathcal{E}_2 be the set of all subsets of \mathcal{E}_1 that are maximally consistent with regard to the condition part φ_i . Note that \mathcal{E}_2 will be exponential in the size of \mathcal{U} . For every member \mathcal{E}_3 of \mathcal{E}_2 we form an update formula

$$\gamma(s) \stackrel{\text{def}}{=} \Delta(s) \wedge (\exists z)\text{State}(\text{Do}(\text{Update}), s) = \text{State}(s) - \vartheta^- + \vartheta^+$$

where

- $\Delta(s)$ denotes the conjunction of all the situation formulas in the set $\{\tau_s(\varphi) \mid \varphi/\psi \in \mathcal{E}_3 \vee \varphi/\text{nil} \in \mathcal{E}_3\}$, and
- ϑ^+ (respectively, ϑ^-) denotes the finite state consisting of the ground fluents corresponding to the assertions ψ such that $\varphi/\psi \in \mathcal{E}_3$ (respectively, $\varphi/\neg\psi \in \mathcal{E}_3$).⁵

Then Σ contains the single state update axiom

$$\Sigma_{\text{sua}} = \{\text{Poss}(\text{Update}, s) \supset \Gamma(s)\},$$

where $\Gamma(s)$ denotes the disjunction of all the $\gamma(s)$ resulting from the above construction. Observe that all the $\gamma(s)$ are mutually exclusive.

Example 2. Consider the update

$$\mathcal{U} = \{\top/\neg\text{Ordered}(\text{John}, \text{NiceBook}), \text{Paid}(\text{NiceBook})/\text{Refund}(\text{John}, \text{NiceBook})\}.$$

⁵ If there is no such assertion we obtain the empty state \emptyset .

The above construction yields – after a little simplification –

$$\begin{aligned} & \text{Poss}(\text{Update}, s) \supset \\ & \quad (\text{Holds}(\text{Paid}(\text{NiceBook}), s) \wedge \text{State}(\text{Do}(\text{Update}, s)) = \\ & \quad \quad \text{State}(s) - \text{Ordered}(\text{John}, \text{NiceBook}) + \text{Refund}(\text{John}, \text{NiceBook})) \\ & \vee \\ & \quad (\neg \text{Holds}(\text{Paid}(\text{NiceBook}), s) \wedge \text{State}(\text{Do}(\text{Update}, s)) = \\ & \quad \quad \text{State}(s) - \text{Ordered}(\text{John}, \text{NiceBook})). \end{aligned}$$

Finally, we define the set of precondition axioms to be

$$\Sigma_{\text{poss}} = \{\text{Poss}(\text{Update}, s) \equiv \top\},$$

completing the definition of Σ .

Before stating the main theorem, we recall a fundamental result about Fluent Calculus [6] that will be essential to our discussion.

Theorem 2 (fluent calculus foundational theorem). *Let ϑ^+ and ϑ^- be two finite states. Then foundational axioms Σ_{state} together with $z' = z - \vartheta^- + \vartheta^+$ entail*

$$\begin{aligned} & \text{Holds}(f, z') \equiv \text{Holds}(f, \vartheta^+) \\ & \vee \\ & \text{Holds}(f, z) \wedge \neg \text{Holds}(f, \vartheta^-). \end{aligned}$$

Theorem 3 (fluent calculus semantics for ABox-Update). *For an ABox \mathcal{A} , an update \mathcal{U} and the corresponding domain axiomatization Σ it holds that \mathcal{A} has a model \mathcal{I} with $\mathcal{I} \implies_{\mathcal{U}} \mathcal{I}'$ if and only if Σ has a model. Moreover, in a model of Σ , $\text{State}(S_0)$ and $\text{State}(\text{Do}(\text{Update}, S_0))$ relate in the same way as \mathcal{I} and \mathcal{I}' .*

Proof. (\implies)

We will only give a sketch of the proof. As in the proof of Theorem 1 we can restrict our attention to Henkin-style term interpretations: When constructing $\tau_z(\mathcal{A})$ we simultaneously construct the first order representation of \mathcal{A} , using the same variable names. A term model of this is readily turned into a model of \mathcal{A} . We then interpret the objects by their equivalence classes, and fluents by fluent terms built from these equivalence classes as in the proof of Theorem 1. We extend this treatment to situations and actions: here we restrict the respective universes to the set of ground situations and actions built using only terms of sort OBJECT occurring in an equivalence class. Interpreting \circ and \emptyset as set union and empty set as before, we fix the interpretation of $\text{State}(S_0)$ as the set of fluents corresponding to atoms that are true in \mathcal{I} . We observe that, once we have fixed the interpretation of $\text{State}(S_0)$, the model of Σ is uniquely determined, due to Theorem 2 and the fact that the conditions $\Delta(s)$ in the state update axiom are mutually exclusive. Theorem 2 is also the key to proving that $\text{State}(S_0)$ and $\text{State}(\text{Do}(\text{Update}, S_0))$ are related in the same way as \mathcal{I} and \mathcal{I}' .

(\Leftarrow)

Let $\mathcal{M}_2 = (\mathfrak{D}_{\text{object}}, \mathfrak{D}_{\text{fluent}}, \mathfrak{D}_{\text{state}}, \mathfrak{D}_{\text{situation}}, \mathfrak{D}_{\text{action}}, \cdot^{\mathcal{M}_2}) \models \Sigma$.

Set $\mathcal{I}_3 = (\mathfrak{D}_{\text{object}}, \cdot^{\mathcal{I}_3})$, $\mathcal{I}_4 = (\mathfrak{D}_{\text{object}}, \cdot^{\mathcal{I}_4})$ where

- $P_i^{\mathcal{I}_3} = \{(\mathbf{t}^{\mathcal{M}_2}) \mid \mathcal{M}_2 \models \text{Holds}(F_i(\mathbf{t}), \text{State}(S_0))\}$,
- $P_i^{\mathcal{I}_4} = \{(\mathbf{t}^{\mathcal{M}_2}) \mid \mathcal{M}_2 \models \text{Holds}(F_i(\mathbf{t}), \text{State}(\text{Do}(\text{Update}, S_0)))\}$ and
- $\cdot^{\mathcal{I}_3}$, $\cdot^{\mathcal{I}_4}$ and $\cdot^{\mathcal{M}_2}$ agree on sort OBJECT.

Then $\mathcal{I}_3 \models \mathcal{A}$ and $\mathcal{I}_3 \Rightarrow_u \mathcal{I}_4$. □

Figure 2 depicts the relationship just established.

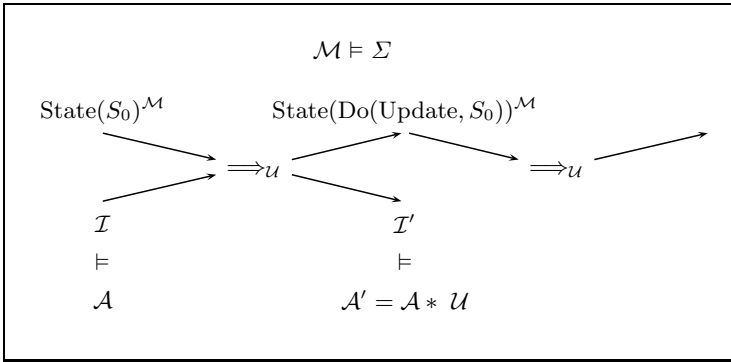


Fig. 2. Fluent Calculus semantics for ABox-Update

This result has two important consequences: On the one hand, by establishing a Fluent Calculus semantics for ABox-Update, it relates the latter to an established, general action formalism. On the other hand, it provides the formal underpinnings of using the update algorithms of [5] for computing updated states in a Fluent Calculus that uses ABoxes as state descriptions. Using an accordingly restricted Fluent Calculus instead of plain ABox-Update the notion of update resides within the language instead of being meta-logical.

3.3 TBoxes and Domain Constraints

The reader already familiar with Description Logics may wonder why we have not yet mentioned TBoxes. By allowing the definition of concepts in terms of other concepts, these contribute considerably to the expressive power of DLs.

Definition 15 (TBox/knowledge base). $C \equiv D$ is a concept definition, where C is a defined concept name and D is a complex concept. A TBox \mathcal{T} is a finite set of concept definitions. An interpretation \mathcal{I} satisfies a concept definition $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. \mathcal{I} satisfies a TBox \mathcal{T} , if it satisfies all concept definitions in \mathcal{T} . A Knowledge Base is a pair $KB = (\mathcal{T}, \mathcal{A})$, with TBox \mathcal{T} and ABox \mathcal{A} .

A TBox \mathcal{T} is a *terminology* if every defined concept is defined only once. A defined concept name C *directly uses* a concept name D if D occurs on the right hand side of the concept definition. A terminology is acyclic if no concept name is connected with itself via the transitive closure of *directly uses*. Reasoning in a knowledge base $KB = (\mathcal{T}, \mathcal{A})$, where \mathcal{T} is an acyclic terminology, can always be reduced to reasoning wrt. the empty TBox by unfolding the definitions [4].

For example, in our online-store scenario we can introduce the concept of a good customer with the help of the TBox

$$\{\text{GoodCustomer} \equiv \text{PurchasedManyItems} \sqcap \text{PaidOnTime}\}.$$

In action formalisms the concept of a domain constraint allows to state general knowledge and laws that have to be satisfied by every world state.

Definition 16 (domain constraint). *In Fluent Calculus a domain constraint is a formula of the form $(\forall s)\Delta(s)$, where $\Delta(s)$ is a situation formula.*

TBoxes are captured neatly by appropriate domain constraints. E.g. we map the above TBox to

$$\begin{aligned} (\forall s. \forall x) \text{Holds}(\text{GoodCustomer}(x), s) \equiv \\ \text{Holds}(\text{PurchasedManyItems}(x), s) \wedge \text{Holds}(\text{PaidOnTime}(x), s). \end{aligned}$$

It is trivial, but potentially useful, to admit acyclic TBoxes. We can faithfully apply the update algorithms from [5] to an ABox serving as world state description after unfolding the TBox, resulting in a potentially exponential blowup. However, in the ABoxes that serve as action preconditions in a domain axiomatization, we can admit defined concepts without unfolding them into the ABox. This is possible since the semantics of the undefined concepts uniquely determines the semantics of the defined ones. The above result on the semantic correspondence between ABox-Update and Fluent Calculus state update axioms can be extended to take acyclic TBoxes into account.

If we admit general TBoxes, semantic problems arise. The semantics of the undefined concepts no longer uniquely determines the semantics of the TBox. As a consequence the one-to-one relation between original and updated interpretation – that is at the heart of ABox-Update – can not be maintained. This issue is well known to researchers in action formalisms as the Ramification Problem [13]. Considerable effort went into singling out intended interpretations, usually by appealing to some notion of causality [14,15,16]. This work should prove helpful when extending the definition of interpretation update.

4 Summary

4.1 Related Work

Recently, a number of works have addressed the issue of finding a decidable yet expressive logical framework for reasoning about actions and change. In the

following we will relate our work to other DL-based approaches. Such approaches have continued to attract considerable interest, not least since Description Logics form the foundation of the Semantic Web, and a dynamic view of the web is intuitively very appealing.

In [17] de Giacomo et al. show that DL-Lite is closed under update in the above sense; they also present a polynomial algorithm for computing updated ABoxes. The Description Logic DL-Lite is of reduced expressivity, but admits tractable reasoning and updated ABoxes of polynomial size. They also address updates in the presence of general TBoxes. If the models of the update and the general TBox have an empty intersection their algorithm guarantees correctness; otherwise it returns with an error. Our framework can also be instantiated with DL-Lite ABoxes; returning an error is not an option for an autonomous agent.

Liu et al. [18] provide an in-depth discussion of the semantic problems that arise when updating ABoxes in the presence of general TBoxes. They also observe that these problems are closely related to the ramification problem. As a solution they propose to provide the domain axiomatizer with a syntactic means to indicate which assertions may fluctuate freely during the update.

Baader et al. [19] is another work on DL-based reasoning about action and change. They employ reasoning similar to regression and among many other results, they outline how their work can be regarded as an instance of the Situation Calculus. Gu and Soutchanski [20] directly define a modified Situation Calculus, based on a DL with role operators that is equally expressive as \mathcal{C}^2 . They adapt regression from the general Situation Calculus to their setting extended with acyclic TBoxes. They address the problem of using progression, i.e. update, instead of regression in [21]. To this end, since fluents are not reified in the Situation Calculus, they have to appeal to second order logic. An in-depth comparison of their work will be subject of future work. The fact that Situation Calculus and Fluent Calculus semantically agree has been shown in [22].

Employing existing DL reasoners we have to start reasoning from scratch after each update. In [23] the problem of incremental maintenance of a solver state is addressed under a very simple semantics for ABox-Update. It would be nice to extend these ideas to updates under the possible models approach.

4.2 Conclusion

We have shown how to integrate Description Logics into a general action formalism. We have thus restricted the latter to a decidable, yet expressive fragment of classical first order logic. To do so, we have proved that ABoxes can serve as a faithful substitute for state formulas in Fluent Calculus. Moreover, by proving that Fluent Calculus state update axioms correctly capture ABox-Update, we have related the latter to established research in reasoning about action and change. Our work lays the theoretical foundations for an integration of DL reasoning and update algorithms into a practical agent programming language. There are a number of interesting open issues for future work:

- Applying existing solutions to the ramification problem to handle ABox-Update in the presence of general TBoxes.

- Integrating inference algorithms for Description Logic problems into a general action programming language, like e.g. FLUX [2](#).

References

1. McCarthy, J.: Situations, actions, and causal laws. Technical Report AIM-2, AI Project, Stanford University (1963)
2. Thielscher, M.: FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming* 5, 533–565 (2005)
3. Lespérance, Y., Levesque, H.J., Lin, F.D., Marcu, R.R., Scherl, R.B.: A logical approach to high-level robot programming—A progress report. In: *Papers from the 1994 AAAI Fall Symposium*, AAAI (1994)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge (2003)
5. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) *KR*, pp. 46–56. AAAI Press (2006)
6. Thielscher, M.: *Reasoning Robots: The Art and Science of Programming Robotic Agents*. Applied Logic Series, vol. 33. Kluwer Academic Publishers, Dordrecht (2005)
7. Sirin, E., Parsia, B.: Pellet: An OWL DL reasoner. In: *Proceedings of the 2004 International Workshop on Description Logics (DL2004)* (2004)
8. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Furbach, U., Shankar, N. (eds.) *IJCAR 2006*. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
9. Haarslev, V., Möller, R.: Racer system description. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) *IJCAR 2001*. LNCS (LNAI), vol. 2083, pp. 701–705. Springer, Heidelberg (2001)
10. Enderton, H.B.: *A Mathematical Introduction to Logic*. Academic Press, London (1972)
11. Winslett, M.: Reasoning about action using a possible models approach. In: *aaai88*, pp. 89–93 (1988)
12. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language, and Information* 14, 369–395 (2005)
13. Ginsberg, M.L., Smith, D.E.: Reasoning about action II: the qualification problem. *Artificial Intelligence* 35, 311 (1988)
14. Lin, F.: Embracing causality in specifying the indirect effects of actions. In: Mellish, C.S. (ed.) *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Canada, pp. 1985–1991. Morgan Kaufmann, San Francisco (1995)
15. Thielscher, M.: Ramification and causality. *Artificial Intelligence Journal* 89, 317–364 (1997)
16. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. *Artificial Intelligence* 153, 49–104 (2004)
17. Giacomo, G.D., Lenzerini, M., Poggi, A., Rosati, R.: On the update of description logic ontologies at the instance level. In: *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI 2006)* (2006)
18. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Description logic actions with general TBoxes: a pragmatic approach. In: *Proceedings of the 2006 International Workshop on Description Logics (DL2006)* (2006)

19. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Proceedings of AAAI-05 (2005)
20. Gu, Y., Soutchanski, M.: A logic for decidable reasoning about services. In: Proceedings of the 4th International Workshop on AI for Service Composition (ECAI 2006) (2006)
21. Gu, Y., Soutchanski, M.: Decidable reasoning in a modified situation calculus. In: Proceedings of International Joint Conference on AI (IJCAI 2007) (2007)
22. Schiffel, S., Thielscher, M.: Reconciling situation calculus and fluent calculus. In: Proceedings of AAAI-06, Boston, MA, pp. 287–292. AAAI Press (2006)
23. Halaschek-Wiener, C., Parsia, B., Sirin, E., Kalyanpur, A.: Description logic reasoning for dynamic aboxes. In: Proceedings of the 2006 International Workshop on Description Logics (DL2006) (2006)

Any-World Access to OWL from Prolog

Tobias Matzner and Pascal Hitzler

Institute AIFB, Universität Karlsruhe, Germany
{tobias.matzner, hitzler}@aifb.uni-karlsruhe.de

Abstract. The W3C standard OWL provides a decidable language for representing ontologies. While its use is rapidly spreading, efforts are being made by researchers worldwide to augment OWL with additional expressive features or by interlacing it with other forms of knowledge representation, in order to make it applicable for even further purposes. In this paper, we integrate OWL with one of the most successful and most widely used forms of knowledge representation, namely Prolog, and present a hybrid approach which layers Prolog on top of OWL in such a way that the open-world semantics of OWL becomes directly accessible within the Prolog system.

1 Introduction

The Web Ontology Language OWL has been recommended by the W3C in 2004 for the representation of ontologies, and its usage is spreading rapidly ever since. One of the design issues for OWL has been that it is decidable and based on the open world assumption, and these two properties – which are both inherited from description logics – have served it well in the last two years.

However, with these design decisions come also some drawbacks as they limit expressiveness of OWL in ways which make working with it cumbersome at times. Even more, due to decidability of the language some things cannot be expressed at all in OWL. Efforts are therefore under way to extend OWL with more expressive features, and there is a growing body of work with proposals and studies how to do this best.

The corresponding research can roughly be classified into two different approaches. The first approach deals with extensions of OWL while adhering as much as possible to the conceptual frame of mind spanned by description logic research. The second approach is based on establishing hybrid systems which combine OWL with other established knowledge representation formalisms in such a way that either approach is encompassed in full, possibly using two different reasoning engines, but allowing for information flow between the subsystems. The work which we present in this paper is of the hybrid kind.

The particular integration which we report on, is based on the following rationales.

- OWL has not been designed to be a stand-alone programming language. OWL ontologies should rather be viewed as declarative knowledge bases,

which require programming in some other language for accessing the knowledge and further processing it. It is a natural choice to use a logic-based declarative programming language for this purpose.

- One of the most requested-for extensions of OWL is the ability to formulate rules, in some established rules language.
- It becomes more and more apparent that closed-world features are required alongside the open-world character of OWL.

Our hybrid system addresses the formulated needs by interlacing OWL with one of the most prominent and historic approaches to logic-based knowledge representation, namely with Prolog. Our system layers Prolog on top of OWL by allowing the querying of OWL ontologies via a standard OWL reasoner. A tight integration is achieved by interpreting the answers given by the OWL reasoner in an open-world fashion, and by processing this answer within Prolog in the same open-world fashion. This is achieved by means of the so-called any-world semantics due to Loyer and Straccia [1].

Technically speaking, the integration is achieved via a hybrid semantics for a language which incorporates calls to an OWL reasoner into standard logic programming. This hybrid semantics is based on the any-world semantics. Algorithmization and an implementation of the approach is provided by means of a transformation of logic programs under the any-world semantics into standard Prolog, in this case realised using SWI-Prolog.

Besides the aforementioned rationales for our approach, we thus arrive at a system with the following additional features.

- Modularity: The user can develop its programs based on Prolog programming and need not deal with the evaluation of OWL-based reasoning and knowledge. It is possible to offer restricted or controlled access to third party knowledge-bases without problems.
- Maturity: We incorporate the KAON2 reasoner and thus offer the performance of a state-of-the-art DL-reasoner to the logic programming world. The logic programming environment can be handled with little more than basic Prolog knowledge.
- Conformity with standards: Available OWL knowledge bases can be used directly. As we do not need one big formal system comprising both approaches, these can be used with no or only little maintenance to do.
- Bridge between ontology language paradigms: One of the most prominent alternatives to OWL for ontology representation is F-Logic [2,3], which can be used both as an ontology language and as a programming language. As F-Logic in its basic form is basically Prolog extended with further syntactic features, our approach can be used directly for realising a hybrid OWL/F-Logic system.

The structure of the paper is as follows. In Section 2, we review the basic facts we need about the any-world semantics and about OWL in order to make this paper relatively self-contained. In Section 3, we prove a theorem which gives the formal rationale for our algorithmisation. In Section 4 we discuss the

implemented system which we provide. In Section 5 we give an extended example which shows the possibilities of our approach. In Section 6 we discuss related work, and we conclude in Section 7.

2 Preliminaries

2.1 The Any-World Semantics

We review the any-world semantics due to Loyer and Straccia [1] in some details as it is crucial for understanding our work.

Bilattices. The any-world semantics is based on a truth-space which is a so-called bilattice [4]. This is a potent mathematical structure which particularly provides two partial orders, which permit to represent (logical) truth and the knowledge contained in these truth-values separately.

Formally, a *lattice* $\langle L, \leq \rangle$ is a non-empty set L with a partial order \leq , where each subset of L containing two elements has a supremum and infimum regarding \leq (also known as *meet* and *join*). It is a *complete lattice* iff every subset has supremum and infimum regarding \leq . We write $x < y$ for $x \leq y$ and $x \neq y$ where $x, y \in L$.

A *bilattice* $\langle B, \leq_t, \leq_k \rangle$ is a non-empty set B with two partial orders, the *truth-order* \leq_t and the *knowledge-order* \leq_k , both of which give B the structure of a complete lattice. Due to completeness, the greatest and least element regarding either of the orders always exists and is unique [4]. The greatest element regarding \leq_t is denoted **true**, the least element **false**. Regarding \leq_k , the greatest element is \top , the least \perp . Meet and join under \leq_t which are denoted \wedge and \vee , correspond to the well-known two-valued conjunction and disjunction regarding the values **true** and **false**. Under \leq_k meet and join are denoted \otimes and \oplus , where $x \otimes y$ extracts the maximum knowledge that is expressed both in x and y whereas $x \oplus y$ unites the knowledge of x and y . Our approach is particularly based on the smallest non-trivial bilattice known as *FOUR* [5] which is depicted in Figure 1. Indeed, although bilattice-based semantics is generally formulated for arbitrary bilattices, *FOUR* is currently the only such lattice of practical relevance, and will entirely suffice for our purposes.

An operator \bullet on a lattice is called *monotone* when $x_1 \leq y_1$ and $x_2 \leq y_2$ implies $x_1 \bullet x_2 \leq y_1 \bullet y_2$. We suppose for all bilattices here considered that all of the operators $\wedge, \vee, \otimes, \oplus$ are monotone w.r.t. both the knowledge- and the truth-order; this is called the *infinitary interlacing condition*. We furthermore assume that all bilattices are *infinitary distributive* i.e. that all distributive laws connecting the aforementioned lattice operators hold. Finally, we assume that all lattices have a *negation*, which is an operator denoted \neg that inverses the truth order, does not inflict the knowledge order and satisfies $\neg\neg x = x$. These assumptions are standard and generally known to be unproblematic in a logic programming context.

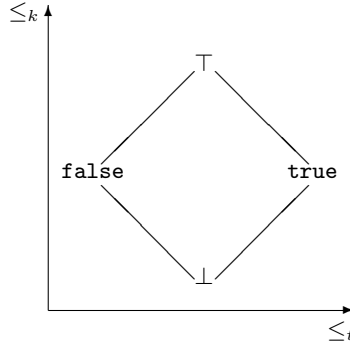


Fig. 1. The bilattice *FOUR*

Logic programs. We extend logic programs from the common case and include not only connectives for disjunction, conjunction and negation but for all the operators of a bilattice: $\wedge, \vee, \otimes, \oplus$ and \neg . So the knowledge order and its operators are not only a tool of analysis and semantics as used for example in [6] but can be used explicitly to determine how the program treats information from the perspective of knowledge. A logic program is based on a set \mathcal{P} of predicates, \mathcal{V} of variables, \mathcal{C} of constants and \mathcal{F} of functions. A *term* is either an element of \mathcal{V} or \mathcal{C} or of the form $f(t_1, \dots, t_n)$ where $f \in \mathcal{F}$ and all t_1, \dots, t_n are terms. The *ground terms* forming the *Herbrand universe* are all the terms that can be built from elements of \mathcal{C} and \mathcal{F} . An *atom* is of the form $p(t_1, \dots, t_m)$ where $p \in \mathcal{P}$ and all t_1, \dots, t_m are terms. The *ground atoms* forming the *Herbrand base* are all the atoms that can be built from the Herbrand universe. A *literal* is of the form A or $\neg A$ where A is an atom. Furthermore we allow the elements of the bilattice as literals. A *formula* is either any literal, or of the form $\varphi_1 \bullet \varphi_2$ where φ_1 and φ_2 are formulas and \bullet is one of the four lattice operators $\wedge, \vee, \otimes, \oplus$, or one of the expressions $\forall\varphi$ respectively $\exists\varphi$ where φ is a formula. A *rule* is of the form $p(x_1, \dots, x_m) \leftarrow \varphi(x_1, \dots, x_m)$ where $p \in \mathcal{P}$, $x_1, \dots, x_m \in \mathcal{V}$ and φ is a formula. We call p the *head* and φ the *body* of the rule. We suppose that the free variables in φ are among $\{x_1, \dots, x_m\}$ and are universally quantified. A *logic program* P is a finite set of rules. Not allowing terms in the heads of rules is not a restriction, e.g. the rules (taken from [1]):

$$\begin{aligned} p(s(x)) &\leftarrow p(x) \\ p(0) &\leftarrow \text{true} \end{aligned}$$

can be rewritten (using a predicate *eq* defining equality) as:

$$\begin{aligned} p(y) &\leftarrow \exists x (eq(y, s(x)) \wedge p(x)) \\ p(x) &\leftarrow eq(x, 0) \end{aligned}$$

With $ground(P)$ we denote all ground instances of members of P over the Herbrand universe.

Interpretations of logic programs. Let B be a bilattice. An *interpretation* of a logic program on B is a mapping I from ground atoms to members of B . It is extended to formulas as follows: $I(b) = b$ where $b \in B$; $I(\varphi_1 \bullet \varphi_2) = I(\varphi_1) \bullet I(\varphi_2)$ where φ_1, φ_2 are formulas and \bullet is one of the operators $\wedge, \vee, \otimes, \oplus$; $I(\neg\varphi) = \neg I(\varphi)$; $I(\exists x\varphi(x)) = \bigvee\{I(\varphi(t)) \mid t \text{ is a ground term}\}$ and finally $I(\forall x\varphi(x)) = \bigwedge\{I(\varphi(t)) \mid t \text{ is a ground term}\}$. The partial orders of the bilattice are point-wise extended to interpretations: $I_1 \leq_t I_2$ iff $I_1(A) \leq_t I_2(A)$ for all ground atoms A . The extension for \leq_k is analogous. Given two interpretations I_1, I_2 we define $(I_1 \bullet I_2)(\varphi) = I_1(\varphi) \bullet I_2(\varphi)$ where \bullet is a lattice operator and φ a formula. Thus the space of all possible interpretations on a bilattice constitutes an infinitary interlaced and distributive bilattice as well. An interpretation I is a *model* of a logic program P iff $I(A) = I(\varphi)$ for all rules $A \leftarrow \varphi$ in P .

Semantics. The semantics is defined via the fixed point of a monotone operator similarly to the well known Kripke-Kleene [7,8] or well-founded [9] semantics. In fact the any-world semantics used here is a generalization of the well-founded semantics. The central idea of the any-world semantics is to overcome the limitations of both the open and the closed world as default assumption. Instead an arbitrary interpretation H called the *hypothesis* is used as default assumption, i.e. the value $H(A)$ is the default value for the atom A . From this point of view, the open world assumption corresponds to the hypothesis $H(A) = \perp$ for all atoms A , we call this hypothesis H_\perp . The closed world assumption can be modelled by $H(A) = \mathbf{false}$ for all atoms A , this hypothesis is denoted $H_{\mathbf{f}}$. Now the information of the program is combined with knowledge extracted from the hypothesis used. To gather information from the program we use the well known *immediate consequence operator* $\Phi_P(I)(A) = I(\varphi)$ where $A \leftarrow \varphi$ is a rule in P . Now we want to augment the interpretation I with the information from a hypothesis H . This is done similarly to the use of the unfounded set in the well-founded semantics. From a knowledge point of view, the unfounded set is the amount of information contributed to the semantics by the closed world assumption. This concept now is generalized to arbitrary hypotheses H . We usually cannot use all the information of H . Instead we want to extract the maximum knowledge of H , expressed as an interpretation J , so that the assumed knowledge J is entailed by the program w.r.t. the augmented interpretation $I \oplus J$, i.e. we want to make sure that $J(A) \leq_k \Phi_P(I \oplus J)(A)$. This idea is modelled using the so called *safe interpretations*. An interpretation J is *safe* w.r.t. a logic program P , an interpretation I and a hypothesis H if $J \leq_k H$ and $J \leq_k \Phi_P(I \oplus J)$. The *support* provided by H to P and I is the greatest (on the knowledge order) safe interpretation w.r.t. P, I and H . It is denoted $s_P^H(I)$. Note that this particularly entails that the support is always smaller than the hypothesis.

In order to simplify the treatment of logic programs using fixed-point semantics, we introduce the transformed program P^* . Given a logic program P and a hypothesis H the program P^* contains the following rules:

- $A \leftarrow \varphi_1 \vee \dots \vee \varphi_n$ if $A \leftarrow \varphi_1, \dots, A \leftarrow \varphi_n$ are all rules in $ground(P)$ with the head A .
- $A \leftarrow H(A)$ if A is not the head of any rule in P .

The second part enforces that for any atom that is not assigned a truth-value by a rule in the program, it is given its value according to the default assumption, i.e. the hypothesis.

Now we define the operator $\tilde{\Pi}_P^H(I) = \Phi_P(I) \oplus s_P^H(I)$ which works on P^* . The fixed points of $\tilde{\Pi}_P^H$ are called the *H-founded models* of P . In [1] it is shown that the support operator $s_P^H(I)$ is monotone in I and H w.r.t. the knowledge order. Furthermore also Φ_P is monotone w.r.t. the knowledge order [6]. By the infinitary interlacing condition, monotonicity of $\tilde{\Pi}_P^H$ is guaranteed. So by the well known Knaster-Tarski theorem [10], there is always a (unique) least *H-founded* model for any logic program, which can be obtained as the least upper bound of the transfinite sequence $(\tilde{\Pi}_P^H \uparrow \alpha)_\alpha$, where α ranges over ordinals, $\tilde{\Pi}_P^H \uparrow 0$ is the least interpretation, $\tilde{\Pi}_P^H \uparrow \alpha + 1 = \tilde{\Pi}_P^H(\tilde{\Pi}_P^H \uparrow \alpha)$ for all α , and $\tilde{\Pi}_P^H \uparrow \alpha = \sup\{\tilde{\Pi}_P^H \uparrow \beta : \beta < \alpha\}$ for limit ordinals α .

The key feature of the any-world semantics is the flexibility of the default assumption. Particularly using a hypothesis that maps ground atoms to the set $\{\mathbf{false}, \perp\}$ it is possible to mix closed- and open-world based information, whereon our hybrid semantics relies. It also includes several well known semantics. Using H_f , the *H-founded* model is the well-founded model [1]. Let H_{KK} be the interpretation that maps all atoms that are the head of a rule in a given program P to \perp , all the other atoms to \mathbf{false} . Using this hypothesis, the *H-founded* model of P is its Kripke-Kleene-model [1]. This reflects that the Kripke-Kleene semantics uses only the immediate-consequence operator Φ_P and consequently the support part in $\tilde{\Pi}_P^H$ is reduced to \perp by the assignment of \perp to all rule heads. (Recall that the support is always smaller than the hypothesis on the knowledge order). However the Kripke-Kleene semantics is based on the closed world assumption. This is manifested in the hypothesis mapping the other atoms to \mathbf{false} . The fact that the hypothesis affects only those atoms will be used later for the hybrid semantics of our system.

2.2 Description Logics

The description logics part of our hybrid system uses the KAON2 OWL DL reasoner [11]. Our approach, however, is independent of the specific reasoner used, and can indeed be used with any reasoning system based on the open world assumption. OWL DL is based on the description logic *SHOIN(D)* [12], but for the purpose of our exhibition we will not need to give many details about OWL DL. It shall suffice to recall that OWL DL allows to specify axioms describing the subsumption relation between complex concepts C and D , written $C \sqsubseteq D$. The (complex) concepts themselves are composed by means of primitive (or atomic) concepts, logical and other connectives, individuals which correspond to logical constants, and roles which describe relationships between individuals. It

¹ See also <http://kaon2.semanticweb.org>

is also possible to specify that some individual a belongs to a class C , written $C(a)$, or to explicitly state that two individuals a and b are connected by a role R , written $R(a, b)$. The special concepts \top and \perp , respectively, are defined as containing all individuals respectively no individual. OWL DL is given an open-world semantics e.g. by mapping it into first-order logic with equality.

Given a set of OWL DL axioms, called an *ontology*, it is possible to derive logical consequences from it by means of well-established algorithms. The most basic inference tasks are

- checking whether an ontology is satisfiable (i.e. logically consistent),
- checking whether a concept C subsumes a concept D , i.e. whether $C \sqsubseteq D$ is a logical consequence,
- checking whether a concept C is satisfiable, i.e. whether there is a model of the knowledge base in which the extension of C is non-empty, and
- checking whether an individual a is contained in a concept C , i.e. whether $C(a)$ is a logical consequence.

3 A Program Transformation for Algorithmising the Any-World Semantics

We provide an extension for Prolog which implements an any-world logic based on *FOUR* and hypotheses that map into $\{\mathbf{false}, \perp\}$. The implementation is based on Theorem 1 below, which acts as a bridge between the any-world semantics and Prolog.

Before we provide the theorem, let us define the specific type of hypotheses which we need for our purposes. Recall that the hypothesis $H_{\mathbf{f}}$ corresponds to the closed world assumption, while H_{\perp} can be interpreted as an open world semantics. Consequently, the hypotheses of interest are a mix between these two.

Definition 1. *Given a logic program P we define the set of hypotheses KKS to be the set of all interpretations that map an atom A to \perp when A is the head of a rule in P , and to either \perp or \mathbf{false} otherwise.*

Note that H_{KK} is in KKS for all programs P .

Theorem 1. *Given a logic program P and a hypothesis $H_{\text{KKS}} \in \text{KKS}$, there exists a program transformation $T^{H_{\text{KKS}}}$ such that the H -founded model of P under H_{KKS} is the same as the H -founded model of $T^{H_{\text{KKS}}}(P)$ under H_{KK} .*

The proof is based on the possibility to add the default assumption chosen as rules of the form $A \leftarrow H(A)$ to the program, such that the resulting program does not have any atoms that are not head of a rule. When evaluated under H_{KK} , accordingly the default assumption \mathbf{false} is not used for any atom. The assumption \perp for atoms that are heads of a rule, i.e. all atoms, is overridden by the rule $A \leftarrow \mathbf{false}$ should it exist, as $\mathbf{false} \oplus \perp = \mathbf{false}$. We first prove the following:

Lemma 1. *Let P be a logic program and H_{KKS} a hypothesis from KKS. Then $s_P^{H_{KKS}}(I) = H_{KKS}$ for any interpretation I .*

Proof. For the following proof we write $s_P^{H_{KKS}}(A)$ for $s_P^{H_{KKS}}(I)(A)$ as the choice of interpretation is without effect. For an atom A that is not the head of any rule there are two possibilities: (1) If $H_{KKS}(A) = \perp$, then the fact that the support is always smaller than the hypothesis on the knowledge-order requires $s_P^{H_{KKS}}(A) = \perp$. (2) If $H_{KKS}(A) = \mathbf{false}$, then the rule $A \leftarrow \mathbf{false}$ is in P^* . As the support is a safe interpretation we require $s_P^{H_{KKS}}(A) \leq_k \Phi_P(I \oplus s_P^{H_{KKS}})(A)$. This now becomes $s_P^{H_{KKS}}(A) \leq_k I(\mathbf{false}) \oplus s_P^{H_{KKS}}(\mathbf{false}) = \mathbf{false}$. As the support is the largest safe interpretation on the knowledge-order we have $s_P^{H_{KKS}}(A) = \mathbf{false}$. Consider now an atom A that is head of a rule in P . Using again that the support is a safe interpretation and thus smaller (in the knowledge order) than the hypothesis, we have that $s_P^{H_{KKS}}(A) = \perp = H_{KKS}$. \square

Now we are ready to prove the theorem:

Proof (of Theorem 7). We use the notation of the theorem. Furthermore we let $P' = T^{H_{KKS}}(P)$. We now show that the operators $\tilde{\Pi}_P^{H_{KKS}}$ and $\tilde{\Pi}_{P'}^{H_{KK}}$ have the same result on every step of their iteration. As the operator $\tilde{\Pi}$ is defined on P^* , for the evaluation of $\tilde{\Pi}_P^{H_{KKS}}$ P^* is constructed from P under the hypothesis H_{KKS} . In this process the same rules are added as when applying $T^{H_{KKS}}$ to P by definition of T . So P' and P^* constructed under the hypothesis H_{KKS} are identical. For the evaluation of $\tilde{\Pi}_{P'}^{H_{KK}}$ the program P'^* is constructed under the hypothesis H_{KK} . Since in P' all atoms are head of rule, the hypothesis H_{KK} has no influence on P'^* . So we have $P^* = P'^*$, thus $\tilde{\Pi}_P^{H_{KKS}}$ and $\tilde{\Pi}_{P'}^{H_{KK}}$ work on the same program.

Now consider an arbitrary iteration step α :

$$(\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha)(\varphi) \oplus s_{P'}^{H_{KK}}(\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha)(A).$$

We have $H_{KK}(A) = \perp$ for all atoms A in P' as all atoms are the head of a rule after the transformation. By Lemma 1 the support is equal to the hypothesis (note that H_{KK} is in KKS) and so the remaining formula is

$$(\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_{P'}^{H_{KK}} \uparrow \alpha)(\varphi). \quad (1)$$

Consider now the operator

$$(\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha)(\varphi) \oplus s_P^{H_{KKS}}(\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha)(A).$$

Again by Lemma 1 we have that $H^{H_{KKS}}(A) = s_P^{H_{KKS}}(\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha)(A)$. For atoms that are head of a rule in P we obtain

$$(\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha + 1)(A) = (\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha)(\varphi). \quad (2)$$

As P and P' contain the same rules, the operators in (1) and (2) yield the same results. For atoms that are not the head of a rule we know that either

$H_{KKS}(A) = \perp$ or $H_{KKS}(A) = \mathbf{false}$. The following argument is analogous for both cases, we consider the latter. If $H_{KKS}(A) = \mathbf{false}$, then there is a rule $A \leftarrow \mathbf{false}$ in P^* and accordingly also in P' . So we have $(\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha)(\varphi) = s_P^{H_{KKS}}(\tilde{\Pi}_P^{H_{KKS}} \uparrow \alpha)(A) = \mathbf{false}$ as well as $(\tilde{\Pi}_{P'}^{H_{KKS}} \uparrow \alpha)(\varphi) = \mathbf{false}$. So both operators give the same result. \square

So we have the possibility to deal with hypotheses mixing open- and closed-world assumption while we only need to compute the Kripke-Kleene semantics.

4 Implementation

In order to arrive at its least fixed point, i.e. at the Kripke-Kleene semantics, Φ_P may need as many as Church-Kleene ω_1 steps. Indeed the Kripke-Kleene semantics is Π_1^1 -complete [13] and thus not even semi-decidable. This means that a sound and complete implementation of the Kripke-Kleene semantics cannot be provided for theoretical reasons.

However, the Kripke-Kleene semantics was originally conceived as a declarative semantics which captures the essence of the Prolog procedural semantics, and indeed they are strongly related, as shown e.g. in [14]. For practical purposes, it thus suffices to view Prolog as an approximate implementation of the Kripke-Kleene semantics.

We therefore provide a library that permits using the logic *FOUR* with all corresponding lattice operations $\oplus, \otimes, \wedge, \vee$ and \neg in Prolog. The user can write programs in a Prolog-like syntax, which is then compiled to SWI-Prolog² such that each predicate is augmented with an additional parameter, which carries the truth-value. A predicate $p(t_1, \dots, t_n, TV)$ is then deducible in Prolog if $p(t_1, \dots, t_n)$ has the truth-value TV .

Within this framework, we offer special atoms, so called *DL-atoms*, that are not evaluated according to the logic programming semantics but by querying the DL-reasoner KAON2. They have the form $dl_q(p_q)$ where q is a query and p_q the respective vector of parameters. The queries we offer are *subsumes*, *unsatisfiable* and *disjoint* regarding concepts and *has_role* regarding roles.

Usually queries to a DL-reasoner have two possible answers: the queried information is either demonstrable or not. However, if the answer is negative, then two cases are possible: Either the negation of the query is demonstrable, or the negation of the query is also not demonstrable. In the first case, the refutation of the query is much stronger than in the second.

In order to give an example, consider the knowledge base specified by the following axioms.

$$\begin{aligned} unicorn &\sqsubseteq \text{appears_in_novels} \\ horned_animal &\sqsubseteq \text{animal} \end{aligned}$$

When queried whether $unicorn \sqsubseteq horned_animal$ holds, the reasoner responds with **No**, which is entirely appropriate as the knowledge base does not allow to de-

² <http://www.swi-prolog.org>

rive any knowledge about the relationship between *unicorn* and *horned_animal*, i.e. the relationship $unicorn \sqsubseteq horned_animal$ can neither be confirmed nor refuted.

Consider now the situation that the knowledge base contains the following additional axioms, where the second describes the assertion that the concepts *unicorn* and *phantasy_animal* are extensionally disjoint.

$$\begin{aligned} unicorn &\sqsubseteq phantasy_animal \\ animal \sqcap phantasy_animal &\sqsubseteq \perp \end{aligned}$$

When now queried whether $unicorn \sqsubseteq horned_animal$ holds, the reasoner again responds with **No**, which is entirely appropriate as the knowledge base implies that *unicorn* and *horned_animal* are in fact extensionally disjoint. The situation compared to the first situation, however, is very different: The first knowledge base did not specify anything about the relation between *unicorn* and *horned_animal*, while the second knowledge base strongly refutes the subsumption relation.

Our framework provides the means to distinguish between these situations by means of a different choice of truth values. In the first situation, the resulting truth value must be \perp , while in the second it must be **false**. Technically, we realise this in such a way that each query to KAON2 results in two calls to the reasoner allowing to retrieve more detailed information. For the atom $dl_{subsumes}(C, D)$, the first query to the reasoner asks for $C \sqsubseteq D$. Given a positive answer, we know that this is demonstrable, thus the DL-atom is evaluated as **true**. When the answer is negative, there are, however, two cases possible: $C \sqsubseteq D$ might be satisfiable, but not formally implied by the knowledge base. In this case the DL-atom should have the value \perp i.e. unknown. On the other hand it is possible that the information in the knowledge-base makes $C \sqsubseteq D$ impossible. Then the DL-atom should be assigned **false**. This is done by the second query, which asks whether $KB \cup \{C \sqsubseteq D\}$ is satisfiable, where KB is the knowledge-base. We summarize the query in the following table.

| result of the query: $C \sqsubseteq D$ | result of the query: Is $KB \cup \{C \sqsubseteq D\}$ satisfiable? | value of $dl_{subsumes}(C, D)$ |
|---|---|-----------------------------------|
| yes | – | true |
| no | yes | \perp |
| no | no | false |

Note that the queries are executed consecutively, i.e. the second query is only performed if the first returned false.

A useful perspective on this is the following: $C \sqsubseteq D$ results in **true** if it holds in *all* models of the knowledge base. It results in **false** if it holds in *none* of the models of the knowledge base. And it results in \perp if it holds in *some*, but not all, models of the knowledge base.

The question of the satisfiability of a concept is reducible to subsumption: a concept C is satisfiable iff $C \sqsubseteq \perp$ does not hold, i.e. if there is *some* model in which the extension of C is non-empty. This situation is best understood by

considering *unsatisfiability* of a concept instead of satisfiability, as this allows us to use exactly the argumentation used above: A concept is unsatisfiable if it is extensionally empty in *all* models of the knowledge base. Similarly to the case of subsumption, we arrive at the execution detailed in the following table.

| result of the query: $C \sqsubseteq \perp$ | result of the query: Is $KB \cup \{C \sqsubseteq \perp\}$ satisfiable? | value of $dl_{unsatisfiable}(C)$ |
|---|---|-------------------------------------|
| yes | – | true |
| no | yes | \perp |
| no | no | false |

Querying for extensional disjointness of concepts is treated similarly, by reducing it to subsumption: two concepts C and D are disjoint iff $C \sqsubseteq \neg D$.

The query whether $C(a)$ holds can be resolved as in the following table. Note that $C(a)$ holds if it is true in *all* models.

| result of the query: $C(a)$ | result of the query: $\neg C(a)$ | value of $dl_{member}(C, a)$ |
|--------------------------------|-------------------------------------|---------------------------------|
| yes | – | true |
| no | yes | false |
| no | no | \perp |

The query $dl_{has_role}(I_1, R, I_2)$ provides information whether two individuals I_1 and I_2 are connected via a role R . When $\langle I_1, I_2 \rangle \in R$ then the DL-atom is **true**. To evaluate the other truth-values, we have to restrict ourselves to the known individuals, as it is not possible in OWL to ask for negated roles [15]. When querying whether two individuals are connected via a role, it is a sensible assumption that this might be possible, i.e. that $\langle I_1, X \rangle \in R$ or $\langle X, I_2 \rangle \in R$. So we assign the value **false** to queries when there exists either an $X \neq I_2$ with $\langle I_1, X \rangle \in R$ or an $Y \neq I_1$ with $\langle Y, I_2 \rangle \in R$ but $\langle I_1, I_2 \rangle \notin R$. All other pairs of individuals get the value \perp .

To integrate these DL-atoms flawlessly with the semantics of our logic programming environment which is based on fixed points, we need to guarantee that the values of the DL-atoms are monotone w.r.t. the knowledge order. For now, we assume that the knowledge-base is static, i.e. it cannot change during the program evaluation. Then the evaluation of the DL-atoms always yields the same result, and thus, trivially, is monotone.

The implemented system, called PrOWLog, is available for download from <http://logic.aifb.uni-karlsruhe.de/wiki/PrOWLog>.

5 An Example

We exemplify our approach by extending an example given in [1], formalising a judge's decision process, as given by the following rules.

$$\begin{aligned}
 is_suspect &\leftarrow has_motive \vee has_witness \\
 is_cleared &\leftarrow \neg contradict_alibi \wedge has_alibi \\
 charge &\leftarrow is_suspect \oplus \neg is_cleared
 \end{aligned}$$

The judge collects information suggesting that a person is suspect as well as information that indicates that the person is cleared. To support suspicion he collects information about the existence of a motive or a witness (first line). To enforce innocence the judge considers an alibi, but only if this is not contradicted by the defendant's testimony (second line). Finally he combines this information (third line). Assume now that the only information the judge has about some person is $has_witness \leftarrow \mathbf{false}$. Only relying on this, the suspect shouldn't be charged. Based on the closed-world assumption we get $has_motive = \mathbf{false}$ and thus $is_suspect = \mathbf{false}$. As $has_alibi = \mathbf{false}$, we obtain $is_cleared = \mathbf{false}$. So when evaluating $charge$ the information is contradictory and $charge$ gets the value \top .

Using the open-world assumption, giving all atoms the default value \perp , we get $is_suspect = \perp$ because $has_motive = \perp$ and $\mathbf{false} \vee \perp = \perp$. Since we know nothing about has_alibi and $contradict_alibi$, the default assumption is used again and we get $is_cleared = \perp$ and finally $charge = \perp$. So neither of the two established assumptions work in a satisfactory way. Consider now the mixed hypothesis H_m defined as follows: $H_m(has_witness) = \mathbf{false}$, $H_m(has_motive) = \mathbf{false}$, $H_m(has_alibi) = \perp$, $H_m(contradict_alibi) = \perp$. Then, like under the closed-world assumption, $is_suspect$ is false. The contradiction we encountered, however, does not exist any more as $is_cleared = \perp$ which reflects that the information is not sufficient to make a decision. Consequently $charge = \mathbf{false}$. This illustrates that the first line of the program is devised according to the closed-world assumption. The second line however is based on a different idea: For $is_cleared$ to become \mathbf{false} , $has_alibi = \mathbf{false}$ is already sufficient. So the meaning of $has_alibi = \mathbf{false}$ is that it has been proven that nobody can provide an alibi for the defendant. Then we need also the possibility to model the fact, that just no alibi is known, which corresponds to $has_alibi = \perp$, and which should be the default case. So the second line is conceived with an open-world setting in mind. H -founded models enable the use of such programs despite the different approaches involved.

To complete the example, it could be assumed that the judge draws his knowledge from an OWL DL knowledge base, by means of the following rules which query a knowledge base about a person Ted who is under investigation.

$$\begin{aligned} has_motive &\leftarrow dl_member(dl_has_motive, Ted) \\ has_witness &\leftarrow dl_member(dl_has_witness, Ted) \\ has_alibi &\leftarrow dl_member(dl_has_alibi, Ted) \\ contradict_alibi &\leftarrow dl_member(dl_contradict_alibi, Ted) \end{aligned}$$

By means of our hybrid semantics, the system will respond with the desired answer.

6 Related Work

Our approach using DL-atoms to link rule-based and ontology-based reasoning is inspired by the approach of Eiter et. al. presented in [16], where *extended logic*

programs and the *answer set semantics* [17] are modified to incorporate DL-atoms to query external reasoners. This approach permits the flow of information in both directions from the DL-enhanced program to the reasoner and vice versa. Extended logic programs make use of two negation operators, distinguishing explicitly negation as failure and classical negation. The any-world-semantics permits to manage this naturally, giving the negation operator of the lattice different meanings respective to the default assumption of the negated expression. In [18] Eiter et. al. generalize their approach to so called HEX-Programs, where the DL-atoms are replaced by atoms permitting to access a variety of different external sources, not only DL-reasoners. To accomplish this, the rule syntax and the answer set semantics are extended. The evaluation of these programs is made possible by a splitting algorithm based on the dependency structure in the program. Also based on DL-atoms, our approach was developed from another perspective. Given the elegant yet expressive any-world semantics and the ease of use of the hypotheses in KKS, we provide a logic programming environment with access to description logics, while remaining close to Prolog programming. We emphasize the use of the particular kind of information that can be drawn from DL-reasoners as an open-world based system with an intuitive semantics.

Motik and Rosati present in [19] an approach for a system combining rules and DL into one formalism. Based on MKNF [20] they join a decidable FOL fragment with logic programming rules. The modality operators in their so called *hybrid MKNF knowledge bases* allow to formulate rules to enforce closed world reasoning while maintaining the open world assumption for the DL-part. Their system also subsumes Rosati's approach in [21]. There and more detailed in [22] he discusses the relation of open and closed semantics in these hybrid systems.

7 Conclusions and Further Work

We have presented the hybrid reasoning system PrOWL_{og}, which allows to combine OWL DL with Prolog in such a way that the open-world semantics of OWL DL can be captured within the Prolog system. To the best of our knowledge, this is the first work which integrates a logic programming language and OWL in such a way.

We perceive basically two lines of further research to follow up on our results. On the one hand, studies remain to be done which show that the approach is useful in practice. We believe that in particular an integration with F-Logic reasoners is worth investigating, as F-Logic and OWL are two complementary ontology paradigms, which are both used in practice. On the other hand, it remains to be investigated whether the integration of Prolog and OWL can be strengthened by weakening the layering, i.e. by allowing some flow of information back to the OWL knowledge base, perhaps in a way similar to [18].

Acknowledgements. We gratefully acknowledge support by the German Ministry for Education and Research under the SmartWeb project grant 01 IMD01 B, by the European Commission under the NeOn project IST-2006-027595, and by

the Deutsche Forschungsgemeinschaft under the ReaSem project. We would also like to thank the members of the OntoLoRe group at AIFB Karlsruhe, and in particular Markus Krötzsch, for helpful discussions.

References

1. Loyer, Y., Straccia, U.: Any-world assumptions in logic programming. *Theoretical Computer Science* 342, 351–381 (2005)
2. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* 42, 741–843 (1995)
3. Angele, J., Lausen, G.: Ontologies in F-logic. In: Staab, S., Studer, R. (eds.) *Handbook on Ontologies*, pp. 29–50. Springer, Heidelberg (2004)
4. Ginsberg, M.L.: Multivalued logics: A uniform approach to inference in artificial intelligence. *Computational Intelligence* 4, 265–316 (1988)
5. Belnap, N.D.: A useful four-valued logic. In: Epstein, G., Dunn, J.M. (eds.) *Modern Uses of Multiple-Valued Logic*, pp. 5–37. Reidel, Dordrecht, Netherlands (1977)
6. Fitting, M.: The family of stable models. *Journal of Logic Programming* 17, 197–225 (1993)
7. Fitting, M.: A kripke-keene semantics for logic programs. *Journal of Logic Programming* 2, 295–312 (1985)
8. Fitting, M.C.: Bilattices in logic programming. In: 20th International Symposium on Multiple-Valued Logic, Charlotte, pp. 238–247. IEEE CS Press, Los Alamitos (1990)
9. van Gelder, A., Ross, K., Schlipf, J.S.: The well-founded semantics for general logic programs. *Journal of the ACM* 38, 620–650 (1991)
10. Tarski, A.: A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics* 5, 285–309 (1955)
11. Motik, B.: Reasoning in description logics using resolution and deductive databases. PhD thesis, Universität Karlsruhe (2006)
12. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Ganzinger, H., McAllester, D., Voronkov, A. (eds.) *LPAR 1999. LNCS*, vol. 1705, pp. 161–180. Springer, Heidelberg (1999)
13. Fitting, M.: Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science* 278, 25–51 (2002)
14. Kunen, K.: Negation in logic programming. *Journal of Logic Programming* 4, 289–308 (1987)
15. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Well-founded semantics for description logic programs in the semantic web. In: Antoniou, G., Boley, H. (eds.) *RuleML 2004. LNCS*, vol. 3323, pp. 81–97. Springer, Heidelberg (2004)
16. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. In: Dubois, D., Welty, C.A., Williams, M.A. (eds.) *KR2004: Principles of Knowledge Representation and Reasoning*, pp. 141–151. AAAI Press, Menlo Park, California (2004)
17. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9, 365–386 (1991)
18. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Effective integration of declarative rules with external evaluations for semantic-web reasoning. In: Sure, Y., Domingue, J. (eds.) *ESWC 2006. LNCS*, vol. 4011, pp. 273–287. Springer, Heidelberg (2006)

19. Motik, B., Rosati, R.: Closing semantic web ontologies. Technical report, University of Manchester, UK (2006)
20. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: Proceedings of IJCAI-91, San Mateo, CA., pp. 381–386. Morgan Kaufmann, San Francisco (1991)
21. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006, pp. 68–78. AAAI Press (2006)
22. Rosati, R.: Semantic and computational advantages of the safe integration of ontologies and rules. In: Fages, F., Soliman, S. (eds.) PPSWR 2005. LNCS, vol. 3703, pp. 50–64. Springer, Heidelberg (2005)

Applying Logical Constraints to Ontology Matching

Christian Meilicke and Heiner Stuckenschmidt

Computer Science Institute
University of Mannheim
A5, 6 68159 Mannheim, Germany
{christian, heiner}@informatik.uni-mannheim.de

Abstract. Automatically discovering semantic relations between ontologies is an important task with respect to overcoming semantic heterogeneity on the semantic web. Ontology matching systems, however, often produce erroneous mappings. In this paper we propose a method for optimizing precision and recall of existing matching systems. The principle of this method is based on the idea that it is possible to infer logical constraints by comparing subsumption relations between concepts of the ontologies to be matched. In order to verify this principle we implemented a system that uses our method as basis for optimizing mappings. We generated a set of synthetic ontologies and corresponding defective mappings and studied the behavior of our method with respect to the properties of the matching problem. The results show that our strategy actually improves the quality of the generated mappings.

1 Motivation

Recently, a number of heuristic methods for matching concepts from different ontologies have been proposed. These methods rely mostly on computing similarities based on linguistic and structural criteria. Evaluation studies have shown that existing methods often trade off precision and recall. The resulting mapping either contains a fair amount of errors or only covers a small part of the ontologies involved [2,4]. Our goal is to provide a component for matching systems that optimizes the results with respect to both recall and precision of the generated mapping. The method that we suggest is based on a reasoning approach that goes beyond existing structural methods and can be classified as semantic-based technique due to Euzenat [3].

1.1 Problem Statement

In accordance to Euzenat [3] the problem of ontology matching can be defined in the following way. For each ontology \mathcal{T} there is a function $Q(\mathcal{T})$ that defines matchable elements of \mathcal{T} . Given two ontologies \mathcal{T} and \mathcal{T}' the task of matching is to determine correspondences between the matchable elements in the two ontologies. Correspondences can be defined as 4-tuples $\langle e, e', r, c \rangle$ where $e \in Q(\mathcal{T})$, $e' \in Q(\mathcal{T}')$, r is a semantic relation and c is a confidence value from a suitable structure $\langle D, \leq \rangle$. In this work, we only consider the simple case where e and e' are concepts and $r = \equiv$ but there is some evidence that our approach can also be extended to $r \in \{\sqsubseteq, \sqsupseteq, \equiv\}$.

The problem we address in this paper is the following. Given a set of correspondences M' between two ontologies \mathcal{T} and \mathcal{T}' generated as intermediary or final result of a matching system and a set of correspondences M that contains all correct semantic relations between elements from the two ontologies, to determine $M' \cap M$. In the following we refer to a set of correspondences between two ontologies as a mapping.

1.2 Approach and Contributions

The approach taken in this work is to interpret the problem defined above on the one hand as optimization problem with respect to the confidences of the correspondences in M' . On the other hand we assume that any acceptable solution to the problem has to fulfill additional logical constraints imposed by the logical theories encoded in the ontologies. In particular, semantic relations between ontologies should not cause any inconsistencies. Therefore, we will develop a method that enables us to find a subset of M' that is the optimal mapping among the consistent mappings in the powerset of M' . The concrete contributions of this work are:

- We propose a method for automatically optimizing automatically generated mappings based on a reasoning approach that is orthogonal to existing matching approaches.
- We give a detailed explanation of the developed algorithm and implemented it in a tool for mapping optimization.
- We applied this tool on several synthetic data sets, to identify success factors for optimizing ontology mappings in terms of properties of the mapping and the matched ontologies.

The paper is organized as follows. First, we discuss related work and explain why our approach goes beyond existing structural methods used in ontology matching. In section 2 we introduce the main principle of our approach and define the mapping property of consistency. In section 3 we describe the algorithms and components that our method is based on and show how these components can be integrated into a system for optimizing mappings. The experiments we conducted are described in section 4. In particular, we analyse the relation between certain properties of the matching problem and their influence on the results of the suggested method. We close with a general discussion of the approach and possible extensions in future work in section 5.

1.3 Related Work

A variety of methods for computing mappings have been proposed. A common feature of these methods is that they are based on an initial mapping created by matching labels that occur in the ontologies and successively improve this initial mapping by combining and updating mappings based on certain heuristics [3]. The work described in this paper is a new approach for combining and updating an initial mapping set.

So far two types of methods for improving an initial mapping have been proposed. The first kind of approaches are so-called structural techniques. They are based on the propagation of evidences for certain mapping hypotheses based on the structure of the

ontologies. The GLUE system [1] uses relaxation labeling to update the probability that a mapping is correct. The main idea is that the label (concept in the target ontology) of a node (concept in the source ontology) is influenced by the features of the nodes neighborhood in the subsumption graph. The OMEN tool [11] uses a Bayesian network, where a node stands for a correspondence between classes or properties of the ontologies and where an edge represents the influences between individual correspondences. In order to generate conditional probability tables for the given network a set of meta-rules is used.

The other kind of methods are so-called semantic methods that try to infer additional correspondences or try to eliminate incorrect correspondences using logical reasoning. The idea is to encode the semantics of concepts as well as the initial mapping in a logical theory. Additional correspondences are inferred by proving implications between formulas that represent concepts in the different ontologies. In [6] a semantic matching approach is proposed that uses propositional logic for encoding the semantics of concept labels and uses a SAT prover to derive mappings. In [8] the approach is extended to the task of matching structured representations by coding them into description logics and inferring subsumption relations across ontologies. A prequel of the approach described in this paper has been suggested in [9] and [10] where we use conflict sets and distributed reasoning to eliminate potentially incorrect correspondences. Notice that the strategy suggested in [10] has only been applied in the context of mapping repairing. The algorithm to solve this problem consists of a sequence of local decisions not taking into account the whole distribution of confidence values as well as the complex interdependences between inconsistencies.

Both of these approaches have shortcomings. While structural approaches that rely on numerical methods have problems in capturing hard semantic constraints, semantic approaches that solely rely on logical reasoning are often too strict to capture all valid mappings and suffer from problems in modelling the soft constraints implied by confidence values. In our work we combine numerical and logical methods thereby leveraging the problems of the individual approaches.

2 General Approach

In this section we revert to some examples based on the scenario that has been described by Quine as radical translation [14]. Radical translation is concerned with the problem of finding a correct translation manual for a fully unknown language L . Obviously, this problem is closely related to the problem of finding a correct mapping between ontologies. Therefore, it is useful to explain the intuition that forms the basis of our strategy first in the context of radical translation. Later in this section, we will shift to the problem of ontology matching and give a formal representation.

2.1 Translation

Suppose that a linguist wants to explore the unknown language L of some people that have not been in contact to human civilization yet. The native people accept the researcher and let him be part of their daily life. At the first stage of his project the linguist

simply observes the linguistic behavior of the natives and establishes some hypothesis about the meaning of the words that are uttered by the natives. The following could be a typical example for such a situation.

Example 1. The linguist and a native are standing in front of an oak tree. A rabbit is sitting close to the tree. The native points at the direction of the tree and utters the word "Gavagai!". The linguist considers two possible hypothesis about the meaning of the word. Gavagai could on the one hand refer to oak or could on the other hand refer to rabbit. He writes both hypothesis in his dictionary and marks them with a q as questionable.

As time goes by, the linguist is able to utter simple sentences in L . He also finds out which words and gestures mean approval and rejection. After a while he also manages to ask questions of the form "Are all x y ?" translated to L . This enables him to apply a more elaborative strategy.

Example 2. From time to time the linguist cleans up the entries in his dictionary. He finds, amongst others, the following three entries.

$$gavagai = rabbit_q \quad (1)$$

$$gavagai = oak_q \quad (2)$$

$$snok = tree \quad (3)$$

In order to find out if the first or the second entry has to be removed he asks the native the question "Are all gavagais snoks?". The native looks quite confused and denies the question. For that reason the linguist removes the second entry and keeps the first one.

The reasoning that is the base for the linguists decision follows this line. If *gavagai* means *oak* and *snok* means *tree* then everything that is a *gavagai* also has to be a *snok*, because the linguist knows that an oak is a special kind of a tree. He transfers this subsumption relation to the concepts *gavagai* and *snok*. By asking the question "Are all gavagais snoks?" the linguist checks if this entailment is accepted by the native. The native denies this question and therefore the linguist is justified in removing the second or the third entry. Since he has marked the second entry as questionable he decides to remove it instead of removing the third entry.

2.2 Formalization

The problem of radical translation is structurally closely related to the problem of automated generation of mappings between ontologies. An ontology can be understood as a formal representation of (parts of) a language that can be used to talk about certain parts of the world. Thus, every entry in the dictionary of the linguist can be interpreted as a correspondence in an ontology mapping.

How can the strategy of the linguist be formalized in order to apply it to the problem of matching ontologies? The linguist uses his dictionary to connect both views of the world. By doing this he derives knowledge about the subsumption relations of the natives concepts. The same can be done in the context of ontologies by defining the union of two ontologies connected by a mapping in the following straight forward way.

Definition 1 (Union of ontologies). Let \mathcal{T}_1 and \mathcal{T}_2 be ontologies (finite sets of axioms). The union $\mathcal{T}_1 \cup_{\mathcal{M}} \mathcal{T}_2$ of \mathcal{T}_1 and \mathcal{T}_2 connected by \mathcal{M} is defined as $\mathcal{T}_1 \cup_{\mathcal{M}} \mathcal{T}_2 = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{t(x) \mid x \in \mathcal{M}\}$ with t being a translation function that converts correspondences into axioms in the following way:

$$t(\langle 1: C, 2: D, \equiv, c \rangle) = 1: C \equiv 2: D$$

Such a union ontology defines a taxonomy determined by the axioms of both ontologies and the additional correspondence axioms. Consider again example 2. The linguist first had the union ontology in mind that results from the mapping consisting of dictionary entries 2 and 3. Therefore, the linguist inferred the axiom $L : gava.gai \sqsubseteq L : snok$. If the native would accept this axiom his ontology would become inconsistent. Thus, the native denies the question of the linguist. We call the corresponding notion mapping consistency and introduce it formally as follows.

Definition 2 (Consistency of a Mapping). Given ontologies \mathcal{T}_1 and \mathcal{T}_2 and a mapping \mathcal{M} between \mathcal{T}_1 and \mathcal{T}_2 . \mathcal{M} is consistent iff there exists no concept $i: C$ with $i \in \{1, 2\}$ such that $\mathcal{T}_i \not\models i: C \sqsubseteq \perp$ and $\mathcal{T}_1 \cup_{\mathcal{M}} \mathcal{T}_2 \models i: C \sqsubseteq \perp$. Otherwise \mathcal{M} is inconsistent.

As we have argued, an inconsistent mapping is a mapping that contains erroneous correspondences. In the next section we will define the closely related notion of pairwise mapping consistency and we will see how to apply it to the problem of mapping optimization.

3 Algorithms

The problem stated in section 1.1 can now be addressed in the following way. On the one hand we have to find a subset M^* of M' that is an approximation of the correct mapping M taking into account the confidence values of the correspondences in M' . Since we are only focussed on equivalence correspondences, we can restrict M^* to be a one-to-one mapping. A one-to-one mapping is a mapping that contains no pairs of correspondences $\langle c_1, c_2 \rangle$ with $c_1 \neq c_2$ such that the source (target) concept of c_1 is also the source (target) concept of c_2 . On the other hand M^* has to be a consistent mapping. Thus, we need an efficient algorithm that finds a consistent mapping M^* that is also optimal with respect to the confidence values of M' . We will now introduce such an algorithm that is based on three components. The first two components provide methods for optimization respectively checking consistency, while the third component combines these methods to solve the problem.

3.1 Optimization

Finding an optimal solution to the one-to-one matching problem based on the confidence values of the elements of M' depends on choosing an appropriate aggregation function. We decided to maximize the sum of all confidence values. The optimization problem can thus be stated in the following way:

Definition 3 (Optimal solution). Given a set of correspondences M' , an optimal one-to-one mapping $M_{opt} \subseteq M'$ is a one-to-one mapping such that for every other one-to-one mapping $M'' \subseteq M'$ we have $\sum_{c \in M_{opt}} confidence(c) \geq \sum_{c \in M''} confidence(c)$.

A standard algorithm to solve this problem is known as the hungarian method [7]. In order to show how this method can be applied to our problem a few explanations have to be given. The hungarian method expects a real-valued matrix as input and creates a one-to-one assignment, such that the sum of the chosen entries is minimal. To use the hungarian method the input mapping M' has to be transformed into a corresponding matrix H . Each concept of the source ontology corresponds to a row and each target concept corresponds to a column. Since the hungarian method finds a minimal assignment an entry in the matrix has to be interpreted as distance between two concepts, where the distance between $1: C$ and $2: D$ is defined as $1 - confidence(\langle 1: C, 2: D, \equiv, c \rangle)$. Without loss of generality we assume that the input confidence values are in the interval $[0, 1]$. If there exists no such correspondence in M' the distance is set to ∞ .

In most matching situations it will not be possible to match all or even the majority of concepts. Matching candidates will thus not be available. Therefore, the input matrix has to be extended by additional concepts that play the role of alternative matching candidates. We call these concepts phantom concepts. Thus, if n is the number of concepts in \mathcal{T}_1 and m is the number of concepts in \mathcal{T}_2 , we add m rows to the input matrix corresponding to m phantom concepts $1: P_1, \dots, 1: P_m$ as well as n columns corresponding to n phantom concepts $2: P_1, \dots, 2: P_n$. The value of the entries in these rows respectively columns is set to $1 + \epsilon$ with $\epsilon > 0$. Thus, for a given concept $1: C$ the algorithm will first try to find a corresponding concept $2: D$. If this is not possible within the context of global minimization one of the phantom concepts $2: P_1, \dots, 2: P_n$ is chosen. In such a case $1: C$ is interpreted as unmatchable.

Example 3. Assume mapping M consists of the following correspondences between \mathcal{T}_1 and \mathcal{T}_2 .

$$\langle 1: C, 2: X, =, 0.94 \rangle \tag{4}$$

$$\langle 1: C, 2: Y, =, 0.29 \rangle \tag{5}$$

$$\langle 1: D, 2: X, =, 0.12 \rangle \tag{6}$$

$$\langle 1: E, 2: X, =, 0.31 \rangle \tag{7}$$

The corresponding input matrix to the hungarian method looks like this:

| | $j: X$ | $j: Y$ | $2: P_1$ | $2: P_2$ | $2: P_3$ |
|----------|----------------|----------------|----------------|----------------|----------------|
| $1: C$ | 0.06 | 0.71 | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ |
| $1: D$ | 0.88 | ∞ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ |
| $1: E$ | 0.69 | ∞ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ |
| $1: P_1$ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ |
| $1: P_2$ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ | $1 + \epsilon$ |

Suppose we set $\epsilon = 9$. Applying the hungarian method will result in the one-to-one mapping consisting of correspondences 5 and 7. Concept $1: D$ is mapped on a phantom concept. The aggregated distance value for the chosen correspondences is 31.4. ϵ is

an important parameter that affects the behaviour of the mapping extraction. Consider again example 3 with ϵ set to 0.001. The hungarian method will now find another optimal extraction consisting of only one correspondence, namely correspondence 4. This mapping extraction has an aggregated cost of 4.1. The algorithm chooses this mapping, since the cost of assigning an additional concept to a phantom concept is less than the relatively high cost of choosing two correspondences with a high distance. We will later see how to set the value of ϵ according to the properties of the matching problem to be solved.

3.2 Reasoning

The second component of our system consists of an incomplete but efficient reasoning method. This method can be used to detect mapping inconsistencies and subsets of the mapping that cause these inconsistencies. As our approach extensively relies on checking consistency in the mapped ontologies, using existing methods for reasoning about ontologies is infeasible.

Therefore, we decided to use the following method that allows us to check mapping consistency for each mapping of cardinality two. First, we use a reasoner to compute the whole concept hierarchy of both \mathcal{T}_1 and \mathcal{T}_2 . For each ontology \mathcal{T}_i with $i \in \{1, 2\}$ we save the results in a subsumption matrix that contains the information if $i : C$ is a subclass of $i : D$ for each pair of concepts $\langle i : C, i : D \rangle$. In the same way we prepare a disjointness matrix that contains the information if two concepts are disjoint. Having once computed these four matrices the reasoning method can be implemented by comparing entries in matrices. Given a set of correspondences $M = \{1 : C, 2 : D, =, v_1\}, \langle 1 : E, 2 : F, =, v_2 \rangle\}$ we first check if $1 : C$ is a subclass or superclass of $1 : E$. If this is the case $2 : D$ and $2 : F$ cannot be disjoint because this would result in an inconsistency in the union ontology $\mathcal{T}_1 \cup_M \mathcal{T}_2$. We also apply the same procedure in the other direction by entailing subsumption relations from \mathcal{T}_2 to \mathcal{T}_1 accompanied by checking disjointness in \mathcal{T}_1 . Notice that inconsistencies can also emerge, if a concept $j : D$ becomes a subclass of $j : C$ even if $j : C$ and $j : D$ are not defined as disjoint in \mathcal{T}_j . This may happen if a subclass of $j : D$ is defined as disjoint to $j : C$. Therefore, all subclasses of a class that becomes subsumed have to be checked for disjointness, too.

In order to apply this strategy to a mapping M that consists of more than two correspondences we check consistency of each dual-element subset of M . This results in a correct but incomplete reasoning method for checking consistency of the whole mapping M . We call a mapping M that is consistent for all pairs of correspondences pairwise-consistent. The according property can be defined as follows.

Definition 4 (Pairwise-Consistency of a Mapping). *Given ontologies \mathcal{T}_1 and \mathcal{T}_2 and a mapping \mathcal{M} between \mathcal{T}_1 and \mathcal{T}_2 . \mathcal{M} is pairwise-consistent iff there exists no $\mathcal{M}' \subseteq \mathcal{M}$ with $|\mathcal{M}'| = 2$ such that \mathcal{M}' is inconsistent. Otherwise \mathcal{M} is pairwise-inconsistent.*

An advantage of this reasoning approach, in addition to its efficiency, is the fact that we can restrict the reason for inconsistency to a pair of correspondences. Due to our considerations in the previous sections, we thus know that one of the elements in the pair cannot be accepted. Note that exactly the same kind of reasoning has been used by the linguist in example 2.

Even though most of all inconsistencies can be detected by this strategy, there are inconsistencies consisting of more than two correspondences. Consider the following example.

Example 4. Let \mathcal{T}_1 and \mathcal{T}_2 be ontologies describing the domain of conferences in a slightly different way. Suppose that both ontologies contain the concepts *Paper* and *Poster*. In \mathcal{T}_1 there is a super concept *Submission* that is defined as the union of both concepts, while in \mathcal{T}_2 the equivalent concept is called *Document*. \mathcal{T}_1 contains the additional concept *ReviewedSubmission* which is defined to be a sub concept of *Submission*. In \mathcal{T}_2 the concept *SubmissionDeadline* is defined to be disjoint with *Document*. Now let \mathcal{M} be a mapping between \mathcal{T}_1 and \mathcal{T}_2 that consists of the following three correspondences.

$$\langle 1: Paper, 2: Paper, =, 1.0 \rangle \quad (8)$$

$$\langle 1: Poster, 2: Poster, =, 1.0 \rangle \quad (9)$$

$$\langle 1: ReviewedSubmission, 2: SubmissionDeadline, =, 0.52 \rangle \quad (10)$$

By using [8](#) and [9](#) it can be inferred that 1: *Submission* and 2: *Document* are equivalent concepts. Now we have the situation that in \mathcal{T}_1 *ReviewedSubmission* is a sub-concept of *Submission* while in \mathcal{T}_2 *SubmissionDeadline* is disjoint with *Document* which results in a conflict with correspondence [10](#). Thus, we have an example for a mapping that is obviously inconsistent due to definition [2](#) but contains no inconsistent pair of correspondences due to definition [4](#).

Notice that example [4](#) is quite complex. Combinations of correspondences that follow a similar pattern will occur relatively infrequently in automatically generated mappings. In experiments on real-world ontologies this consideration could be verified.

3.3 Search

The two components described in the sections above can be combined in a uniform cost search. Thus, it is possible to find the best solution to the matching problem among the set of all pairwise-consistent solutions. The algorithm starts with the optimal solution that results from applying the hungarian method to the original matching problem as root node. Therefore, we convert the input mapping M to a distance matrix H as described in section [3.1](#) and compute the optimal one-to-one mapping M' . If M' is a consistent mapping, we have found a solution to the problem in the first step. If M' is not pairwise-consistent, there exists at least one conflicting pair of correspondences $\langle c_1, c_2 \rangle$ in M' . We now know that a consistent solution must not contain both c_1 and c_2 , while it is possible that one of these correspondences is contained in the final solution that we are searching for. Therefore, we have to consider two branches in our search tree. In the first branch we have to search for a solution that must not contain c_1 , while in the second branch we have to search for a solution that must not contain c_2 .

Now remember that the hungarian method will never chose a mapping that consists of a correspondence with a distance value of ∞ . We can make use of this property to

create the branches in the search tree. Therefore, we have to set the cell corresponding to c_1 to ∞ in the first branch while setting the cell corresponding to c_2 to ∞ in the second branch. In the following we describe this procedure as locking a cell in H , which corresponds to making the associated correspondence unavailable in this branch. In this way we create new search states based on the results of the consistency checks applied to every state that gets expanded. For each new search state we compute its aggregated minimum value by applying the hungarian method and save the state and its associated minimum in minimum-priority queue. Notice that a search state is fully described by a set of locked cells and the associated minimum with respect to a particular input matrix H . In each step of the search we expand the state with lowest minimum. Thus, our algorithm finally results in a uniform cost search that uses the techniques described in the sections above to create new search states and to decide which states to expand first.

The algorithm terminates if it expands a state that corresponds to a pairwise consistent mapping. Due to the property of the uniform cost search, to first expand a set of locks with lowest aggregated distance, any other set of locks with a lower aggregated distance must have been expanded in an earlier step of the search, and thus has to be a pairwise-inconsistent mapping. Therefore, the algorithm will always find an optimal pairwise-consistent solution to the matching problem.

The runtime of the algorithm is exponential in worst case. The actual runtime depends on the structure of the input matrix. The most influential parameters are the size of the input mapping M , the size of the ontologies, the number of pairwise-inconsistencies caused by M , the quality of the confidence ordering in M with respect to correct and incorrect correspondences, and the number of matching alternatives in M .

4 Experiments

In the following we describe some of the experiments we conducted on synthetic data sets. In these experiments we examine the relation between certain parameters of the matching problem on the one hand and precision and recall of our algorithm on the other hand. In particular, we will address the following questions:

- How does the **fraction of correct correspondences** in the mapping affect the results of our approach?
- In how far does the **fraction of concepts to be covered via correspondences** influence the results?
- How strong and under which circumstances does the **quality of the confidences** affect the optimization process?
- Does the **structure of the ontologies** influence the results of our approach?

By using synthetic datasets we can vary these parameters with respect to the question under discussion. Besides the properties of the matching problems we will also vary ϵ and study interrelations between different ϵ -values and the characteristics of the matching problem.

4.1 Experimental Settings

Data sets. We construct some synthetic ontologies $\mathcal{T}_{b,d}$ where b denotes the branching factor and d the depths of the subsumption hierarchy. Further, we define sibling

concepts to be disjoint. We consider one-to-one mappings $M_{b,d}$ to the same ontology, matching each concept to itself. In the experiments we randomly choose subsets of $M_{b,d}$ of varying size to represent correct correspondences between concepts. We refer to the size of this subset as coverage in the following. In addition, we add a certain amount of incorrect correspondences by linking randomly chosen concepts $C \neq D$ from $\mathcal{T}_{b,d}$. To simulate matching systems that differ in the quality of confidence estimations we assign confidence values to correct and incorrect correspondences according to certain patterns that will be explained below. Notice that a synthetic mapping M' constructed in this way has to be understood as an intermediate result of a matching system. The final result has to be extracted from M' . This final step is often done using the hungarian method [7] which produces an global optimum (see [3]). Therefore, we will always compare the synthetic mapping after applying the hungarian method, to the mapping after applying our algorithm.

Experiments. The first experiment is based on the synthetic ontology $\mathcal{T}_{3,3}$ and the associated mapping $M_{3,3}$. We generated $M_{3,3}$ and decided to analyze subsets with a coverage from 0.1 to 1.0 increasing the coverage stepwise by 0.05. For each of these mappings we added n incorrect correspondences respectively $n/3$ incorrect correspondences where n is the number of concepts in the particular mapping. Thus, we created mappings with a precision of 0.5 respectively 0.75. For each mapping we distributed confidence values randomly not distinguishing between correct and incorrect correspondences. Notice that such a distribution is a challenge for every approach that tries to improve the quality of an input mapping. It can be adopted that matching systems will generate more reliable confidence estimations in most cases. Normally, the recall of a mapping M' is defined as $|M' \cap M|/|M|$, where M is the reference mapping that contains all correct correspondences between the ontologies to be matched. Since M is not known to us we compare the number of correct correspondences in the synthetic mapping to the number of correct correspondences left after applying the standard extraction respectively our algorithm and interpret the fraction as recall.

In the second experiment we are concerned with the probability of a correct correspondence having a greater confidence value than an incorrect correspondence. We refer to this probability as the perfection of a mapping. Notice that this parameter is different from the precision of a mapping. More precisely, the perfection of a mapping is the probability that a randomly chosen correct correspondence has a higher confidence than a randomly chosen incorrect correspondence. Again, we created $M_{3,3}$ and randomly chose subsets with a coverage of 0.2. In contrast to the first experiment we added more incorrect correspondences resulting in a relatively low precision of 0.3. To study different distributions of confidence values we increased perfection from 0.5 to 0.9 stepwise by 0.1. A mapping with low precision but relatively high perfection can be regarded as the intermediary result of a matching system which is optimized for recall. In order to compare the effects of different ϵ -values we applied the algorithm with $\epsilon = 0.001$ and $\epsilon = 100$. For each configuration we repeated the experiment 100 times focussing on the resulting mean values.

The third experiment deals with the issue of different subsumption hierarchies. Therefore, we decided to apply our algorithm on ontologies that differ in depth and branching factor to obtain information about the influence on the optimization process.

4.2 Experimental Results

The results of the first experiment are presented in figure 1. The left side shows the results with respect to precision, the right side refers to recall. There are four curves plotted in both parts of the figure. The data series resulting from an input mapping with a precision of 0.75 are marked with a rectangle. The data series resulting from an input mapping with a precision of 0.5 are marked with a circle. For both experiments we compared the mapping after applying the hungarian method (dashed grey line) to the mapping after applying our algorithm (black solid line). The results are based on randomly creating 500 mappings for each setting computing the mean of precision and recall with ϵ set to a high value, that forces our algorithm to find a consistent mapping that has a maximum number of elements.

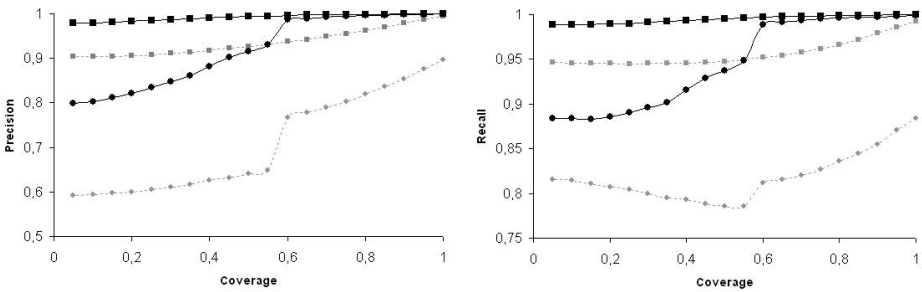


Fig. 1. Precision and recall with respect to coverage and precision of the input mapping

First, let us consider the results with respect to precision. For an input precision of 0.5 and a coverage from 0.05 to 0.6 we observe a difference of approximately 20% comparing the straight forward one-to-one mapping extraction to the results based on applying our algorithm. The differences become smaller with increasing coverage. Notice that in most real world matching problems there will only be a relatively low coverage. Even if two ontologies describe largely overlapping domains their concepts can be overlapping without being equivalent. Equivalence correspondences will thus only cover a small number of concepts. Therefore, the results for a high coverage are of minor interest. For an input precision of 0.75 we can observe a similar pattern. The precision of the one-to-one mapping extraction starts at 0.9 while applying logical constraints increases this value to 0.97 even for a coverage of 0.05. For a coverage above 0.35 the average precision of the optimized mapping is continuously greater than 0.99.

The results for recall are similar to the results for precision. As mentioned above we defined recall in our setting with respect to the correct correspondences of the input mapping. Thus, the first extraction as well as the final extraction decreases the actual recall of the input mapping that we interpreted as the initial or intermediary result of a matching system. But since any matching system that aims at generating a one-to-one mapping has to find an extraction from an intermediary result - which could e.g. be a whole similarity matrix - we are well-founded in comparing the extraction based on the hungarian method to the application of our algorithm. For settings with a low coverage

we can increase recall by approximately 5% for both mappings with a precision of 0.5 and 0.75. These results look unusual at first sight. One might have expected a trade-off between precision and recall. The reason for increasing both precision and recall is based on the fact that our method does not filter out particular correspondences as long as an alternative is available. More precisely, instead of selecting a subset of the first extraction, the algorithm forces rearrangements upon the assignments available. These rearrangements will with some probability result in choosing more correct correspondences compared to the first or some of the previous mapping extractions computed in the search procedure. This approach is opposed to the method suggested in [9] where inconsistencies are used to filter out correspondences by removing the correspondence with the lowest confidence in a conflict set.

We can therefore conclude that the proposed approach is capable of increasing precision as well as recall of an input mapping to a substantial extent for both low and high input precision. Actually, our algorithm has stronger effects on input mappings with a low precision. The explanation for this observation is simply based on the fact that with increasing number of incorrect correspondences the probability of conflicting pairs of correspondences increases, too. Nevertheless, this consideration can only be extended to a certain degree. For an input mapping M with extremely low precision there could also be a subset M^* of M consisting mostly of incorrect correspondences that are pairwise-consistent. If this subset is larger than the subset of correct correspondences M' (extended by some incorrect and not conflicting correspondences) the algorithm will choose M^* instead of M' . You should also notice that there is a substantial variance not depicted in the mean values of figure 1. By taking a look at the negative outliers that cause this variance, the pattern just mentioned can be detected. Since the probability of consistent sets of type M^* decreases with increasing size of M , the variance decreases also with increasing coverage.

Table 1. The influence of perfection on precision and recall for low and high ϵ

| Perfection | $\epsilon = 0.001$ | | | $\epsilon = 100$ | | | Δ f-value |
|------------|--------------------|--------|---------|------------------|--------|---------|------------------|
| | Precision | Recall | f-value | Precision | Recall | f-value | |
| 0.5 | 0.370 | 0.551 | 0.443 | 0.399 | 0.634 | 0.490 | -0.047 |
| 0.6 | 0.389 | 0.589 | 0.468 | 0.393 | 0.630 | 0.484 | -0.016 |
| 0.7 | 0.431 | 0.659 | 0.521 | 0.417 | 0.671 | 0.514 | 0.007 |
| 0.8 | 0.458 | 0.705 | 0.555 | 0.434 | 0.702 | 0.536 | 0.019 |
| 0.9 | 0.478 | 0.739 | 0.581 | 0.446 | 0.724 | 0.552 | 0.029 |

In the second experiment we focussed on the relation between the value of ϵ and the perfection of the input mapping. The results of this experiment are summarized in table 1. Each row in the table shows the results for a particular input perfection. Besides precision and recall we also added the f-value which is the weighted harmonic mean of both. This value provides us with an overall estimation of the performance of our

algorithm. In order to compare the results for $\epsilon = 0.001$ and $\epsilon = 100$ we listed the differences of both approaches with respect to the resulting f-values in the last column.

First, we see that there is a strong positive correlation between perfection of the input mapping and precision respectively recall of the outcome. This result is not surprising. Nevertheless, it shows that the algorithm makes use of the additional information encoded in the confidence values. It is more interesting that for $\epsilon = 0.001$ this information has much stronger positive effects. With respect to the f-value we gain an advancement of 0.138 for $\epsilon = 0.001$ if we compare both extremes in perfection, while the advancement for $\epsilon = 100$ is limited to 0.062. What is the reason for this difference? Assume there are (amongst others) two overlapping pairs of conflicting correspondences $\langle c_1, c_2 \rangle$ and $\langle c_1, c_3 \rangle$ in the input mapping. There are two possibilities to solve this problem: Discard c_1 or discard c_2 and c_3 . In the context of the algorithm we would put a lock on the associated cells in the matrix. If we now choose a high ϵ value the algorithm is forced to take the first option, not at all taking into account the confidence values involved. The situation changes if we set $\epsilon = 0.001$. Now the algorithm will make its decision based on comparing $confidence(c_1) + 1.01$ to $confidence(c_2) + confidence(c_3)$. A decision that is based on this comparison, obviously, makes sense only if the confidence values under consideration are with some probability correct estimations. By definition the perfection of a mapping is a parameter that is characteristic for this probability. For a high perfection decisions based on the consideration explained above will expand the search tree in the correct direction with a high probability, while for a low precision these considerations will be misleading. This explains the values presented in the last column. For a low precision the conservative strategy - using a high ϵ value and therefore keeping as much correspondences as possible - works better, while for a high perfection the conservative strategy cannot exploit the additional information encoded in the confidence values to its full extent.

In the third experiment we studied the behavior of the algorithm working with ontologies that differ in their hierarchical structure. Obviously, the applicability of our algorithm relies on the existence of pairwise inconsistencies. With respect to the synthetic ontologies we used in our experiments the number of subsumption statements and disjointness statements is determined by the branching factor and the depth. Thus, we varied these parameters to understand their impact on the results. In one of our test-cases we compared mappings for $\mathcal{T}_{7,2}$ (relatively flat subsumption hierarchy with 56 concepts) and $\mathcal{T}_{2,5}$ (deep subsumption hierarchy with 62 concepts). We could achieve stronger effects in optimizing mappings linking concepts of $\mathcal{T}_{2,5}$. Compared to $\mathcal{T}_{7,2}$ we measured 8% more precision and 3% more recall. There are several reasons for this difference. With respect to $\mathcal{T}_{7,2}$ pairwise inconsistencies will on the one hand only occur, if there are some correspondences in the input mapping that link concepts of the first level. On the other hand incorrect correspondences linking sibling concepts that are leaves will never cause pairwise-inconsistencies in a one-to-one mapping. If we compare results for e.g. $\mathcal{T}_{4,3}$ to $\mathcal{T}_{2,5}$ the differences become noticeable smaller. We can conclude that our approach works slightly better on matching ontologies with a deep subsumption hierarchy.

5 Discussion and Conclusions

We presented an approach to optimize matching systems based on a combination of numerical optimization and logical reasoning, thereby leveraging the problems of existing approaches that are solely based on optimization or logical reasoning, respectively.

We introduced the basic principle of the approach based on the idea of radical translation and transferred it to the problem of matching ontologies. We defined the properties of mapping consistency and pairwise mapping consistency as a basis for automating this principle. We presented an algorithm for computing an optimal and pairwise-consistent mapping that combines the hungarian method with a uniform cost search over the space of pairwise consistent mappings. We ran several experiments on synthetic data sets and showed that our method increases precision and recall of a mapping in comparison to the result of applying standard optimization techniques without considering mapping consistency. In particular, we showed that even for mappings with poor confidence estimations our approach works quite well.

There are two main lines of future work. The first is concerned with improving the efficiency of our method to scale up to large real world ontologies. While the incomplete reasoning method we use is rather efficient due to the pre-compilation of the subsumption and the disjointness matrix, the complexity of the current approach is mainly influenced by the optimization and the search procedure. The hungarian algorithm currently used for the optimization has a complexity of $O(n^3)$. Sacrificing global optimality we can improve this by replacing the hungarian method with the Gale-Shapley algorithm [5] which runs in $O(n^2)$. We expect major improvements from using more sophisticated search procedures instead of uniform cost search to deal with the combinatorial explosion of space of consistent mappings. We will also investigate the use of efficient but incomplete search methods such as greedy or stochastic local search.

The second major point for future work is the systematic application of the method to real data sets. In this paper, our aim was to better understand the behavior of our method in terms of the influence of different problem characteristics. For this purpose, artificial data sets are better suited than real ones. Now that we gained an understanding of the success factors, we are ready to tackle real matching problems. First experiments we carried out on data sets from the ontology alignment evaluation initiative have shown that the critical point here is the assumption we make about the presence of disjointness statements in the ontologies. It turned out that ontologies often do not contain such statements even though the concepts are clearly disjoint. A possible solution is to simply add disjoint statements for all sibling concepts. In [12] it has been shown that this radical approach works well in many cases. A more sophisticated approach is to try to learn disjointness statements for underspecified ontologies based on suitable text corpora and background knowledge. In [13] first results for this approach are reported that suggest that learning is indeed feasible. We will investigate and contrast these different approaches on real data in future work.

Acknowledgement

The work has been partially supported by the German Science Foundation (DFG) in the Emmy Noether Programme under contract STU 266/3-1.

References

1. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.: Learning to match ontologies on the semantic web. *The International Journal on Very Large Data Bases* 12, 303–319 (2003)
2. Euzenat, J., Mochol, M., Shvaiko, P., Stuckenschmidt, H., Svab, O., Svatek, V., van Hage, W.R., Yatskevich, M.: First results of the ontology alignment evaluation initiative 2006. In: Benjamins, R., Euzenat, J., Noy, N., Shvaiko, P., Stuckenschmidt, H., Uschold, M. (eds.) *Proceedings of the ISWC 2006 Workshop on Ontology Matching*, Springer, Heidelberg (2006)
3. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (to appear, 2007)
4. Euzenat, J., Stuckenschmidt, H., Yatskevich, M.: Introduction to the ontology alignment evaluation 2005. In: *Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies*, Banff, Canada (2005)
5. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. *American Mathematical Monthly*, 9–14 (1962)
6. Giunchiglia, F., Yatskevich, M., Shvaiko, P.: Semantic matching: Algorithms and implementation. *Journal on Data Semantics* (to appear, 2007)
7. Kuhn, H.W.: The hungarian method for the assignment problem. *Naval Research Logistics*, 83–97 (1955)
8. Sceffer, S., Bouquet, P., Serafini, L., Zanobini, S.: Matching hierarchical classifications with attributes. In: Sure, Y., Domingue, J. (eds.) *ESWC 2006*. LNCS, vol. 4011, pp. 4–18. Springer, Heidelberg (2006)
9. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Improving automatically created mappings using logical reasoning. In: *ISWC-06 Workshop on Ontology Matching*, Athens, GA, USA (2006)
10. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Repairing ontology mappings. In: *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, Vancouver, Canada (2007)
11. Mitra, P., Noy, N.F., Jaiswal, A.R.: Omen: A probabilistic ontology mapping tool. In: *ISWC-04 Workshop on Meaning Coordination and Negotiation*, Hiroshima, Japan (2004)
12. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: Gómez-Pérez, A., Euzenat, J. (eds.) *ESWC 2005*. LNCS, vol. 3532, Springer, Heidelberg (2005)
13. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: Learning disjointness. In: *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, Springer, Heidelberg (2007)
14. Quine, W.V.: *Word and Object*. MIT Press, Cambridge (1960)

Resolving Inconsistencies in Probabilistic Knowledge Bases

Marc Finthammer, Gabriele Kern-Isberner, and Manuela Ritterskamp

Department of Computer Science, University of Dortmund, Germany

Abstract. The focus of this paper is on the practical aspects of efficiently resolving inconsistencies when merging probabilistic rule sets. We consider the problem of prioritized merging, when one core knowledge base is to be used without modifications and to be extended by information from other sources. This problem is addressed by our flexible system HEUREKA that aims at restoring consistency by finding those parts of the additional rule bases which are compatible with the core base and are considered most valuable by the user. We give an overview on the methodological framework of the system and describe some details of its main techniques. In particular, HEUREKA offers a convenient interface to inductive probabilistic reasoning on maximum entropy. An example from the domain of auditing illustrates the problem and the practical applicability of our framework.

1 Introduction

When building up a knowledge base for modeling a problem domain, inconsistencies may easily arise during the process of merging information from different sources. For instance, to get a broader view on the problem under consideration and a deeper understanding of the domain, several experts might be asked for their opinions. Although each of them expresses her knowledge consistently, their different perspectives may yield an inconsistent knowledge base when simply joining the corresponding pieces of information. Or, for similar reasons, statistical knowledge discovered by a data mining process might prove incompatible to subjective knowledge extracted from textbooks or practical experiences.

This is all the more true for probabilistic knowledge representation. A probabilistic knowledge base is inconsistent iff there is no probability distribution satisfying it. Methods for eliminating inconsistencies in classical knowledge bases (cf. e.g. [FFJ+00]) do not help much as the inconsistencies arise from the probabilities, the interactions of which are hard to control. For instance, there is no distribution satisfying $\mathcal{R} = \{(B|A)[0.6], (B|\neg A)[0.7], A[0.5], B[0.7]\}$, so \mathcal{R} is nearly useless, as no further knowledge can be derived. In the context of economics, fusion operators for probability distributions have been developed (cf. e.g. [GZ86, CW99]) which lack, however, the transparency of knowledge based methods. An overview on AI approaches to information fusion for purposes such as query answering, or decision making is given in [BH01]. In [KIR04], a method to fuse probabilistic rule bases with reasonable formal properties is described.

The focus of this paper is rather on the practical aspects of efficiently resolving inconsistencies when merging probabilistic rule sets. To be more precise, we will consider the problem of prioritized merging, when one core knowledge base is to be extended by pieces of information from other sources. So, while it is ensured that all rules of the core base will be used without modifications, the methods to be presented in this paper aim at finding those parts of the additional rule bases which are most valuable and compatible with the core base. This is in contrast to the approach described in [KIR04], where all available knowledge is used but possibly weakened. As our main concern is not on formal properties but on flexibility, practical applicability and efficient computations, our optimization criteria make use of heuristics which are, however, clearly related to theoretical foundations. The practical relevance of such heuristics is obvious. Building up a probabilistic knowledge base might be quite expensive, but any extension with incompatible information will make it inconsistent and hence useless, even if the added information is consistent and valuable of its own. We will present several heuristics to solve this problem in an optimal way. All optimization criteria will be made explicit and clear, in order to justify the heuristics and to help the user find the most appropriate solution.

All heuristics have been implemented by making use of the probabilistic expert system SPIRIT [RRK06], which is able to process incomplete, uncertain and subjective knowledge on optimum entropy. SPIRIT features especially efficient data structures and algorithms and provides a class library (the SPIRIT-API), which makes the functionality and data structures available to other applications. We make use of the SPIRIT-API in our system HEUREKA to realize the processing of SPIRIT knowledge bases and to implement an efficient consistency check. A running example will be given to illustrate the problem and to present HEUREKA. Both the heuristics and the implementation go back to [Fin06].

The rest of this paper is organized as follows: In Sec. 2 we start with a motivating example in the context of auditing. In Sec. 3 we describe the structure of our framework, which was designed to automatically eliminate inconsistencies in a very user-orientated way. Section 4 covers the user's options to specify his objectives. In Sec. 5 we exemplarily present two of the most important heuristics of our framework in detail. In Sec. 6 we revert to the auditing example to show the framework's usefulness in practice and conclude in Sec. 7 with perspectives for further work.

2 A Motivating Example from the Domain of Auditing

While performing an audit, the auditor has to estimate the risk that the financial statement of the company has been manipulated. It is common practice that the auditor uses a checklist with appropriate risk indicators, so called *red flags*, to estimate the fraud risk. For each red flag on the checklist, the auditor has to determine whether it is present or absent in the company.

Albrecht and Romney [AR86] have explored the significance of commonly accepted red flags. Their study gives detailed information about the correlation

between the presence of red flags and the presence of management fraud. For each of the 31 analyzed red flags, the study shows how frequently the red flag was observed in cases of manipulated and not manipulated financial statements, respectively. Table 1 shows a short excerpt from that study.

Table 1. Some red flags from the study and their presence in fraud and no-fraud cases

| Red Flag | In Case of Fraud | In Case of No Fraud |
|---|---------------------|------------------------|
| Key Executives with High Personal Debts or Financial Losses | 44 % | 5 % |
| Key Executives Who Continually Rationalize Failures | 40 % | 6 % |
| Poor Staffing of Accounting Department | 44 % | 11 % |
| Significant Related Party Transactions | 50 % | 23 % |

2.1 Building a Knowledge Base

We used the statistical data the study provides for 31 red flags to build up a probabilistic knowledge base. Every red flag corresponds to a binary variable in the knowledge base. Moreover, the binary variable **Fraud** indicates the presence respectively absence of fraud. For each red flag variable, we insert two probabilistic rules into the knowledge base according to the schema which is obvious from Table 1. For instance, $\mathbf{Fraud} \rightsquigarrow \mathbf{KE_Debts} [0.44]$ and $\neg\mathbf{Fraud} \rightsquigarrow \mathbf{KE_Debts} [0.05]$ encode the information from the first line of Table 1.

In this way, we insert two rules for each of the red flags, so we end up with 62 rules that model the relation between red flags and fraud. In addition, the fact $\mathbf{Fraud} [0.50]$ is added to the knowledge base to model the a priori probability of fraud, i.e. the estimated probability of fraud without further knowledge. To simplify matters, we have chosen an a priori probability of 50 % to make the relative probability change easy to read. In principle, a more realistic fraud probability, e.g. 1 %, or other probabilities reflecting subjective beliefs could be applied instead. These 63 rules form the rule set \mathcal{R}_{RF} , which is used as a knowledge base for SPIRIT. As a statistical investigation underlies it, \mathcal{R}_{RF} is consistent. So SPIRIT can be applied to calculate the maximum entropy distribution over the 32 variables, on the basis of which a red flag checklist can be evaluated by instantiating the flags. In particular, the marginal probability of the variable **Fraud** corresponding to the current fraud risk will be computed by SPIRIT.

2.2 Extending the Knowledge Base

A closer look at the 31 red flags reveals that some of these red flags have quite a similar meaning or cover the same circumstances in a more general or more specific way. However, these correlations have not been modeled in the K_{RF} knowledge base yet; just to the contrary, the modeling in K_{RF} makes them conditionally independent given **Fraud**. As a consequence, if these meaning-related red flags are observed together, the fraud risk is often overestimated. To handle

this problem, we have to append some additional rules to the knowledge base, like the following ones:

| | |
|---|--------|
| KE_Greed \rightsquigarrow KE_Low_Moral_Character | [0.60] |
| KE_Living_Beyond_Means \rightsquigarrow KE_Debts | [0.85] |
| No_Internal_Auditing_Staff \rightsquigarrow Inadequate_Internal_Controls | [0.90] |
| Lack_of_Personnels_Policies \rightsquigarrow No_Rules_of_Personal_Conduct | [0.75] |

These rules reflect common sense knowledge or subjective beliefs, respectively, and are called *interdependency rules* here. We composed 20 interdependency rules overall, which form the rule set \mathcal{R}_I . The extended knowledge base, containing the rules $\mathcal{R}_{RF} \cup \mathcal{R}_I$, is called K_{RFI} .

Adding the 20 interdependency rules to the knowledge base aimed at refining statistical information by expert knowledge, but leads to much more complex connections between the variables. The increased complexity causes the K_{RFI} knowledge base to be inconsistent, i.e. there exists no distribution that fulfills all 83 rules of K_{RFI} . As a consequence of this inconsistency, it is not possible to query the knowledge base, since there is no distribution to evaluate a query. Therefore, inference is not feasible and the knowledge base is virtually useless, as long as its rule set is inconsistent.

At this point, the user of the knowledge base has to perform a very difficult and time consuming task: He has to modify the inconsistent rule set in some way to resolve the contradictions among the rules, in order to obtain a modified but consistent rule set. Since there are no hints, which rule has to be modified in what way, the user has to solve quite a complex problem. Even if he succeeds in constructing a consistent rule set, he cannot be sure if there might have been a better way to eliminate the inconsistencies. This means that the user has no clue if the constructed rule set is really the best alternative to the inconsistent one or if another consistent rule set could have been constructed by less “severe” modifications.

In principle, there are three ways to modify a probabilistic rule set in order to resolve inconsistencies. One kind of modification is simply to remove certain rules from the set to eliminate the contradictions caused by these rules. Removing rules results in a loss of knowledge, but the knowledge is not altered in a wrong way. Another kind of modification is to change the prescribed probabilities of some rules. Because of the complex connections between probabilistic rules, even a slight change might be enough to eliminate the inconsistencies. However, modifying the probabilities alters the knowledge expressed by these rules. Whether this modification is still acceptable or it must be seen as a falsification of the knowledge, depends on the degree of modification. A third kind of modification is to change the logical structure of some rules. Changing the logical formula of the premise or the conclusion of a rule involves generally a severe change in the meaning of the rule. Thus, it is applicable only in exceptional cases and must be done with care not to falsify the whole knowledge expressed by the rule set. Hence, this kind of modification is particularly not practical for an automated removal of inconsistencies.

3 Overall Concept of Heureka

HEUREKA is an implemented system that helps building up probabilistic knowledge bases from possibly inconsistent sets of probabilistic rules. Both premises and conclusions of these rules are propositional formulas, and each rule comes along with a probability value which is interpreted as the corresponding conditional probability. HEUREKA is tailor-made for inductive probabilistic reasoning based on optimum entropy [KIR04], as it can be carried out by the aid of SPIRIT, but it can also be used as a stand-alone system helping to restore consistency in probabilistic knowledge bases in an automated but quite flexible and especially user-friendly way. To this aim, HEUREKA offers several algorithms that make use of heuristics which are based on sophisticated techniques. Four central cornerstones make up a frame for HEUREKA (cf. also Fig. 1):

Consideration as an optimization problem: Most importantly, the task of extracting consistent rule sets out of inconsistent knowledge bases is considered as an optimization problem. Every valid consistent rule set provided by a modification procedure represents a solution to this problem. The user can choose the *optimization criterion* (see Sec. 4), which offers him great flexibility when looking for solutions. Regarding the criterion, a total order on the set of solutions is created.

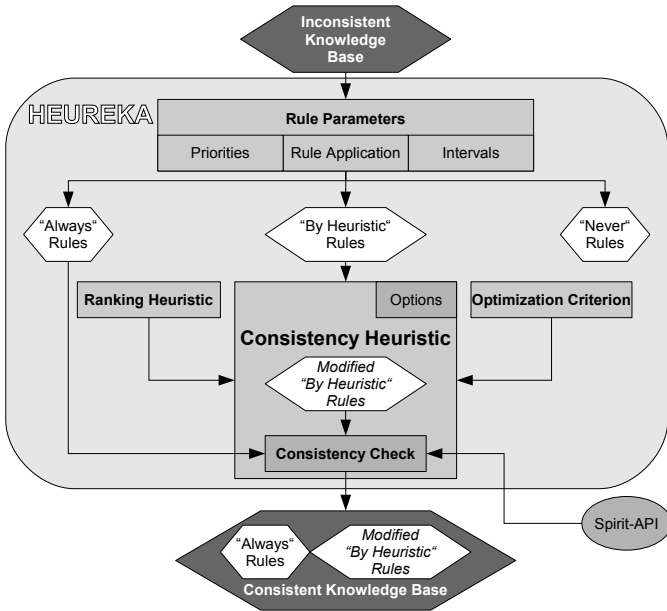


Fig. 1. Components of the framework and their interaction to eliminate inconsistencies

Modular structure of the system of heuristics: The user can use and combine the designed heuristics in a multifaceted way. He is enabled to easily extend them and adapt them to his desires and needs. This is all possible due to the modular structure of the multilayer system of the heuristics. The highest layer consists of the so-called *consistency heuristics* (see Sec. 5), which follow diverse approaches to resolve inconsistencies. An important distinctive feature of these heuristics is the way the inconsistent knowledge base is modified: Solutions can be generated by deleting some rules or by changing the probabilities of rules within preassigned intervals. The consistency heuristics make use of several subheuristics, for example heuristics to solve subproblems (see Sec. 5.1) or to choose subsets of rules (see Sec. 4), all of them can easily be replaced or extended.

Rule parameters: The combination of the heuristics is one option to customize the way the inconsistencies are solved. In order to exert a more detailed influence on how the rules are addressed, the user may specify special parameters of the rules. This will have an effect on the modus operandi of the heuristics and enables the user to define the optimization problem in more detail. The most fundamental of these parameters is the parameter *rule application*, as its options define whether the rule will be taken into consideration for being part of a solution at all. The possible values are “*never*”, “*always*”, and “*by heuristic*”. The first option is mainly used for query rules in SPIRIT. The option “*always*” enforces rules to be part of any solution; e.g., the rules of the core knowledge base are always-rules. The last option leaves the decision to the heuristic. Furthermore, *priority* values between 1 and 10 can be assigned to the rules which can make rules more or less important both for the consistency heuristics and the optimization criteria. Finally, the parameters *lower delta* and *upper delta* are used for defining lower and upper bounds of a probability interval when rule probabilities are to be modified.

Consistency check: The last key concept of our framework is the consistency check, which is crucial for each consistency heuristic in terms of correctness and calculation speed. From the heuristic’s point of view, the consistency check module is a black box accomplishing this test. In turn, the consistency check module does not consider the context under which the check is performed. Thus, our module, which utilizes SPIRIT-API functions, can easily be substituted by another algorithm if desirable.

We will explain the first three of these cornerstones in more detail in the following sections.

4 Optimization Criteria and Ranking Heuristics

In order to meet the particular needs of the user in a most flexible and appropriate way, HEUREKA offers him various optimization criteria to be applied:

Number of Used Rules → max: A solution is considered the better, the more rules it contains. This criterion is very useful if the user is interested in keeping as many rules as possible.

Sum of Priorities → **max**: In this case, the custom priorities, which the user assigned to the rules, are considered. The priorities of the rules in a solution are added up. The higher the sum, the higher is the objective function value of the solution.

Weight → **max**: This criterion gives the user even more flexibility: He can define weighting factors for the weighted sum of the number of the rules and the sum of the priorities. The higher the weighted sum, the better is the solution.

Entropy → **min**: For every solution, the entropy of the appertaining maximum entropy distribution is computed. The lower the entropy, which means the lower the indeterminacy, the better is the solution. This criterion is very valuable if the user is interested in most informative solutions.

An important property of all these optimization criteria is their *monotony*: They do not provide a higher objective function value if a rule is removed from a given solution, or, in turn, adding a rule to a given solution does not decrease the quality of the solution if consistency is maintained. The monotony is a basic prerequisite for the procedural methods within the consistency heuristics, as it assures an efficient calculation of the solutions by focusing on one rule at a time.

Therefore, an ordering on the rules is needed in various places of the consistency heuristics. For this, so-called *ranking heuristics* are used, which select the next rule to be considered. A sophisticated selection may highly increase the performance of the consistency heuristics. Therefore, we made the ranking heuristics exchangeable modules, in order to provide more flexibility and room for improvements. For evaluation purposes, we also implemented ranking heuristics which produce random orders.

5 Consistency Heuristics

We have developed several consistency heuristics, which generate solutions either by removing some rules or by changing the prescribed probability of rules. Besides the kind of modification they perform, the particular consistency heuristics and their underlying strategies differ in terms of solution quality compared to calculation speed. We will describe two of the developed consistency heuristics in this section.

5.1 Consistency Heuristic “Rules Removal: Branch and Bound”

The “Rule Removal: Branch and Bound” consistency heuristic (*B&B Heuristic*) generates solutions by removing certain rules from the original rule set \mathcal{R} , i.e. each solution is a consistent subset of \mathcal{R} . This heuristic proceeds in accordance with the branch-and-bound principle to calculate an optimal solution (in compliance with the selected optimization criterion) as quickly as possible. This means that – besides the calculation of good solutions – this heuristic focuses on proving the optimality of the best solution so far at an early stage. By using the branch-and-bound principle, it is often sufficient to calculate only a small portion of the overall possible solutions in order to determine an optimal solution. Furthermore, the minimum quality of the best solution so far (which is

defined as the value of the best solution so far divided by the value of the best upper bound solution so far) can be determined anytime. W.l.o.g. the following explanations cover the case of using a maximizing optimization criterion.

According to the branch-and-bound principle, the B&B Heuristic partitions a problem S (starting with the root problem, i.e. the inconsistent rule set \mathcal{R}) into two smaller, disjoint subproblems S' and S'' . For each of these subproblems, a lower bound and an upper bound are calculated. Our way of partitioning (cf. Fig. 2) is quite similar to the approach of the well-known branch-and-bound algorithm for the knapsack problem: In each branch step, at first the preassigned ranking heuristic is employed to select a *branching rule*. Next, one smaller subproblem is created by *forcing* the branching rule, and another smaller subproblem is created by *prohibiting* the branching rule. The *forced rules* $\mathcal{R}_{\text{Forc}}(S)$ of a subproblem S must be contained in every solution of S (i.e. none of these rules can be removed to build a solution). Its *prohibited rules* $\mathcal{R}_{\text{Proh}}(S)$, however, must not be contained in any solution (i.e. they have to be removed). The rules of S that are neither forced nor prohibited are called its *free rules* $\mathcal{R}_{\text{Free}}(S)$.

```

Partition( $S$ ) {
  // use the ranking heuristic to determine a branching rule for  $S$ 
  branchRule := RankingHeuristic.select( $\mathcal{R}_{\text{Free}}(S)$ );

  // define a smaller subproblem by force
   $\mathcal{R}_{\text{Forc}}(S') := \mathcal{R}_{\text{Forc}}(S) \cup \{\text{branchRule}\}$ ;
   $\mathcal{R}_{\text{Proh}}(S') := \mathcal{R}_{\text{Proh}}(S)$ ;
   $\mathcal{R}_{\text{Free}}(S') := \mathcal{R}_{\text{Free}}(S) \setminus \{\text{branchRule}\}$ ;

  // define a smaller subproblem by prohibition
   $\mathcal{R}_{\text{Forc}}(S'') := \mathcal{R}_{\text{Forc}}(S)$ ;
   $\mathcal{R}_{\text{Proh}}(S'') := \mathcal{R}_{\text{Proh}}(S) \cup \{\text{branchRule}\}$ ;
   $\mathcal{R}_{\text{Free}}(S'') := \mathcal{R}_{\text{Free}}(S) \setminus \{\text{branchRule}\}$ ;

  // calculate an upper and a lower bound for both  $S'$  and  $S''$ 
  ...
}

```

Fig. 2. Partitioning a subproblem S into two smaller subproblems S' and S''

Upper and Lower Bound Heuristics. The B&B Heuristic uses an upper bound heuristic that provides only a very rough estimation of an upper bound for a subproblem S . If $\mathcal{R}_{\text{Forc}}(S)$ is consistent, then the upper bound value is set to the solution value that the rule set $\mathcal{R}_{\text{UB}}(S) := \mathcal{R}_{\text{Forc}}(S) \cup \mathcal{R}_{\text{Free}}(S)$ would have, if it was a solution (i.e. regardless of the consistency of $\mathcal{R}_{\text{UB}}(S)$). The monotony of the optimization criterions ensures no solution of S can have a higher value than the rule set $\mathcal{R}_{\text{UB}}(S)$.

The real work of the B&B Heuristic (in terms of consistency checks) is done when a lower bound heuristic is employed to calculate the lower bound of a sub-

problem S . We have developed different lower bound heuristics (*LB heuristics*) that can alternatively be employed by the B&B Heuristic. They have all been designed to facilitate the fast calculation of quite a good solution for a given subproblem. However, they differ in terms of calculation speed and solution quality. We want to give a brief description of two of the LB heuristics:

Binary Search: This LB heuristic provides the fastest way to calculate a solution for a given subproblem S . Therefore, it is also utilized by other LB heuristics to calculate an initial solution. The “Binary Search” LB heuristic (cf. Fig. 3) uses the predefined ranking heuristic to generate a sorted rule vector $\mathbf{V}(\mathcal{R}_{\text{Free}}(S))$ containing the free rules of S . The rules are in descending order, hence the rule considered most important for a solution (according to the ranking heuristic) is at the first position in this vector. Similar to a classical binary search strategy, the LB heuristic successively divides the vector $\mathbf{V}(\mathcal{R}_{\text{Free}}(S))$ in two halves. The left half is checked each time for consistency with $\mathcal{R}_{\text{Forc}}(S)$. Once a “left subset” of $\mathbf{V}(\mathcal{R}_{\text{Free}}(S))$ consistent with $\mathcal{R}_{\text{Forc}}(S)$ is found, the “Binary Search” LB heuristic stops since a solution to S has been calculated.

```

LB.BinarySearch( $S$ ) {
  // sort descending to move most import rules to first vector positions
   $\mathbf{V}(\mathcal{R}_{\text{Free}}(S)) := \text{RankingHeuristic.sortDescending}(\mathcal{R}_{\text{Free}}(S));$ 

  // check for inconsistency with forced rulers of  $S$ 
  while ( $\mathbf{V}(\mathcal{R}_{\text{Free}}(S)) \cup \mathcal{R}_{\text{Forc}}(S)$  is inconsistent) {
    // continue with only the left half of the vector
     $\mathbf{V}(\mathcal{R}_{\text{Free}}(S)) := \text{getLeftHalf}(\mathbf{V}(\mathcal{R}_{\text{Free}}(S)));$ 
  }

  // return a consistent subset, i.e. a solution to  $S$ 
  return  $\mathbf{V}(\mathcal{R}_{\text{Free}}(S)) \cup \mathcal{R}_{\text{Forc}}(S);$ 
}

```

Fig. 3. “Binary Search” LB heuristic to calculate a solution of a subproblem S

Binary Search Plus Measuring Contradiction Potential: This LB heuristic uses the “Binary Search” LB heuristic to calculate an initial solution \mathcal{R}_{BS} and then tries to improve \mathcal{R}_{BS} by re-adding some of the previously removed rules. To decide which rules offer a good chance to be re-added to \mathcal{R}_{BS} without causing inconsistency, we developed the following strategy: First, an entropy-optimal distribution Q fulfilling \mathcal{R}_{BS} is computed. Next, for each previously removed rule the (absolute) difference between its prescribed probability and its probability under Q is calculated. This probability difference is considered as a measure for the “contradiction potential” the rule has (relating to the distribution Q). Finally, the rules the contradiction potential of which does not exceed a predefined limit are successively checked for consistency with the

best solution so far for the subproblem (starting with \mathcal{R}_{BS}) and, if applicable, re-added to this solution to further improve it. A more greedy version of this strategy has been implemented, too.

Utilization of a Cache and a Special Cache Strategy. Due to the branch-and-bound approach and its way of constructing subproblems, quite a number of consistency checks for already checked rule sets have to be handled during the execution of the B&B Heuristic. In order to avoid unnecessary and time consuming computations, a subset check is performed which is based on ideas similar to those underlying the AprioriGen-Algorithm for data mining [AMS+96]: Subsets of consistent sets are also consistent, and supersets of inconsistent sets are inconsistent, too. We use an efficient data structure as a cache to look up the results of previously performed consistency checks and pursue the following *Subset Superset Cache Strategy*: If the consistency of a rule set \mathcal{R}_{Chk} has to be checked and a consistent superset $\mathcal{R}_{\text{Con}} \supset \mathcal{R}_{\text{Chk}}$ (inconsistent subset $\mathcal{R}_{\text{Incon}} \subset \mathcal{R}_{\text{Chk}}$) is already in the cache, then \mathcal{R}_{Chk} is consistent (inconsistent), too. Since the cache has been implemented as a hash tree, it can be determined at no cost (especially compared to a consistency check) if an appropriate superset (subset, respectively) is already in the cache.

5.2 Consistency Heuristic “Probability Change: Intervals”

The consistency heuristic “Probability Change: Intervals” (*Interval Heuristic*) follows the approach to eliminate inconsistencies in a rule set \mathcal{R} by modifying the prescribed probabilities of rules. The modified probability $p_{\text{Mod}}(R)$ of each rule $R \in \mathcal{R}$ has to be in a preassigned interval $I_{\text{Lim}}(R)$ making use of the parameters Lower and Upper Delta (see Sec. 3). Assigning these modified probabilities to the rules in \mathcal{R} leads to a consistent rule set \mathcal{R}_{Mod} , which differs from \mathcal{R} only in the probabilities of its rules. Thus, the number of rules and *their propositional logical structure* are identical in both rule sets.

In the following, we will give a brief description of the calculation method we used to determine modified probabilities $p_{\text{Mod}}(R)$ for rules of an inconsistent rule set \mathcal{R} . Each such $p_{\text{Mod}}(R)$ has to respect a specified interval $I(R) := [s(R), t(R)]$. The basic concepts of our *interval calculation*, which forms a fundamental step of the Interval Heuristic, go back to ideas presented by Rödder and Xu in [RX01]. An interval calculation proceeds as follows:

1. Each interval rule $R : A \rightsquigarrow B[s(R), t(R)]$ is replaced by two (fixed-probability) *auxiliary rules*, introducing an *auxiliary variable* W_R for each rule R :

$$R_{\text{Min}} : A \wedge W_R \rightsquigarrow B[s(R)] \quad \text{and} \quad R_{\text{Max}} : A \wedge \neg W_R \rightsquigarrow B[t(R)]$$

2. A consistency check for the set of auxiliary rules is performed and, if successful, an entropy-optimal distribution P^* that fulfills this set is calculated. If the consistency check fails, then even the application of intervals does not lead to a consistent rule set and the interval calculation is terminated.
3. The modified probability $p_{\text{Mod}}(R)$ for each rule R is calculated based on P^* :

$$p_{\text{Mod}}(R) := P^*(B|A) = s(R) \cdot P^*(W_R|A) + t(R) \cdot P^*(\neg W_R|A)$$

4. Assigning these modified probabilities to the rules leads to the consistent rule set \mathcal{R}_{Mod} .

This kind of interval calculation allows us to handle interval rules, but still use the same calculation techniques, especially the consistency check, as for fixed-probability rules. Since our implementation of the consistency check is already based on an entropy-optimal computation, we get the calculation of the entropy-optimal distribution P^* (in step 2) at no cost.

It should be mentioned that our approach differs significantly from other techniques applying the maximum entropy principle plainly to interval probabilities. As maximum entropy chooses probabilities in a most cautious way, the weakest probability within an interval is chosen, i.e. the one value that restores consistency and is closest to 0.5. By handling interval information as it is done by HEUREKA, the modified probability is realized as a mean value, using as much of the available information as possible. In this way, techniques similar to those presented in [KIR04] are applied locally.

The primary goal of the Interval Heuristic is to calculate a solution whose modified probabilities differ as little as possible from the prescribed probabilities. A single execution of an interval calculation based on the preassigned interval $I_{\text{Lim}}(R) := [p_{\text{Pre}}(R) - \Delta_{\text{Low}}(R), p_{\text{Pre}}(R) + \Delta_{\text{High}}(R)]$ of each rule results in a solution whose modified probabilities exploit the whole interval $I_{\text{Lim}}(R)$ in an unnecessary way most times. To avoid this effect, the Interval Heuristic initially uses smaller intervals, whose size is only a fraction of the preassigned intervals $I_{\text{Lim}}(R)$. Then an interval calculation based on these smaller intervals is performed. If it is successful, then it delivers modified probabilities that can only have an accordingly small derivation. Else, the size of the intervals is incremented successively and corresponding interval calculations are performed, until a solution can be calculated. In a more detailed view, the Interval Heuristic proceeds as follows:

During the execution of the Interval Heuristic, every interval calculation is based on the *effective interval* $I(R)$ of each rule R . The effective interval $I(R)$ is called “opened at v %” according to its size. The specific size of an effective interval $I(R)$ opened at v % is defined on basis of the preassigned interval $I_{\text{Lim}}(R)$:

$$I(R) := [p_{\text{Pre}}(R) - v/100 \cdot \Delta_{\text{Low}}(R), p_{\text{Pre}}(R) + v/100 \cdot \Delta_{\text{High}}(R)]$$

When the Interval Heuristic starts, all effective intervals are initially opened at 0 %, i.e. they merely specify a single probability instead of a probability range. During the execution of the Interval Heuristic, the effective interval of each rule is opened in s steps, i.e. after s steps, an effective interval $I(R)$ is opened at 100 % and matches $I_{\text{Lim}}(R)$. The Interval Heuristic performs several *opening steps*. In every opening step, the ranking heuristic determines the rules whose effective intervals are opened one more step (while the effective intervals of all other rules keep their current opening). Then an interval calculation is performed based on the effective intervals of all rules. If the interval calculation provides a solution, the Interval Heuristic ends, since it has successfully calculated a solution. Otherwise, it continues with the next opening step.

6 Heureka – Walkthrough by Example

We have developed the program HEUREKA that implements all afore presented concepts and heuristics. HEUREKA combines these elements with an easy to use graphical user interface. HEUREKA was written in Java and utilizes functions of the SPIRIT-API to perform calculations and other operations on SPIRIT knowledge bases. HEUREKA can load knowledge bases in the SPIRIT file format and eliminate existing inconsistencies in different ways. Solutions (i.e. consistent knowledge bases) that have been calculated can again be saved in the SPIRIT file format to be seamlessly used in the SPIRIT-Shell. In this way, full interoperability is achieved.

Continuing the Auditing Example. We resume the auditing example from the beginning (see Sec. 2.2) to present some of the main features of HEUREKA in an exemplary way. Therefore, we open the (inconsistent) knowledge base K_{RFI} in HEUREKA and configure some of the rule parameters. The parameter “rule application” is left at its default value “by heuristic” for all 20 interdependency rules. However, this parameter is set to “always” for all 63 other rules that arouse from the statistics of the study. This way, we ensure that only the 20 interdependency rules can be changed by any heuristics and that the 63 so-called *always-rules* will be included in every solution without any modifications. We use the lower and upper delta parameters to assign probability intervals (that we considered appropriate) for all interdependency rules. In addition, we (exemplarily) assigned the lowest priority 1 to some rules (instead of the default priority 5).

Figure 4 illustrates the execution of the B&B Heuristic on the knowledge base K_{RFI} . HEUREKA generally shows all options of a consistency heuristic in the left part of the window. Before a consistency heuristic is started, you use these options to select the ranking heuristic, the optimization criterion, etc. that will be employed by the consistency heuristic. The right part of the window documents the execution of a consistency heuristic in detail, so that all steps of the heuristic can easily be traced. Important items for a specific consistency heuristic are displayed (and permanently updated) below that message area (the global bounds, the minimum quality, etc. in case of the B&B Heuristic). Furthermore, the current overall progress of the heuristic is shown.

Using the Consistency Heuristics. In this example, the B&B Heuristic takes about 30 seconds to calculate¹ two solutions that are optimal in respect to the selected optimization criterion (“Number of Used Rules \rightarrow max” in this example). Each of these optimal solutions contains 13 of 20 interdependency rules (besides the 63 mandatory always-rules). Accordingly, seven rules had to be removed in each case to achieve a consistent knowledge base that contains all always-rules.

Due to the heuristic approach, the computation time could take considerably longer. In the worst-case, an exponential number of rule sets have to be checked

¹ All calculation times refer to a Windows XP computer system with an AMD Athlon XP 3000+ CPU running Java 2 Runtime Environment 5.0.

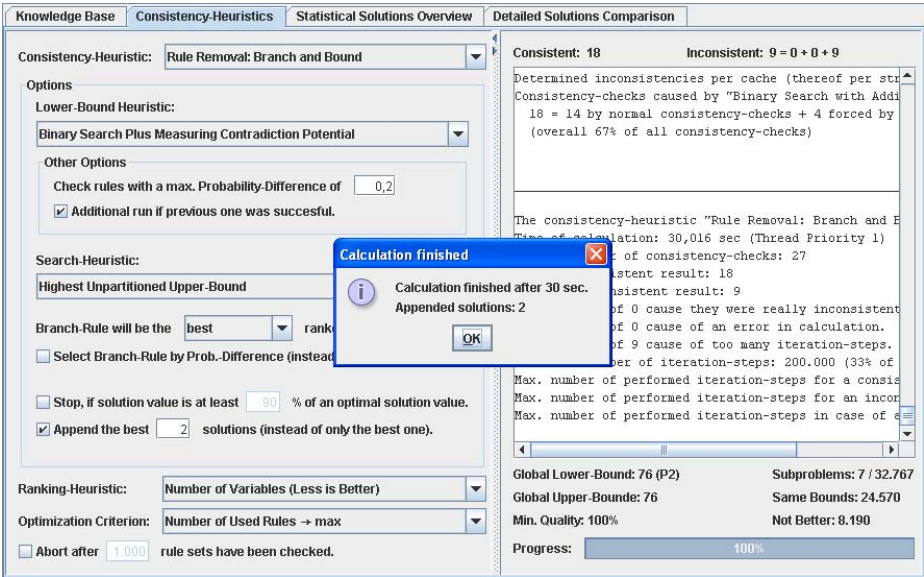


Fig. 4. B&B Heuristic applied to the K_{RFI} knowledge base

to compute an optimal solution. The overall calculation time of the B&B Heuristic depends on the number of consistency checks that have to be performed and the average calculation time for each of these consistency checks. The average time for a consistency check in this example is about 0.1 seconds, whereas for some these rule sets it takes up to 2 seconds to determine their inconsistency. In our tests of the B&B Heuristic, the number of necessary consistency checks for an optimal solution was always considerably below the worst-case number. For some quite adverse rule sets in these tests, the B&B Heuristic had to calculate about 0.4 % of the worst-case consistency checks, which still provided an optimal solution within a few minutes.

Applying the Interval Heuristic to the knowledge base K_{RFI} with regard to the preassigned intervals needs only 30 seconds to calculate a best solution (by opening intervals in $s = 10$ steps). The probabilities of the interdependency rules were modified to achieve a consistent rule set, leaving the probabilities of the always-rules unchanged (as required). Nine of the interdependency rules suffered merely a slight change, so that their absolute deviation is below 0.01. Since the Interval Heuristic generally does not remove rules, that solution consequently contains all 83 rules of K_{RFI} (except for the modified probabilities of some rules).

Compared to the B&B Heuristic, the worst-case calculation time of the Interval Heuristic can quite exactly be estimated, because it directly depends on the maximal number of opening-steps that have to be performed. Since in every opening-step an interval calculation is performed and every interval calculation requires only one consistency check, the worst-case number of (time-consuming) consistency checks can be predetermined and is relatively low compared to those

of an average run of the B&B Heuristic. The Interval Heuristic provides appropriate options to configure the maximum number of opening steps to perform (which influences indirectly the quality of the calculated solution in terms of derivation), so the overall calculation time can be kept at an acceptable level (e.g. less than one minute for rule sets with about 100 rules).

Presenting the Solutions. HEUREKA offers two different ways of presenting its computed solutions: One presentation displays statistical information for each solution (e.g. the number of removed rules or the average derivation of the modified probabilities). These values characterize a solution and hence make it easy to compare different solutions. This enables the user to decide ultimately, what solution is the best alternative to the inconsistent knowledge base (from his point of view). The other presentation provides a very detailed rule-based comparison of solutions. This presentation shows every rule of the original knowledge base and emphasizes each rule's particular modification in each solution. Thus, it becomes obvious what rules have been removed and what probabilities have been modified respectively. In addition, rules that had the parameter "never" or "always" assigned at computation time are marked accordingly, to denote their special treatment by the consistency heuristic. The modified probabilities are highlighted in different colors corresponding to the degree of modification. Thereby the user can identify rules with a severe modification at a glance.

Resuming the Work in the Spirit-Shell. Every presented solution can directly be saved as a SPIRIT knowledge base, in order to be subsequently opened and processed in the SPIRIT-Shell. In our example, the user can decide between three consistent knowledge bases. Each of these knowledge bases enables him to evaluate a red flag checklist in the SPIRIT-Shell. But unlike before, each knowledge base is consistent and includes interdependency rules. Thus, for the first time, the user can observe to what degree the interdependency rules affect the knowledge processing. A practical review of both the solutions with 13 interdependency rules and the solution with 20 (more or less modified) interdependency rules pointed out that the knowledge modeling was significantly improved by adding the interdependency rules (compared to the knowledge base \mathcal{R}_{RF} , which is just based on statistical data). Since the interdependency rules model certain correlations between the red flags now, the overestimation of the fraud risk could considerably be lowered.

7 Conclusion and Further Work

The presented consistency heuristics and their integration in the developed framework form a flexible system for sophisticated inconsistency elimination in probabilistic knowledge bases. The program HEUREKA makes it possible to utilize all developed heuristics and concepts in praxis. Thus, further useful information about dealing with inconsistencies can be obtained. The user can choose between several optimization criteria to specify his objective for a most appropriate solution. The flexibility of the underlying framework allows an easy

integration of new heuristic strategies and further ideas. The practical employment of HEUREKA has pointed out that especially the development of significant rule-based inconsistency measures for probabilistic knowledge bases offers a great potential for further improvements. If such inconsistency measures would be available, they could simply be integrated in our existing system by adding a corresponding ranking heuristic. A ranking heuristic based on a powerful inconsistency measure could immediately improve any existing consistency heuristic by reliably identifying very problematic rules. Therefore, one future goal is to develop sophisticated inconsistency measures for probabilistic knowledge bases. These inconsistency measures shall allow us to draw conclusions about what rules are especially responsible for an existing inconsistency.

References

- [AMS⁺96] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast Discovery of Association Rules, pp. 307–328 (1996)
- [AR86] Albrecht, S.W., Romney, M.B.: Red-flagging management fraud: a validation. *Advances in Accounting* 3, 323–333 (1986)
- [BH01] Bloch, I., Hunter, A., et al.: Fusion: General concepts and characteristics. *International Journal of Intelligent Systems* 16, 1107–1134 (2001)
- [CW99] Clemen, R.T., Winkler, R.L.: Combining probability distributions from experts in risk analysis. *Risk Analysis* 19(2), 187–203 (1999)
- [FFJ⁺00] Felfernig, A., Friedrich, G.E., Jannach, D., Stumptner, M.: Consistency-based diagnosis of configuration knowledge bases. In: *Proceedings of ECAI* (2000)
- [Fin06] Finthammer, M.: Entwicklung und Implementierung von Heuristiken zur Behandlung von Inkonsistenzen in probabilistischen Wissensbasen mit Anwendungen im Bereich der Wirtschaftsprüfung. Master’s thesis, Universität Dortmund (2006)
- [GZ86] Genest, C., Zidek, J.V.: Combining probability distributions: A critique and an annotated bibliography. *Statistical Science* 1(1), 114–135 (1986)
- [KIR04] Kern-Isberner, G., Rödder, W.: Belief revision and information fusion on optimum entropy. *International Journal of Intelligent Systems* (2004)
- [RRK06] Rödder, W., Reucher, E., Kulmann, F.: Features of the expert-system-shell spirit. *Logic Journal of the IGPL* 14(3), 483–500 (2006)
- [RX01] Rödder, W., Xu, L.: Behebung von Inkonsistenzen in der probabilistischen Expertensystem-Shell Spirit. In: *Operations Research Proceedings 2000*, pp. 260–265. Springer, Heidelberg (2001)

Extending Markov Logic to Model Probability Distributions in Relational Domains

Dominik Jain, Bernhard Kirchlechner, and Michael Beetz

Intelligent Autonomous Systems Group
Department of Informatics
Technische Universität München

Abstract. Markov logic, as a highly expressive representation formalism that essentially combines the semantics of probabilistic graphical models with the full power of first-order logic, is one of the most intriguing representations in the field of probabilistic logical modelling. However, as we will show, models in Markov logic often fail to generalize because the parameters they contain are highly domain-specific. We take the perspective of generative stochastic processes in order to describe probability distributions in relational domains and illustrate the problem in this context by means of simple examples.

We propose an extension of the language that involves the specification of a priori independent attributes and that furthermore introduces a dynamic parameter adjustment whenever a model in Markov logic is instantiated for a certain domain (set of objects). Our extension removes the corresponding restrictions on processes for which models can be learned using standard methods and thus enables Markov logic networks to be practically applied to a far greater class of generative stochastic processes.

1 Introduction

In artificial intelligence (AI), a variety of applications can greatly benefit from a unification of logical and probabilistic knowledge representations. The former enable us to deal with complex, relational domains by offering an expressive language, and the latter allow us to soundly handle uncertainty. In any sufficiently complex AI application, both are of utmost importance and need to be fully integrated. Yet for a long time, the development of both strands has been largely separate. Especially in recent years, however, a number of alternative approaches towards a unification have been proposed. Theoretical contributions to the field that has now emerged as statistical relational learning date back to at least Nilsson [1], Halpern [2] and Bacchus [3], while the more practically oriented research has only recently gained momentum [4,5,6,7]. One of the most promising approaches currently available is Markov logic [8,9], as it essentially combines, unlike most other approaches, the full power of first-order logic with probability. It is thus one of the most general, yet it is still supported by a suite of tools that are geared towards practical applicability (the Alchemy system [10]). Because of their exceptional expressivity and simplicity, Markov logic

networks have gained a lot of attention lately, including an invited talk by Pedro Domingos at AAAI-06.

In principle, any language that unifies statistical and relational representations and that is furthermore supported by a number of sufficiently efficient learning and inference algorithms, so that it can truly be applied in practice, is not only inherently appealing but can also serve as an interface layer between learning and inference on the one side and higher-level AI applications on the other [11], increasing the interoperability between implementations on both sides of the layer if it is established as a standard. Markov logic is a prime candidate for a representation language on which such an interface layer could be based.

1.1 Application Scenarios

There are countless applications for probabilistic relational representations and the corresponding interface layer. Consider, for example, a system such as ASPO-GAMO [12], which observes football games, recognizes and classifies the ball actions that occur within them, determines the parameters of the actions, characterizes the situations in which the actions are performed and analyzes the effects of the actions. In the context of such an application, we could use the data that is collected to build comprehensive models of the game process, which could then be used to answer a wide range of queries automatically. For instance, we might ask for the probability that a pass played by a certain type of player succeeds in a given situation — or the probability that a ball passed by a side midfielder is generally received by a striker if it is played with specific parameters (speed, direction) in a given situation.

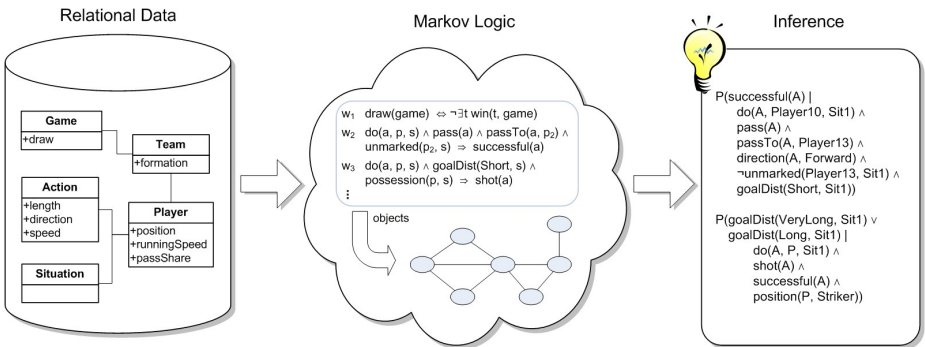


Fig. 1. Applying Markov logic

Figure 1 depicts a schematic application of Markov logic that could be capable of answering such queries. It shows the typical process of learning a model from relational data and reasoning about it. The source data could, for example, be given in a relational database. Ideally, we would then only need to provide a set

of formulas that hold in the domain and learn the model’s parameters in order to be able to query the model by instantiating it for a concrete set of objects and calculating the conditional probabilities we are interested in.

There are several obvious characteristics of such applications and the representations they need in order to perform their tasks. First, the models created and used must represent uncertainty: football games are notoriously non-deterministic, for results of actions are caused by the interplay of situations and actions that are only incompletely characterized, and important information for analysis, such as the intentions of players, cannot be inferred. Second, the questions we ask essentially make use of the power of natural language: They ask about classes of players, infer information about the relation of actions’ effects and the situations they are performed in, and the domains of events we query are dynamically determined. Whatever the concrete application, a probabilistic relational representation that is able to represent full-joint distributions over any given domain of a certain type is desirable, for complex queries cannot be answered otherwise.

1.2 Contributions

In this paper, we lay down a number of properties of probabilistic logical representation languages that are essential for the modelling of some aspects of probabilistic relational domains. In particular, the language must be capable of describing general principles about multiple objects having similar properties, which should then be applicable in several contexts, i.e. in several different domains. Markov logic networks that are learned using standard methods in many cases do not possess some of these properties and consequently fail to represent the intended probability distribution when we move from one domain to another, i.e. when we perform a *domain shift*.¹ Because Markov logic seems to have been used mainly in rather specialized applications, the problem we describe may have been hitherto unnoticed. We illustrate the problem using particularly simple examples and subsequently propose an extension of the language that solves it. In essence, our solution involves the formulation of hard constraints on probabilities that are used to dynamically calculate some of the probabilistic parameters of models in Markov logic that would otherwise remain fixed.

In the following section, we thus begin by stating the properties of representation languages we deem desirable. Next, in Sect. 3, we briefly introduce Markov logic networks and define the problems that limit their applicability to relational domains. Subsequently, in Sect. 4, we show why Markov logic networks are, under certain conditions, unable to handle domain shifts and, in Sect. 5, we suggest a corresponding extension of the language, which we explain in detail. Finally, we summarize our results and provide an outlook on future research.

¹ Note that the term *domain* is used at two different levels. At the higher level, we mean the general scenario that a model deals with, i.e. a specification of the types of objects, their attributes and the relations that are considered. At the lower level, which is relevant in this particular case, we mean a concrete set of constants referring to objects in the world (i.e. instances of the types declared at the higher level).

2 Demands on the Representation Language

We believe that, beyond the ability of handling uncertainty as well as a high degree of complexity, a probabilistic logical language should fulfill a number of additional requirements. In this paper, we take the perspective of probabilistic knowledge representation in relational domains, i.e. we seek to model probability distributions over relational data. A common way to view a probability distribution is to see it as the result of a generative stochastic process. Correspondingly, the goal of probabilistic logical modelling is simply to obtain an accurate model of the underlying process. In a relational setting, the process typically generates objects with certain attributes as well as relations between objects according to a set of rules. The representation language should allow us to concisely describe these rules. In this context, the aforementioned criteria that the representation language and its associated learning and inference mechanisms should ideally fulfill are:

1. It should be possible for a domain expert to specify his/her knowledge in a straightforward, declarative way. The addition of new, relevant knowledge should lead to an improvement of the corresponding model (or at least leave the model unchanged); and the failure to represent specific aspects of a domain should not render the model useless with respect to aspects untouched by the omission.
2. It should be possible to unambiguously define a probability distribution over arbitrary domains (of a certain type) by characterizing the corresponding stochastic process. In particular, a model should be universally valid, specifying arbitrarily general rules that may be independent of concrete objects. The probability model should generalize to arbitrary domains and arbitrary objects within them — much in the same way as universally quantified formulas in first-order logic that deal with objects of a certain type are applicable to arbitrary objects of these particular types.

When working with Markov logic, issues related to the first point can be observed — in the sense that additional, relevant formulas may negatively affect certain queries in an unforeseeable fashion — but we found this to be a result of a problem that is more closely related to the second point, for which we provide a solution further on. In Chap. 3.2, we point out the cause of the problem, and in Sect. 4, we analyze it in more detail, providing some examples. But first, we lay the necessary groundwork by introducing Markov logic networks.

3 Markov Logic Networks

Markov logic networks (MLNs) are probabilistic logical models that combine the semantics of probabilistic graphical models (namely Markov networks) with the full power of first-order logic. An MLN can be seen as a set of constraints on the set of possible worlds that is implicitly defined by a set of logical predicates and a set of constants, as each logical atom that can be constructed using these

domain elements is viewed as a boolean variable. Specifically, the constraints are formulas in first-order logic with attached numeric weights that quantify their hardness.

Formally, a *Markov logic network* L is a set of pairs (F_i, w_i) , where F_i is a formula in first-order logic and w_i is a real number, the weight of formula F_i . Together with a finite set of constants C , an MLN defines a Markov network $M_{L,C}$, the *ground Markov network*, as follows:

1. $M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in the formulas of the Markov logic network L .
2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L . The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is w_i .

The ground Markov network's set of variables X is the set of ground atoms that is implicitly defined by the predicates in the MLN and the set of constants C . The Markov logic network specifies a probability distribution over the set of possible worlds \mathcal{X} , i.e. the set of possible assignments of truth values to each of the ground atoms in X , as follows,

$$\begin{aligned}
 P(X = x) &= \frac{1}{Z} \cdot \exp \left(\sum_i w_i \cdot n_i(x) \right) \\
 &= \frac{\exp(\sum_i w_i \cdot n_i(x))}{\sum_{x' \in \mathcal{X}} \exp(\sum_i w_i \cdot n_i(x'))} \tag{1}
 \end{aligned}$$

where the inner sums are over indices of MLN formulas and $n_i(x)$ is the number of true groundings of the i -th formula in possible world x .

3.1 Inference

Since $\textcircled{\text{I}}$ provides the full-joint distribution over the variables in X , it can be used to compute arbitrary conditional probabilities: The probability that a formula F_1 holds given that formula F_2 does can be computed as

$$\begin{aligned}
 P(F_1 \mid F_2, L, C) &= P(F_1 \mid F_2, M_{L,C}) = \frac{P(F_1 \wedge F_2 \mid M_{L,C})}{P(F_2 \mid M_{L,C})} \\
 &= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2}} P(X = x)}{\sum_{x \in \mathcal{X}_{F_2}} P(X = x)} \\
 &= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2}} \exp(\sum_i w_i \cdot n_i(x))}{\sum_{x \in \mathcal{X}_{F_2}} \exp(\sum_i w_i \cdot n_i(x))} =: \frac{W_{F_1 \wedge F_2}}{W_{F_2}} \tag{2}
 \end{aligned}$$

where \mathcal{X}_{F_i} is the set of possible worlds in which F_i holds, and $W_{F_1 \wedge F_2}$ and W_{F_2} are the sums of exponentiated sums of weights for possible worlds where $F_1 \wedge F_2$ holds and where F_2 holds respectively (this is a notation that we continue to use further on).

3.2 Problems

Since a Markov logic network is not (necessarily) specific to concrete domain elements but is instead designed to be applicable to arbitrary domains over the classes of objects that it models, MLNs should satisfy the generality requirement made above. However, with the current set of concepts in place, it is in many cases not possible to learn the characteristics of the respective generating processes from data, because parameter learning in MLNs is an ill-posed problem, and the solution that is obtained is usually specific to the concrete set of objects that were used for learning. An MLN obtained via parameter learning cannot, without restrictions, be applied to a domain with a different number of objects than the one it was learned with.

The weights in an MLN are usually learned using MAP estimation or, in the absence of a prior distribution over parameter settings, maximum likelihood — i.e. they are chosen in such a way that the probability of a training database, which specifies the truth values of ground atoms for one particular domain, is maximized. Yet clearly, there can be more than one generative stochastic process that could have produced any given training database, and obviously, a process cannot be uniquely identified through a single sample taken from it (such as a training database). Unfortunately, the MLN language itself is not sufficient in order to make the necessary distinctions prior to parameter learning, for it lacks the ability to specify unconditional independencies in terms of structure; in many cases, the MLN language therefore does not allow us to adequately characterize the concrete process we are dealing with when the weights are yet unknown.

When applying an MLN whose weights were learned using a particular training database is applied to a different domain, it is assumed that the parameters that adequately describe the concrete distribution observed in the training data also fully characterize the process that generated it. This assumption, however, is rarely justified. And whenever the assumption is indeed not justified, the MLN models the desired probability distribution only for a single instantiation, namely the training database. It is thus no more useful than the corresponding ground Markov network, and the general, relational character of the model is essentially lost.

4 Analyzing the Problem

Let us consider a simple example to explain why this is the case. We first introduce the domain we will use for our experiments before turning to the problem of domain shifts in detail.

4.1 The Example Domain

In our example domain, we consider the simplest of scenarios where there are two types of objects and a relation connecting them. Suppose the concrete types of objects are people and drinks, and that there exists a relation that captures the consumption of drinks. Both people and drinks are characterized by a single attribute that classifies them: Each person has a rank (either *Student* or *Professor*),

and each drink has a type (either *Tea* or *Coffee*). In an MLN, both attributes can be represented using appropriate predicates, e.g. $rank(person, rankvalue)$ and $drinkType(drink, typevalue)$.² Now suppose there is a difference in the drinking habits of students and professors; professors might, for example, consume more drinks than students do and students might not drink any coffee at all. In general, we can describe arbitrary drinking habits in a Markov logic network simply by including the following conjunctions³

$$\begin{array}{ll}
 w_1 & consumed(p, d) \wedge rank(p, Student) \quad \wedge \quad drinkType(d, Tea) \\
 w_2 & consumed(p, d) \wedge rank(p, Student) \quad \wedge \quad drinkType(d, Coffee) \\
 w_3 & \neg consumed(p, d) \wedge rank(p, Student) \quad \wedge \quad drinkType(d, Tea) \\
 w_4 & \neg consumed(p, d) \wedge rank(p, Student) \quad \wedge \quad drinkType(d, Coffee) \\
 w_5 & consumed(p, d) \wedge rank(p, Professor) \quad \wedge \quad drinkType(d, Tea) \\
 w_6 & consumed(p, d) \wedge rank(p, Professor) \quad \wedge \quad drinkType(d, Coffee) \\
 w_7 & \neg consumed(p, d) \wedge rank(p, Professor) \quad \wedge \quad drinkType(d, Tea) \\
 w_8 & \neg consumed(p, d) \wedge rank(p, Professor) \quad \wedge \quad drinkType(d, Coffee)
 \end{array}$$

which exhaustively define the various cases for each possible consumption of a drink by a person. The weight w_i of each conjunction describes, when viewed relative to the weights of the other conjunctions, how likely the respective case really is.

4.2 Domain Shifts

The most important observation in this context is that if the only formulas the MLN includes are the above conjunctions, then the marginal distributions of people's ranks and drink types are fully determined by the drinking habits. People are less likely to have a certain rank if people of that rank are less likely to (not) consume drinks. Especially in the case of ranks, a perhaps more intuitive model would state that the marginal distribution of ranks (when we know nothing about consumptions) is independent of *potential* consumptions and that it is in fact rather the ranks that lead to a specific drinking behaviour. During parameter learning, this is, however, a distinction that is not being made, since we obtain any one of infinitely many weight vectors that accurately represent the distribution present in the training data but which generalize to variable-size domains in different ways.

For example, let us look at a single person, say P , and the drinks that P can potentially have consumed. Let the set \mathcal{X}_m contain the set of possible worlds for a domain with only one person P and m drinks. The probability of P being a student is, following (2), given by

² Note that when modelling the attributes in this way, it is necessary to include constraints that define the attribute values as mutually exclusive and exhaustive. Henceforth, we silently assume that, for each attribute, the corresponding constraints are included in each model.

³ We adopted the convention of using lower-case letters as variables and words beginning with upper-case letters as constants. Any free variables in the formulas are (implicitly) universally quantified.

$$\begin{aligned}
p_m &= P_{\mathcal{X}_m}(\text{rank}(P, \text{Student})) \\
&= \frac{f(m, w_1, w_2, w_3, w_4)}{f(m, w_1, w_2, w_3, w_4) + f(m, w_5, w_6, w_7, w_8)}
\end{aligned} \tag{3}$$

with

$$\begin{aligned}
&f(m, x_1, x_2, x_3, x_4) \\
&= \sum_{t=0}^m \sum_{c_t=0}^t \sum_{c_c=0}^{m-t} \binom{m}{t} \binom{t}{c_t} \binom{m-t}{c_c} \exp(x_1 c_t + x_2 c_c + x_3(t - c_t) + x_4(m - t - c_c))
\end{aligned} \tag{4}$$

where t is the number of teas, c_t is the number of teas that are consumed and c_c is the number of coffees that are consumed. If there is an asymmetry in the impact of the weights for students and professors, then p_m is not independent of m . For example, if $w_2 = -100$ and all other weights are 0 (i.e. students hardly ever drink coffee but all other consumption events are equally likely), then $p_1 \approx \frac{4-1}{8-1} \approx 0.4286$, $p_2 \approx \frac{16-7}{32-7} \approx 0.3600$ and $p_3 \approx \frac{64-37}{128-37} \approx 0.2967$. The probability that any given person is a student thus decreases as the number of drinks in the domain increases, which was to be expected, because the worlds in which P is a student and consumes even one cup of coffee are (virtually) impossible, and the fraction of such worlds increases with the number of drinks. So depending on the number of drinks, the weight vector apparently determines a different marginal for $\text{rank}(p, \text{Student})$. However, the generating process might dictate a more intuitive view, where the marginal probability of a certain rank is independent of the number of drinks; it might, for example, state that the marginal probability of a person being a student is $\frac{1}{3}$.

Let us consider ways in which we might address the problem. A perhaps straightforward formula to add to the MLN would be a unit clause such as $\text{rank}(p, \text{Student})$, which can obviously be used to influence the probabilities of worlds in which there are students and hence the marginal probability of the rank Student . In fact, we could argue that a unit clause is the only candidate formula for such a correction, because unit clauses are the only formulas that affect *only* the marginal distribution we are interested in and nothing else. Unfortunately, the impact of the unit clause's weight does not depend on the number of drinks, so a correction for arbitrary domains is not possible. This is evident from the fact that the sum of exponentiated sums of weights for worlds where P is a student would only be scaled by a factor of $\exp(w)$ if w was the weight of the newly introduced formula $\text{rank}(p, \text{Student})$. Therefore, we cannot obtain a size-invariant marginal distribution for P 's rank when using an asymmetric configuration of weights of the eight conjunctions, because the unit clause can only have the desired effect for a fixed number of objects; and unfortunately, the weight configurations obtained via standard parameter learning are not ensured to be symmetric.⁴ The inclusion of the unit clause can, however, result in precisely the correction that we want for a *fixed* number of drinks. Yet obviously, our generality requirement is not met.

⁴ By symmetric, we here mean a symmetric impact of the weights, such that the resulting distribution is a uniform distribution.

Whenever we learn the parameters of an MLN that contains at least one proper relation connecting objects of different types, we will usually obtain very accurate weights for the concrete set of objects found in the training database (provided that our formulas are sufficiently expressive), yet the weights will not generalize to other domains if we impose the requirement that certain marginals remain invariant. In particular, if a relation exists, then the unit clause’s weight on its own gives no indication of the probability of the corresponding attribute, for its purpose is primarily to counterbalance the effects of the relation — and these effects depend on the number of tuples that can be constructed.

To further illustrate this problem, let us compare a Markov logic network, which we can instantiate on demand, to several Bayesian networks that we can specifically create for each concrete domain (set of objects) according to a well-defined schema (as we might describe it in, for example, a Bayesian logic program [6]). As MLN formulas, we simply use the eight conjunctions describing consumption behaviour listed above as well as unit clauses for *rank* and *drinkType*, plus a rule that states that a drink cannot be consumed by more than one person. To learn the MLN’s parameters, i.e. the formulas’ weights, we use a training database containing one student (who drank one cup of tea) and two professors (where the first drank one cup of tea and one cup of coffee and the other drank just one cup of coffee). Let L be the resulting Markov logic network — obtained via maximum likelihood parameter learning.

Let us now perform inferences in several ground Markov networks based on L . We compare results obtained for several domains: By $D_{n,m}$ we denote a domain where the set of people contains n elements, $\{P_1 = P, \dots, P_n\}$, and the set of drinks contains m elements, $\{D_1, \dots, D_m\}$.⁵ In particular, we will look at $D_{1,1}$ and $D_{1,3}$; Figure 2 shows the corresponding Bayesian networks, in which we assume that the marginal distributions of both ranks and drink types are to remain fixed.

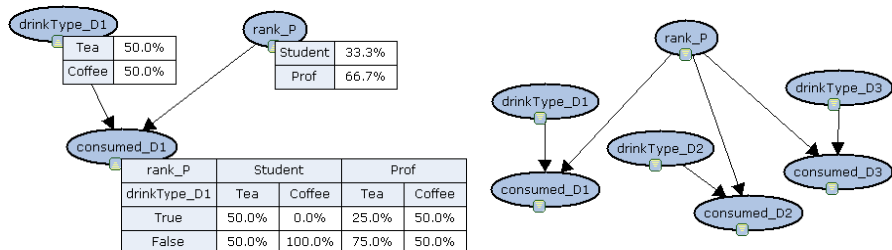


Fig. 2. Bayesian Networks $B_{D_{1,1}}$ (left) and $B_{D_{1,3}}$ (right). The conditional probability tables in $B_{D_{1,3}}$ are the same as in $B_{D_{1,1}}$.

⁵ Note that we here assume a typed predicate logic, where the set of constants is not a mere set but rather an ordered set of sets containing one set of objects for each class.

Comparing the results in Table III, we gather that the probabilities computed using the Markov logic network are clearly dependent on the number of objects (as expected). Only for domain $D_{3,4}$, which contains precisely the number of objects found in the training database, does the MLN compute the desired probabilities. There is, unfortunately, no formula that we could have added to the MLN in order to encode the invariants of the generative stochastic process that we imagine to have created the training database.

Table 1. Inference results: ground Markov networks and Bayesian networks compared

| | $M_{L,D_{1,1}}$ | $B_{D_{1,1}}$ | $M_{L,D_{1,3}}$ | $B_{D_{1,3}}$ | $M_{L,D_{3,4}}$ |
|---|-----------------|---------------|-----------------|---------------|-----------------|
| $P(\text{drinkT}(D_1, \text{Tea}))$ | 68.3% | 50.0% | 68.3% | 50.0% | 50.0% |
| $P(\text{rank}(P, \text{Stud}))$ | 29.3% | 33.3% | 32.0% | 33.3% | 33.3% |
| $P(\text{rank}(P, \text{Stud}) \mid \text{cons}(P, D_1))$ | 30.1% | 25.0% | 32.7% | 25.0% | 25.0% |
| $P(\text{rank}(P, \text{Stud}) \mid \neg \text{cons}(P, D_1))$ | 29.0% | 37.5% | 31.6% | 37.5% | 37.5% |
| $P(\text{rank}(P, \text{Stud}) \mid \text{cons}(P, D_1) \wedge \text{drinkT}(D_1, \text{Tea}))$ | 45.4% | 50.0% | 48.5% | 50.0% | 50.0% |
| $P(\text{drinkT}(D_1, \text{Tea}) \mid \text{cons}(P, D_1) \wedge \text{rank}(P, \text{Prof}))$ | 51.8% | 33.3% | 51.8% | 33.3% | 33.3% |
| $P(\text{cons}(P, D_1) \mid \text{drinkT}(D_1, \text{Tea}) \wedge \text{rank}(P, \text{Prof}))$ | 25.0% | 25.0% | 25.0% | 25.0% | 25.0% |
| $P(\text{rank}(P, \text{Stud}) \mid \text{cons}(P, D_1) \wedge \text{cons}(P, D_2))$ | | | 33.5% | 18.2% | 18.2% |

4.3 Domain-Specific Modifications

In fact, the set of formulas included above would in theory already be sufficient to represent precisely the size-invariant probability distribution that we want, since all the formulas are conjunctions as they would appear in an MLN translated from a model obtained via knowledge-based model construction (KBMC) [8]. (Such an MLN contains a conjunction of literals for each entry of each conditional probability table in the KBMC model — the conjunction capturing the parent-child configuration that corresponds to the entry and the weight being the logarithm of the probability value.) We still fail to find the set of weights that will generalize in the intended way, because, as mentioned previously, parameter learning is an ill-posed problem: There are infinitely many weight vectors that represent the same probability distribution given a specific domain/set of objects, yet only a subset of these weight vectors generalizes in the intended way to domains of variable size.

Notice that in an MLN derived from a KBMC model, the distributions over attribute values for which unit clauses exist would be uniform distributions if the unit clauses themselves were removed from the MLN; therefore, there is no dependence on domain size. Starting with a weight vector obtained via conversion from a KBMC model, we can, however, construct infinitely many weight vectors such that the probability distribution remains unchanged for a concrete number of objects but changes to varying degrees as we change the number of objects with which the model is instantiated. The idea behind the construction is that if the weight of any unit clause was to be modified, the weights of other formulas within which the unit clause appears could be adjusted to compensate for the modification of the unit clause in such a way that the probability distribution remains unchanged for a specific number of objects. Yet clearly, any modification of a unit clause’s weight prevents the remaining formulas from generating a

uniform distribution for an arbitrary number of objects. Hence size-dependence results.

In an MLN L derived from a KBMC model, the set of formulas can be partitioned into classes of mutually exclusive and exhaustive formulas, as for each set of conjunctions representing a certain conditional distribution, there is exactly one true grounding among all the groundings of the conjunctions with the same variable bindings. Let C be the set of classes minus the classes that contain only unit clauses. Now if the weight of a unit clause F_j (an atom that makes a statement about an attribute of an object of some type T) was to be changed by adding an arbitrary Δw in a modified MLN L' , we could choose an arbitrary non-empty subset $S \subseteq \{C_i \in C \wedge \text{contains}(C_i, F_j)\}$ of the classes that contain formulas that contain F_j and adjust certain formula weights to cancel out the previous change. In particular, for a concrete domain D , we apply to each $(F_i, w_i) \in C_k$ where F_i contains F_j and $C_k \in S$ the following modification,

$$w'_i = w_i - \Delta w \cdot \frac{1}{n_{C_i, D} \cdot \frac{1}{|D_T|}} \cdot \frac{1}{|S|} \quad (5)$$

where $n_{C_i, D}$ is the number of groundings of each of the conjunctions in class C_i for domain D and D_T is the set of domain elements of type T in D .

We can easily show that the probability of any possible world x defined over the domain D remains unchanged by our modification,

$$\begin{aligned} P_{M_{L', D}}(x) &= \frac{1}{Z'} \cdot \exp\left(\sum_i w'_i \cdot n_i(x)\right) \\ &= \frac{1}{Z'} \cdot \exp\left(\sum_i w_i \cdot n_i(x) + \Delta w \cdot n_j(x) - \right. \\ &\quad \left. \sum_{C_i \in S} \Delta w \cdot \frac{1}{n_{C_i, D} \cdot \frac{1}{|D_T|}} \cdot \frac{1}{|S|} \cdot \frac{n_{C_i, D}}{|D_T|} \cdot n_j(x)\right) \\ &= \frac{1}{Z} \cdot \exp\left(\sum_i w_i \cdot n_i(x)\right) = P_{M_{L, D}}(x) \end{aligned} \quad (6)$$

as for each object O of the $n_j(x)$ objects for which the unit clause F_j is true, there are $\frac{n_{C_i, D}}{|D_T|}$ combinations for grounding all the variables appearing in each of the formulas in C_i except the one variable we assume to be bound to O , and for each combination, there is exactly one true grounding of a formula with a modified weight.

Therefore, provided that we instantiate ground Markov networks only for domain D , L' thus yields exactly the same probability distribution as L . So when learning parameters using a training database over domain D , L and L' are both optimal solutions, since the way in which the model should generalize to other domains is not considered.

5 Extending Markov Logic

The problem described above essentially arises only because the learning process has no knowledge of fixed marginal distributions or unconditionally independent attributes. While we can modify the learning process to learn weights that are equivalent to the weights we would obtain via a translation from a KBMC model (simply by precomputing the probability distributions of attributes that are subject to a fixed marginal distribution from the training database, using the logarithms of the probabilities as the initial weights of the corresponding unit clauses during learning and ensuring that the weights remain constant throughout the optimization process), this approach requires the MLN to contain conjunctions (or equivalent formulas) that are fully capable of describing an appropriate factorization of the full-joint, which may be impractical.

We now propose an extension of the MLN language that allows us to ensure fixed marginal distributions regardless of the set of formulas. It involves the specification of hard constraints on individual formula probabilities. In our example on the consumption of drinks, we might, for example, specify a constraint such as $P(\text{rank}(p, \text{Student})) = 0.3$. When instantiating a ground Markov network, the probability information can be used to dynamically modify the weight of the corresponding unit clause $F := \text{rank}(p, \text{Student})$. If the probability that is indicated by the model is originally q and the constraint requires it to be q' , then with W_F (see (2)) as the sum of exponentiated sums of weights for possible worlds in which the formula holds (for an arbitrary binding of the variable p), $W_{\neg F}$ as the sum for the remaining worlds and $q = \frac{W_F}{W_F + W_{\neg F}}$ and $q' = \frac{W_F \cdot \lambda}{W_F \cdot \lambda + W_{\neg F}}$, we obtain

$$\lambda = \frac{q'}{q} \cdot \frac{1 - q}{1 - q'} \quad (7)$$

i.e. we need to add $\log(\lambda)$ to the formula's weight in order to obtain the desired marginal probability.

If the underlying process dictates more than one such constraint on the marginal distributions of certain attributes, we can proceed in a similar fashion. However, specifying just a probability constraint for each corresponding unit clause may not suffice, because the a priori independence of the attributes may need to be modelled explicitly. For example, if the marginal distribution over drink types, too, was to be fixed in all domains, adding a constraint such as $P(\text{drinkType}(d, \text{Tea})) = 0.5$ would not render ranks and drink types independent. Therefore, if independence is to be modelled, we instead propose to add to the MLN all conjunctions of value statements for all independent attributes⁶ in order to indirectly represent the independence by including the corresponding part of the full-joint. In our example, we would add the constraints

$$\begin{aligned} P(\text{rank}(p, \text{Student}) \wedge \text{drinkType}(d, \text{Tea})) &= q_1 = q_S \cdot q_T \\ P(\text{rank}(p, \text{Student}) \wedge \text{drinkType}(d, \text{Coffee})) &= q_2 = q_S \cdot q_C \\ P(\text{rank}(p, \text{Professor}) \wedge \text{drinkType}(d, \text{Tea})) &= q_3 = q_P \cdot q_T \\ P(\text{rank}(p, \text{Professor}) \wedge \text{drinkType}(d, \text{Coffee})) &= q_4 = q_P \cdot q_C \end{aligned}$$

⁶ For conjunctions not already present, we initially assume a zero weight.

where $\sum_i q_i = 1.0$ (in our above example, $q_S = \frac{1}{3}$, $q_P = \frac{2}{3}$, $q_T = q_C = 0.5$)⁷ Naturally, the actual inclusion of these constraints could be handled by the learner, and the user would need to specify only which marginal distributions are to remain fixed. All the necessary probabilities can then automatically be computed from the training database.

In general, if m attributes are to have a fixed marginal distribution and the domain of the j -th attribute contains d_j elements, then there are $n := \prod_{j=1}^m d_j$ conjunctions for which probability constraints must be specified (we denote these conjunctions by C_i with $i \in \{1, \dots, n\}$). Because these conjunctions are atomic (sub-)events (for a particular binding of the variables), they partition the set of possible worlds \mathcal{X}_D for a concrete domain D , for which the MLN is instantiated, into n corresponding parts, i.e. $\mathcal{X}_D = \bigsqcup_{i=1}^n (\mathcal{X}_D)_{C_i}$. Let the sum of exponentiated sums of weights of possible worlds in the i -th partition be $W_i := W_{C_i}$; the normalizing constant in (II) is thus $Z_D = \sum_{i=1}^n W_i$. If \mathbf{q} is the vector of conjunction probabilities, then we obtain the scaling factors λ_i with which the weights of the n conjunctions need to be corrected by solving the following linear equation system:

$$\begin{aligned} \frac{W_k \lambda_k}{\sum_{i=1}^n W_i \lambda_i} &= q_k \quad \text{for } k \in \{1, \dots, n\} \\ \Rightarrow (W_k - q_k W_k) \lambda_k - \sum_{i=1, i \neq k}^n q_k W_i \lambda_i &= 0 \quad \text{for } k \in \{1, \dots, n\} \end{aligned} \quad (8)$$

If the probabilities that are specified in vector \mathbf{q} are not contradictory (which the learning process can guarantee), then the solution to the above equation system is unique and well-defined, and we obtain the desired marginals by adding $\log(\lambda_i)$ to the weight of the i -th conjunction C_i .

Since the part of the joint probability distribution that consists exclusively of marginals is fully determined by the above calculations, any existing MLN formulas that contain only atoms that are subject to constraints on marginals cannot provide additional information as far as the probability distribution is concerned. Therefore, they can be removed prior to the above calculations, and the formulas we explicitly require (i.e. the conjunctions C_i), along with their weights ($\log(\lambda_i)$), can be added automatically when instantiating a ground Markov network.

All in all, we can apparently model arbitrary marginal distributions and independencies. Together with a set of conditional distributions, we can thus describe a wide range of probability distributions in relational domains.

In general, we can show that probabilities conditioned on all marginals are unaffected by the dynamic weight modifications described above. Let R be the set of ground atoms for which the marginal distribution is explicitly modelled in an extension to a Markov logic network L , and let \bar{L} be the Markov logic network we obtain from L by applying the constraints for a concrete domain/

⁷ One of the four constraints is redundant and may be left out.

set of constants C , i.e. by adding the formulas $F_{R=r}$ with weights w_r for $r \in \mathbb{B}^{|R|}$. We thus need to show that $P(F_1 \mid F_2, R = r, M_{\bar{L}, C}) = P(F_1 \mid F_2, R = r, M_{L, C})$:

$$\begin{aligned} P(F_1 \mid F_2, R = r, M_{\bar{L}, C}) &= \frac{P(F_1 \wedge F_2 \wedge R = r \mid M_{\bar{L}, C})}{P(F_2 \wedge R = r \mid M_{\bar{L}, C})} \\ &= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2} \cap \mathcal{X}_{R=r}} \exp(\sum_i w_i \cdot n_i(x) + w_r)}{\sum_{x \in \mathcal{X}_{F_2} \cap \mathcal{X}_{R=r}} \exp(\sum_i w_i \cdot n_i(x) + w_r)} \\ &= \frac{W_{F_1 \wedge F_2 \wedge R=r} \cdot \exp w_r}{W_{F_2 \wedge R=r} \cdot \exp w_r} = P(F_1 \mid F_2, R = r, M_{L, C}) \quad (9) \end{aligned}$$

Returning to our example from Chap. 4.2, we now apply dynamic modifications (8) to the MLN that was learned, enforcing a fixed marginal on *rank* and *drinkType* by introducing the corresponding constraints but leaving the MLN unchanged otherwise. Looking at Table 2, we observe that the generative stochastic process which we assumed can now clearly be described in the intended way.

Table 2. Inference results: ground Markov networks (with dynamic modifications applied) and Bayesian networks compared

| | $M_{\bar{L}, D_{1,1}}$ | $B_{D_{1,1}}$ | $M_{\bar{L}, D_{1,3}}$ | $B_{D_{1,3}}$ |
|---|------------------------|---------------|------------------------|---------------|
| $P(\text{drinkT}(D_1, \text{Tea}))$ | 50.0% | 50.0% | 50.0% | 50.0% |
| $P(\text{rank}(P, \text{Stud}))$ | 33.3% | 33.3% | 33.3% | 33.3% |
| $P(\text{rank}(P, \text{Stud}) \mid \text{cons}(P, D_1))$ | 25.0% | 25.0% | 25.0% | 25.0% |
| $P(\text{rank}(P, \text{Stud}) \mid \neg \text{cons}(P, D_1))$ | 37.5% | 37.5% | 37.5% | 37.5% |
| $P(\text{rank}(P, \text{Stud}) \mid \text{cons}(P, D_1) \wedge \text{drinkT}(D_1, \text{Tea}))$ | 50.0% | 50.0% | 50.0% | 50.0% |
| $P(\text{drinkT}(D_1, \text{Tea}) \mid \text{cons}(P, D_1) \wedge \text{rank}(P, \text{Prof}))$ | 33.3% | 33.3% | 33.3% | 33.3% |
| $P(\text{cons}(P, D_1) \mid \text{drinkT}(D_1, \text{Tea}) \wedge \text{rank}(P, \text{Prof}))$ | 25.0% | 25.0% | 25.0% | 25.0% |
| $P(\text{rank}(P, \text{Stud}) \mid \text{cons}(P, D_1) \wedge \text{cons}(P, D_2))$ | | | 18.2% | 18.2% |

6 Conclusion

We have shown that Markov logic networks that are learned using standard parameter learning may require additional language constructs if they are to be applicable to arbitrary domains — and not just the single domain that was used for learning — whenever the attributes of related objects are a priori not to be affected by the probability with which they are potentially related to other objects, i.e. whenever there are constraints on marginal probabilities of attribute values. This is in fact a quite common case, for the processes that we intuitively imagine generate worlds sequentially, i.e. there is a certain order in which objects and relations are created (which could be akin to a causal structure). If objects and the attributes that belong to them are created simultaneously, then there is usually some attribute whose values are subject to a fixed marginal. A typical example of such a case is an attribute such as a person’s *gender* or *rank* in our above example. Our extension to the MLN language solves this problem by introducing hard constraints on formula probabilities that are used to dynamically modify the weights of the respective formulas whenever a model is instantiated for a concrete domain.

The concrete constraints that are necessary to handle domain shifts need not be specified by a user but can instead be learned from a training database, given that information on unconditional independencies is provided. If we learn with more than one training database, then we can even attempt to deduce these independencies and alleviate the user from specifying any additional information at all.

Nevertheless, there are further problems that may still prevent MLNs from being practically applicable if the goal is to model probability distributions in relational domains. The use of exact parameter learning algorithms based on log-likelihood is intractable, and we found the approximate learning methods involving pseudo-likelihood to be too inexact to be acceptable — at least for some domains. Solving the problems that remain in this regard will be the subject of our future investigations.

Acknowledgements

We would like to thank Pedro Domingos for fruitful email discussions.


References

1. Nilsson, N.J.: Probabilistic Logic. *Artif. Intell.* 28, 71–87 (1986)
2. Halpern, J.Y.: An analysis of first-order logics of probability. In: *Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence*, Detroit, US, pp. 1375–1381 (1989)
3. Bacchus, F.: *Representing and Reasoning with Probabilistic Knowledge*. MIT Press, Cambridge (1990)
4. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: *IJCAI*, pp. 1300–1309 (1999)
5. Milch, B., Marthi, B., Russell, S.J., Sontag, D., Ong, D.L., Kolobov, A.: BLOG: Probabilistic Models with Unknown Objects. In: *IJCAI*, pp. 1352–1359 (2005)
6. Kersting, K., Raedt, L.D.: Bayesian Logic Programming: Theory and Tool. In: Getoor, L., Taskar, B. (eds.) *An Introduction to Statistical Relational Learning*, MIT Press, Cambridge (2005)
7. Neville, J., Jensen, D.: Dependency networks for relational data. In: *ICDM 2004*, pp. 170–177. IEEE Computer Society, Los Alamitos (2004)
8. Richardson, M., Domingos, P.: Markov Logic Networks. *Mach. Learn.* 62(1-2), 107–136 (2006)
9. Domingos, P., Richardson, M.: Markov Logic: A Unifying Framework for Statistical Relational Learning. In: *Proceedings of the ICML 2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*, pp. 49–54 (2004)
10. Kok, S., Singla, P., Richardson, M., Domingos, P.: The Alchemy system for statistical relational AI (2004), <http://alchemy.cs.washington.edu/>
11. Domingos, P.: What’s Missing in AI: The Interface Layer. In: Cohen, P. (ed.) *Artificial Intelligence: The First Hundred Years*, AAAI Press (2006)
12. Beetz, M., Gedikli, S., Bandouch, J., von Hoyningen-Huene, N., Kirchlechner, B., Perzylo, A.: Visually Tracking Football Games Based on TV Broadcasts. In: *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)* (2007)

A Multilingual Framework for Searching Definitions on Web Snippets

Alejandro Figueroa and Günter Neumann

Deutsches Forschungszentrum für Künstliche Intelligenz - DFKI,
Stuhlsatzenhausweg 3, D - 66123, Saarbrücken, Germany
{figueroa, neumann}@dfki.de

Abstract. This work  presents **Mdef-WQA**, a system that searches for answers to definition questions in several languages on web snippets. For this purpose, **Mdef-WQA** biases the search engine in favour of some syntactic structures that often convey definitions. Once descriptive sentences are identified, **Mdef-WQA** clusters them by *potential senses* and presents the most relevant phrases of each *potential sense* to the user. The approach was assessed with TREC and CLEF data. As a result, **Mdef-WQA** was able to extract descriptive information for all definition questions in the TREC 2001 and 2003 data-sets.

1 Introduction

In recent years, search engines have considerably improved their power of indexing in response to the constantly increasing number of documents on the Internet and the growing need of users for smarter ways of searching and presenting the information. Nowadays, one pressing need is to find definitions of concepts. High-performance search engines, such as Google, provide hence a feature which helps users to retrieve definitions from specialised online resources like WordNet and Wikipedia. Google is additionally urged to supply an interface of Wikipedia in other languages, in order to satisfy users all around the world.

Google relies upon the coverage and the high cachet of these specialised resources, especially upon the fact that the first sentence they provide is extremely likely to yield a definition. Unfortunately, this coverage tremendously varies over languages. For instance Wikipedia contains more than 1700000 articles in English whereas about 220000 in Spanish. Further, Google does not make allowances for the redundancy on the responses (i. e. “*George Bush*” in English). Furthermore, Google provides undesirable definitions for some well-known concepts, for example “*George Bush*” in German. Moreover, Google does not present to the user definitions grouped by their respective senses (i. e. “*Tesla*”).

During the last years, the problem of finding definitions for a specific concept (the *definiendum*) has been addressed by Question Answering Systems (QASs)

¹ The work presented here was partially supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project HyLaP (FKZ: 01 IW F02) and the EC-funded project QALL-ME.

in the context of the Text REtrieval Conference (TREC) and the Cross Language Evaluation Forum (CLEF). In TREC, QASs answer definition questions in English, such as “*What is a quasar?*”, by extracting as much as possible non-redundant descriptive information (‘nuggets’) about the *definiendum* from the ACQUAINT corpus.

In order to discover definition utterances, definition QASs usually align sentences with surface patterns in the target corpus at the word and/or the part-of the speech level [5]. Hence, the probability of matching sentences increases as long as the size of the target collection grows, and accordingly, the performance substantially improves [6]. Along with surface patterns, definition QASs take advantage of wrappers around online resources, WordNet glossaries and web snippets [1]. In addition, QASs, like Google, have also shown that definition web-sites are a fertile source of descriptive information in English, in particular, providing answers to 42 out of 50 TREC–2003 questions [1]. However, web snippets have not yet proven to be a valuable source of descriptive phrases so far [1].

Full documents have also been used for extracting definitions. For example, in [10], 250-characters long windows that convey a definition are obtained from the top 50 documents fetched by an IR engine. The windows were then ranked by a *Support Vector Machine*, which was trained using previously tagged windows according to the criteria of [6], and some automatically acquired phrasal attributes. This system obtained one acceptable definition within the top-five ranked windows for 116 out of 160 TREC–2000 questions and 116 out of 137 TREC–2001 questions.

However, TREC focuses its attention solely on English, whereas CLEF aims essentially at European Languages. In the context of CLEF, surface patterns have also shown to be useful for recognising descriptive sentences in other languages. For instance, the best system in CLEF–2005 answered 40 out of the 50 definition questions in the Spanish track by means of surface patterns [13,11].

QASs normally tackle redundancy by: (a) randomly removing one sentence from every pair that shared more than 60% of their terms [5], or (b) filtering out candidate sentences by ensuring that their cosine similarity to all previously selected utterances is below a threshold. It is also worth to remark that definition QASs have not yet made effort to deal with the disambiguation of the different senses of the *definiendum*.

Our contribution

Unlike current definition QASs or search engines, we propose a QAS (named **Mdef-WQA**) that extracts descriptive phrases *directly* from web snippets by rewriting the prompted query in such a way that the probability of aligning surface patterns with web snippets increases (i. e. snippets from specialised definition web-sites like Wikipedia). Since **Mdef-WQA** bases its search on the efficiency of surface patterns and its coverage on the entire web, we show that the framework of **Mdef-WQA** is applicable to *several languages*, in particular English and Spanish. Moreover, we present a novel approach to cluster descriptive utterances according to *potential senses*, which are used to provide a partition of the

most relevant and diverse utterances to the user. **Mdef-WQA** was evaluated in detail using the TREC and CLEF data-sets. The results show that **Mdef-WQA** is promising for answering definition questions in several languages directly from web snippets. In particular, **Mdef-WQA** found out descriptive information for all definition questions in the TREC 2001 and 2003 data sets.

2 Mining the Web for Definitions

Like [10], **Mdef-WQA** receives the *definiendum* δ as input, assuming that it is previously identified by an external query analysis module or entered by the user. Analogously, **Mdef-WQA** receives the language ζ of the original query Q , because it cannot be inferred directly from δ , especially for proper names (i. e. “*John Kennedy*”). **Mdef-WQA** proceeds then as follows:

1. **Mdef-WQA** uses δ and ζ for **rewriting** Q according to a set Π^ζ of pre-defined surface patterns for ζ . These generated queries are then submitted to the search engine. This **rewriting** boosts the retrieval of descriptive utterances by biasing the search engine in favor of sentences that match Π^ζ . Hence, **Mdef-WQA** **avoids** the implementation of **specialised wrappers** and **downloading full documents**, contrary to the trend of current definition QASs.
2. **Mdef-WQA** aligns these patterns with sentences in fetched snippets. Due to its complex internal structure [12], δ might match the *definiendum* δ' only partially within the retrieved descriptive utterances. **Mdef-WQA** recognises δ by means of relaxed pattern matching based on the *Jaccard Measure*. The motivation for using this relaxed matching strategy is that it provides **Mdef-WQA** with a higher degree of language independence compared to current definition QAS. In particular, we avoid the specification of additional word addition/ordering rules [12] or the integration of more sophisticated linguistic processing such as chunking [5].
3. **Mdef-WQA** groups sentences by *potential senses*, which are discovered by **observing the partitions generated by the closest neighbours** of δ in the reliable semantic space supplied by Latent Semantic Analysis (LSA). LSA supplies of language independent framework for drawing semantic inferences.
4. **Mdef-WQA** takes advantage of a variation of Multi-Document Maximal Marginal Relevance [4] for reducing redundancy and maximising diversity in selected utterances. This guarantees a fast summarisation framework which only makes use of a language-specific stop-list.

2.1 Obtaining Descriptive Sentences

In recent years, surface patterns for English have proven to be useful for distinguishing definition utterances in natural language texts [12,10,5,6,7]. These surface patterns provide syntactic structures that are properly aligned with sentences in order to detect descriptive utterances. The syntactic structures are, more precisely, based largely upon punctuation and words that often convey

Table 1. Surface Patterns for English (Π^{en})

| |
|---|
| π_1^{en} : δ' [is are has been have been was were] [a the an] η' |
| e.g., “ Noam Chomsky is a <u>writer and critic...</u> ” |
| π_2^{en} : $[\delta' \eta']$, [a an the] [η' \delta'] [, .] |
| e.g., “ The new iPod , an <u>MP3-Player</u> ,...” |
| π_3^{en} : δ' [become became becomes] η' |
| e.g., “ In 1957 , Althea Gibson <u>became the...</u> ” |
| π_4^{en} : δ' [which that who] η' |
| e.g., “ Joe Satriani who <u>was inspired to play...</u> ” |
| π_5^{en} : δ' [was born] η' |
| e.g., “ Alger Hiss was born <u>in 1904 in USA...</u> ” |
| π_6^{en} : $[\delta' \eta']$, or [η' \delta'] |
| e.g., “ Sting , or <u>Gordon Matthew Sumner</u> ,...” |
| π_7^{en} : $[\delta' \eta']$ [[, .]][also is are] [called named nicknamed known as] [η' \delta'] |
| e.g., “ Eric Clapton , <u>nicknamed 'Slowhand'</u> ...” |
| π_8^{en} : $[\delta' \eta']$ ($[\eta' \delta']$) |
| e.g., “ The United Nations (<u>UN</u>)..” |

definitions. Simply put, these syntactic structures make available the way to identify the *definiendum* δ' and its definition nugget η' within utterances.

Mdef-WQA takes advantage of these syntactic structures not only for distinguishing definitions, but also for biasing the search engine in favor of web snippets that convey definitions. Table 1 shows surface patterns that we found to be particularly useful for this purpose. From this manually specified set of patterns, Mdef-WQA automatically generates the following set of ten different queries used by the search engine. The first submission q_1 corresponds to “ δ ”, and the next four queries aims at π_1^{en} :

q_2 : “ δ is a ” \vee “ δ was a ” \vee “ δ were a ” \vee “ δ are a ”
 q_3 : “ δ is an ” \vee “ δ was an ” \vee “ δ were an ” \vee “ δ are an ”
 q_4 : “ δ is the ” \vee “ δ was the ” \vee “ δ were the ” \vee “ δ are the ”
 q_5 : “ δ has been a ” \vee “ δ has been an ” \vee “ δ has been the ” \vee “ δ have been a ” \vee “ δ have been an ” \vee “ δ have been the ”

π_1^{en} is split into four queries, because it retrieves many descriptive utterances. The next query q_6 attempts to discover snippets that match π_2^{en} or π_6^{en} :

q_6 : “ δ , a ” \vee “ δ , an ” \vee “ δ , the ” \vee “ δ , or ”

The reason to merge these two patterns into one query is two-fold: (a) π_6^{en} has a low occurrence within web snippets (see also [7]), and (b) π_6^{en} often yields a synonym of δ (i. e. “*myopia*, or *nearsightedness*”). Alternative names of persons, organisations or abbreviations are seldom expressed in this way, but are likely to match the other clauses within q_6 . Consequently, the combination of both patterns helps Mdef-WQA to reduce the number of search calls. The queries q_7 , q_8 and q_9 aim at π_7^{en} , π_3^{en} and π_4^{en} respectively as follows:

q_7 : (“ δ ” \vee “ δ also ” \vee “ δ is ” \vee “ δ are ”) \wedge (called \vee nicknamed \vee “known as”)
 q_8 : “ δ became ” \vee “ δ become ” \vee “ δ becomes ”
 q_9 : “ δ which ” \vee “ δ that ” \vee “ δ who ”

Finally, q_{10} : “ δ was born ” \vee “(δ)” attempts to fetch snippets that match π_5^{en} and π_8^{en} . Similarly to q_6 , **Mdef-WQA** merges both patterns into one query on the ground that π_5^{en} deals with δ regarding persons and π_8^{en} focuses basically on acronyms [7]. Hence, **Mdef-WQA** avoids an unproductive retrieval without diminishing the number of fetched descriptive sentences.

Surface patterns for English have been studied widely, especially in TREC, whereas patterns for other languages have been systematically explored only in the context of the CLEF campaigns. Until 2005, CLEF focused exclusively on definition questions aiming at abbreviations and the position of persons [9,13]. These surface patterns are therefore specialised for recognising this specific sort of descriptive information. Systems in TREC are encouraged in extracting as much as possible useful descriptive information about δ [5]. Thus, these surface patterns provide a wider coverage than patterns known for other languages.

For the particular case of surface patterns for Spanish, two additional issues complicates the identification of descriptive information from the web. Firstly, the patterns are based largely upon punctuation signs [11] and closed class words [3], which are usually ignored by some search engines. Secondly, these punctuation signs and closed class words tend to be separated by a large span of text, which usually contains δ' and/or its respective definition η' . Therefore, supplying syntactic structures seems to be unsuitable for rewriting the query. An illustrative example is the pattern “*El* η' , δ' , *se*”, which matches sentences such as “*El presidente de España, Jose Luis Zapatero, se...*”. The snippets obtained by the respective query rewriting “*El*” \wedge “, δ , *se*” are unlikely to yield definitions, and additionally, portions of the large span of text between δ and the closed class word “*El*” can be replaced with an intentional break (often denoted by ...) by the search engine.

All things considered, **Mdef-WQA** seeks to explore whether the translation of surface patterns from English to Spanish provide a wider coverage, and whether they are more efficient for retrieving sentences that convey definitions from the web. Table 2 shows the respective translations of the first five patterns π_p^{en} to Spanish. The translations of π_6^{en} and π_7^{en} as well as some translations of π_3^{en} were not taken into account, because we found them to be unlikely to occur within web snippets. π_8^{en} , which actually does not need any translation, was deliberately omitted for two reasons: it is commonly used by systems in CLEF for resolving abbreviations [11], and one of the motivations behind our research is measuring the contribution of the translated patterns.

From table 2 it can also be observed that pattern π_1^{es} generates 60 cues (e.g., “*es la*”, “*es lo*”, “*son una*”), in contrast to its homologous π_1^{en} , which brings about 18 cues. This substantial increase is due to the fact that Spanish is morphologically richer than English causing a decisive impact on the form and number of queries that **Mdef-WQA** must submit to the web. **Mdef-WQA** necessarily needs to regulate the trade-off between recall and retrieval time. Thus, it is unfeasible

Table 2. Surface Patterns for Spanish (Π^{es})

π_1^{es} : δ' [es|son|fueron|fue|ha sido|han sido] [la|lo|el|un|una|uno|unos|unas|las|los] η'
 e.g., “**Jose Luis Zapatero es el relevo de Felipe Gonzalez para los socialistas.**”

π_2^{es} : δ' [,:;] [un|una|uno|la|lo|el|los|las] η' [,:;:]
 e.g., “**Silvio Rodriguez, uno de los exponentes de la Nueva Trova cubana,...**”

π_3^{es} : δ' [ha llegado a ser|llego a ser|se transformo|se ha transformado] η'
 e.g., “**España se ha transformado en un país democrático.**”

π_4^{es} : δ' [,:] [el cual|la cual|los cuales|quien|que] η'
 e.g., “**Michelle Bachelet quien es la primera presidenta de la historia de Chile,...**”

π_5^{es} : δ' [nacio|fue fundado|fue fundada] η'
 e.g., “**Jose Luis Rodriguez Zapatero nacio en Valladolid el 4 de Agosto de 1960.**”

Table 3. Generated queries for Spanish

q_1 : “ δ ” q_{11} : “ δ es una” \vee “ δ fue lo” \vee “ δ ha sido un”
 q_2 : “ δ , fue un” \vee “ δ son lo” \vee “ δ , la” q_{12} : “ δ se transformo” \vee “ δ fue uno” \vee “ δ , las”
 q_3 : “ δ fue la” \vee “ δ es el” \vee “ δ son el” q_{13} : “ δ la cual” \vee “ δ , una” \vee “ δ ha sido una”
 q_4 : “ δ que” \vee “ δ son las” \vee “ δ , lo” q_{14} : “ δ es uno” \vee “ δ nacio” \vee “ δ el cual” \vee “ δ , los”
 q_5 : “ δ es un” \vee “ δ ha llegado a ser” \vee “ δ son la” \vee “ δ fueron las”
 q_6 : “ δ fue el” \vee “ δ son unas” \vee “ δ , uno” \vee “ δ ha sido la”
 q_7 : “ δ quien” \vee “ δ los cuales” \vee “ δ , un” \vee “ δ son una”
 q_8 : “ δ se ha transformado” \vee “ δ es lo” \vee “ δ fue fundado”
 q_9 : “ δ , el” \vee “ δ son unos” \vee “ δ fue una” \vee “ δ fue fundada”
 q_{10} : “ δ es la” \vee “ δ llevo a ser” \vee “ δ ha sido el” \vee “ δ son un”

to send each cue individually to the web or to follow a criteria similar to the one used for designing the queries for English, because of the number of cues and the fact that they do not present any usefull disjunction. Consequently, the next three key aspects were considered for designing the queries (table 3): (a) cues that are more likely to retrieve descriptive utterances are distributed in different queries, and some unproductive combinations in π_1^{es} are discarded, (b) cues aiming at different tenses and genders were also spread over different queries; this way Mdef-WQA decreases the number of fruitless retrievals, and (c) the number of clauses in a query is limited by the length of queries accepted by search engines.

Once all snippets are fetched Mdef-WQA removes all orthographic accents and splits them into sentences by means of intentional breaks and a sentence splitter². Patterns are then applied to discriminate descriptive utterances within retrieved snippets. Since δ does not exactly match δ' , Mdef-WQA takes advantages of the *Jaccard Measure* for distinguishing more reliable descriptive sentences. The *Jaccard Measure* J of two terms w_i, w_j is the ratio between the number of different *uni-grams* that they share, and the total number of different *uni-grams*: $J(w_i, w_j) = \frac{|w_i \cap w_j|}{|w_i \cup w_j|}$. Consider for example the *definiendum* $\delta^* = \text{“John$

² We are using the one provided by JavaRAP, cf. <http://www.comp.nus.edu.sg/~qiul/NLPTools/JavaRAP.html>.

Kennedy”, which might also be expressed as $\delta_1'^*$ = “*John Fitzgerald Kennedy*” or $\delta_2'^*$ = “*Former US President Kennedy*”. The values for $J(\delta^*, \delta_1'^*)$ and $J(\delta^*, \delta_2'^*)$ are $\frac{2}{3}$ and $\frac{1}{5}$ respectively. **Mdef-WQA** filters reliable descriptive utterances by means of a pattern specific threshold, avoiding additional purpose-built hand-crafted rules and ad-hoc linguistic processing. Of course, some sentences containing useful nuggets will be discarded, but these discarded nuggets can also be found in other retrieved phrases, e.g., “*Former US President Kennedy*” in “*John Fitzgerald Kennedy was a former US President.*”. In short, **Mdef-WQA** trusts implicitly in the redundancy of the web for discovering several paraphrases.

2.2 Potential Senses Identification

There are many-to-many mappings between names and their concepts. On the one hand, the same name or word can refer to several meanings or entities. On the other hand, different names can indicate the same meaning or entity. To illustrate this, consider the next set S of recognised descriptive utterances:

1. John Kennedy was the 35th President of the United States.
2. John F. Kennedy was the most anti-communist US President.
3. John Kennedy was a Congregational minister born in Scotland

In these sentences, “*US President John Fitzgerald Kennedy*” is referred to as “*John Kennedy*” and “*John F. Kennedy*”, while “*John Kennedy*” indicates also a Scottish Congregational minister. In the scope of this work, a *sense* is one meaning of a word or one possible reference to a real-world entity.

Mdef-WQA disambiguates senses of δ by observing the correlation of its neighbours in the reliable semantic space provided by LSA. This semantic space is constructed from the term-sentence matrix M , which considers δ as a *pseudo-sentence* which is weighted according to the traditional *tf-idf*. **Mdef-WQA** builds the dictionary of terms W from normalised elements in S , which consists of uppercasing, removal of html-tags, and the isolation of punctuation signs. **Mdef-WQA** distinguishes then all possible different *n-grams* in S together with their frequencies. The size of W is then reduced by removing *n-grams*, which are substrings of another equally frequent term. This reduction allows the system to speed up the computation of M as UDV' using the *Singular Value Decomposition*. Furthermore, the absence of syntactical information of LSA is slightly reduced by considering strong local syntactic dependencies.

Mdef-WQA makes use of \hat{D} , the greatest three eigenvalues of D , and the corresponding three vectors \hat{U} and \hat{V} for constructing the semantic space as $R = \hat{U}\hat{D}^2\hat{U}'$. **Mdef-WQA** prefers the dot product above the traditional cosine as a measure of the semantic relatedness $R(w_i, w_j) = \hat{u}_i\hat{D}^2\hat{u}_j'$ ($\hat{u}_i, \hat{u}_j \in \hat{U}$) of two terms $w_i, w_j \in W$. The major reasons are (a) it was observed experimentally that, because of the size of web snippets (texts shorter than 200 words), the cosine draws an unclear distinction of the semantic neighbourhood of δ , bringing about spurious inferences [15], and (b) the length of vectors was found to draw a clearer distinction of the semantic neighbourhood of δ as this biases R in favour of contextual terms, which LSA knows better [2].

In this semantic space, the neighbourhood of a particular word w_i provides its context [28]. Consequently, it determines its right meaning by pruning, for instance, inappropriate senses [8]. Similarly, δ is also a term defined by its neighbourhood in this semantic space. For this reason, **Mdef-WQA** selects a set $\bar{W} \subseteq W$ of the forty highest closely related terms to δ , that is, terms that are likely to define its meaning. However, as a result of the relaxed pattern matching, **Mdef-WQA** must also account for all n -grams $\delta^+ \in W$ in δ , because some internal n -grams could be more likely to occur within descriptive utterances (i.e., names or surnames are more frequent than their respective full names). In our working sentences and illustrative variations of δ , “*Kennedy*” has a higher frequency than “*John Kennedy*”. **Mdef-WQA** considers therefore the forty highest pairs $\{w_i, R_{max}(\delta, w_i)\}$, where $R_{max}(\delta, w_i) = \max_{\delta^+ \in W} R(\delta^+, w_i)$. **Mdef-WQA** normalises terms in \bar{W} according to:

$$\hat{R}(\delta, w_i) = \frac{R_{max}(\delta, w_i)}{\sum_{\forall w_j \in \bar{W}} R_{max}(\delta, w_j)}$$

Since words that indicate the same sense co-occur, **Mdef-WQA** identifies *potential senses* by finding a set $\bar{W}^\lambda \subseteq \bar{W}$ of words, for which their vectors form an orthonormal basis. In order to discriminate these orthonormal terms, **Mdef-WQA** builds a term-sentence matrix Φ , where a cell $\Phi_{is} = 1$, if the term $w_i \in \bar{W}$ occurs in the descriptive phrase $S_s \in S$, zero otherwise. The degree of correlation amongst words in \bar{W} across S is then given by $\hat{\Phi} = \Phi\Phi'$. For example, for the words in \bar{W} : $w_1 = \text{“Scotland”}$, $w_2 = \text{“President”}$ and $w_3 = \text{“35th”}$, the computed values for Φ and $\hat{\Phi}$ are:

$$\Phi = \begin{pmatrix} & S_1 & S_2 & S_3 \\ w_1 & 0 & 0 & 1 \\ w_2 & 1 & 1 & 0 \\ w_3 & 1 & 0 & 0 \end{pmatrix} \quad \hat{\Phi} = \begin{pmatrix} & w_1 & w_2 & w_3 \\ w_1 & 1 & 0 & 0 \\ w_2 & 0 & 2 & 1 \\ w_3 & 0 & 1 & 1 \end{pmatrix}$$

Hence, the number of non-selected words $w_j \in \bar{W} - \bar{W}^\lambda$ that co-occur with a term $w_i \in \bar{W}$ across S is given by:

$$\gamma(w_i) = \sum_{\forall w_j \in \bar{W} - \bar{W}^\lambda: \hat{\Phi}_{ij} > 0} 1$$

In our working example, $\gamma(w_1) = 1$ and $\gamma(w_2) = \gamma(w_3) = 2$, because “*President*” and “*35th*” co-occur in S_1 , and “*Scotland*” does not co-occur with any other element of \bar{W} . Then, **Mdef-WQA** adds the w_i to \bar{W}^λ that:

$$\max_{w_i \in \bar{W}} \gamma(w_i) \tag{1}$$

subject to:

$$\hat{\Phi}_{ij} = 0, \quad \forall w_j \in \bar{W}^\lambda \tag{2}$$

$$\gamma(w_i) > 0 \tag{3}$$

In words, a term w_i signals a new sense, if it does not co-occur at the sentence level with any other already selected term $w_j \in \bar{W}^\lambda$, and it has the highest number of co-occurring non-selected terms $w_j \in \bar{W}$. Incidentally, **Mdef-WQA** breaks ties by randomly selecting a term. In our illustrative example, if w_3 is randomly selected, then $\gamma(w_i)$ is equal to one for the three words in the next cycle. w_1 is then selected, because w_3 was already selected and w_2 co-occurs with w_3 ($\hat{\Phi}_{23} > 0$), and accordingly, \bar{W}^λ is {"Scotland", "35th"}. Words are added to \bar{W}^λ until no other term w_i fulfils conditions (2) and (3). Next, sentences are divided into clusters C_λ according to terms in \bar{W}^λ . Sentences that do not contain any term in \bar{W}^λ are collected in a special cluster C_0 . For our working example, the clusters are: $C_0 = \{S_2\}$, $C_1 = \{S_3\}$ and $C_2 = \{S_1\}$.

Finally, **Mdef-WQA** attempts to reassign each sentence S_s in C_0 by searching for the strongest correlation between its named entities (NEs) and the NEs of a cluster C_λ :

$$\max_{C_\lambda} \sum_{\forall e \in S_s} freq_{C_\lambda}(e) > 0, \quad \lambda \neq 0$$

where $freq_{C_\lambda}(e)$ is the frequency of NEs e in the cluster C_λ . The assumption here is that the same NEs tend to occur in the same sense. To illustrate this, S_2 is assigned to C_2 .

2.3 Redundancy Removal

For each cluster C_λ , **Mdef-WQA** determines incrementally a set Θ_λ of its sentences S_λ to maximise their comparative relevant novelty:

$$\max_{S_s \in S_\lambda - \Theta_\lambda} coverage(S_s) + content(S_s)$$

subject to:

$$coverage(S_s) \geq \psi^* > 0 \tag{4}$$

$$W_{type}(S_s) = 0 \tag{5}$$

The comparative relevant novelty of a sentence S_s is given by the relative coverage and content of its nuggets respecting Θ_λ . Let $N(S_s)$ be the set of normalised nuggets associated with S_s and W_N then the **set** of terms of all normalised nuggets. $W_{N(S_s)}$ is the **set** of words in $N(S_s)$. Coverage is then defined as follows:

$$coverage(S_s) = \sum_{\forall w_i \in W_{N(S_s)} - W_{\Theta_\lambda}} P_i$$

where P_i is defined as the probability of finding a word $w_i \in W_N$, and is arbitrarily set to zero for all stop words. W_{Θ_λ} is the **set** of words occurring in preceding selected sentences Θ_λ .

Coverage aims at measuring how likely are novel terms (not seen in Θ_λ) within $N(S_s)$ to belong to a description. Thus, diverse sentences are preferred over

sentences with many redundant words, which are consequently filtered according to an experimental threshold ψ^* . On the other hand, content discriminates the degree, in which $N(S_s)$ conveys definition aspects of δ based upon highly close semantic terms and entities, and is given by:

$$\text{content}(S_s) = \sum_{\forall w_i \in \bar{W}} \Phi_{i_s} \hat{R}(\delta, w_i) + \sum_{\forall e \in N(S_s) - E_\lambda} P_e^\lambda$$

The first sum measures the semantic bonding of terms in the respective nuggets, and the second sum the relevance of novel entities (E_λ is the set of entities in Θ_λ). Each novel entity e is weighed according to its probability P_e^λ of being in the normalised nuggets of C_λ . Incidentally, $W_{\text{type}}(S_s)$ is the amount of undesirable symbols in S_s such as pronouns, unclosed brackets or parenthesis, URLs. Consequently, condition 5 bans sentences containing such symbols from Θ_λ . In sum, **Mdef-WQA** ranks sentences according to the order they are inserted into Θ_λ . This means that higher ranked sentences are more diverse, less redundant, and are likely to contain entities along with terms that describe aspects of δ .

Note further that C_0 is processed last in order to initialise Θ_λ with all sentences selected from previous clusters, so that only sentences with novel pieces of information remain in C_0 .

3 Experiments and Results

Mdef-WQA was assessed by means of standard question sets.³ The following data sets were considered for English: (1) TREC 2001, (2) TREC 2003, (3) CLEF 2004, (4) CLEF 2005, and (5) CLEF 2006. For Spanish only (4) and (5) were taken into account. All surface patterns thresholds were set to 0.25, apart from thresholds for patterns π_1^{en} , π_5^{en} , π_1^{es} and π_4^{es} , which were set to 0.33, 0.5, 0.33 and 0.4 respectively. These values were determined after experimentally testing different thresholds from 0.2 to 0.7, and thus manually counting the corresponding number of non-descriptive or spurious selected sentences. The threshold that controls redundancy ψ^* was set to 0.01 for both languages.

Three baselines were designed, one for English (**Baseline EN-I**) and two for Spanish (**Baseline ES-I** and **Baseline ES-II**). Like **Mdef-WQA**, **Baseline EN-I** retrieves 300 hundred snippets by submitting “ δ ” to the web. The retrieved snippets are split into sentences by means of JavaRAP, interpreting intentional breaks as end of sentences. **Baseline EN-I** also accounts solely for a stricter matching of δ by setting all pattern Π^{en} thresholds to one. A random sentence from a pair that shares more than 60% of their terms is discarded, cf. 5, as well as sentences that are a substring of another sentence. **Baseline ES-I** and **Baseline ES-II** do the same processing as **Baseline EN-I**, but they retrieve 420 snippets. These two baselines also differ from **Baseline EN-I** in the number of terms that two sentences must share to be considered as redundant. They

³ Along this section, \pm stands for standard deviation, and CLEF data-sets consider all English translations from all languages.

Table 4. Length of output sentences

| | | with white spaces | without white spaces |
|----------|-------|-------------------|----------------------|
| Baseline | ES-I | 98.11 ± 44.90 | 81.06 ± 37.69 |
| Baseline | ES-II | 104.98 ± 36.43 | 85.88 ± 29.87 |
| Mdef-WQA | ES | 135.78 ± 45.21 | 113.70 ± 37.97 |
| Baseline | EN-I | 118.168 ± 50.20 | 97.81 ± 41.80 |
| Mdef-WQA | EN | 125.70 ± 44.21 | 109.74 ± 42.15 |

account for a threshold of 90% instead of 60%, because the coverage of web space for Spanish is smaller than English and some relevant nuggets are missed along with the redundant content. The difference between the Spanish baselines is that **Baseline ES-I** aims at Π^{es} whereas **Baseline ES-I** at the patterns in [11].

In general, **Mdef-WQA** outputs short sentences, in particular, output sentences for English are comparative longer than the 100 characters (without considering white spaces) nuggets of [5] and smaller than the 250 characters (considering white spaces) fixed windows of [10]. Given the lengths of the outputs of **Baseline EN/ES-I** and **Mdef-WQA EN/ES** (see table 4), it can be concluded that the increase indicates that **Mdef-WQA outputs more complete sentences**, lessening the effects of intentional breaks on web snippets. Due to the acceptable length of descriptive sentences and the fact that many nuggets seems odd without their context [5], **Mdef-WQA** outputs sentences instead of only nuggets.

The degree of redundancy of a sentence S_s was roughly approximated at the word level by looking for a sentence $S_{s'}$ in the same response that shares the maximum number of terms with S_s :

$$redundancy(S_s) = \max_{S_{s'} \neq S_s} \frac{ns(S_s \cap S_{s'})}{ns(S_s)}$$

where $ns(S_s)$ is the number of words in S_s excluding stop-words. As a result, **Baseline ES-II** generates an output, at least, two times redundant as **Mdef-WQA**, which supplies longer sentences (see table 5). By and large, **Mdef-WQA outputs comparative longer and less redundant sentences**.

The coverage of surface patterns for English has been studied widely [5,6,7], by the same token table 6 shows the number of descriptive sentences in the final output that match each pattern in Π^{es} . Each cell represents the number

Table 5. Redundancy overview

| | (1) | (2) | (3) | (4) | (5) | |
|----------|-------|-------------|-------------|-------------|-------------|-------------|
| Baseline | ES-I | | | 0.32 ± 0.16 | 0.38 ± 0.25 | |
| Baseline | ES-II | | | 0.54 ± 0.24 | 0.64 ± 0.39 | |
| Mdef-WQA | ES | | | 0.25 ± 0.17 | 0.25 ± 0.16 | |
| Baseline | EN-I | 0.58 ± 0.26 | 0.61 ± 0.26 | 0.57 ± 0.25 | 0.62 ± 0.25 | 0.53 ± 0.23 |
| Mdef-WQA | EN | 0.47 ± 0.18 | 0.50 ± 0.20 | 0.45 ± 0.18 | 0.45 ± 0.17 | 0.45 ± 0.19 |

Table 6. Coverage of patterns

| | | π_1^{es} | π_2^{es} | π_3^{es} | π_4^{es} | π_5^{es} |
|----------|------|--------------|--------------|--------------|--------------|--------------|
| Baseline | ES-I | 78/37 | 17/10 | 00/00 | 13/10 | 05/03 |
| Mdef-WQA | | 470/254 | 168/95 | 03/01 | 59/58 | 54/36 |

Table 7. Results overview. (TQ = Total number of questions in the question-set).

| Corpus | Baseline EN-I | | | Mdef-WQA | | | | |
|--------|---------------|-----|---------------|-------------|------------|--------------|-------------|---------|
| | TQ | AQ | NS | Accuracy | AQ | NS | Accuracy | AS (%) |
| (1) | 133 | 81 | 7.35 ± 6.89 | 0.87 ± 0.2 | 133 | 18.98 ± 5.17 | 0.94 ± 0.07 | 16 ± 20 |
| (2) | 50 | 38 | 7.7 ± 7.0 | 0.74 ± 0.2 | 50 | 14.14 ± 5.3 | 0.78 ± 0.16 | 5 ± 9 |
| (3) | 86 | 67 | 5.47 ± 4.24 | 0.83 ± 0.19 | 78 | 13.91 ± 6.25 | 0.85 ± 0.14 | 5 ± 9 |
| (4) | 185 | 160 | 11.08 ± 13.28 | 0.84 ± 0.2 | 173 | 13.86 ± 7.24 | 0.89 ± 0.15 | 4 ± 11 |
| (5) | 152 | 102 | 5.43 ± 5.85 | 0.85 ± 0.22 | 136 | 13.13 ± 6.56 | 0.86 ± 0.16 | 8 ± 14 |

of matches for the CLEF 2005/2006 corpus respectively. π_1^{es} provides the wider coverage, while π_3^{es} the most limited. Given the marked increase in the number of recognised descriptive utterances in the final output, it can be concluded that our query rewriting strategy strongly biases the search engines not only in favour of **redundant** descriptive sentences, but also in favour of **diverse** utterances. On the one hand, redundant sentences are undesirable in the final output, on the other hand, they are useful for distinguishing more relevant and reliable descriptive utterances.

We considered an entirely different evaluation for each language for the following reasons: (a) the way the performance of definition QASs is measured differs between TREC and CLEF, and (b) CLEF gold standards for definition questions supply only one nugget regarding abbreviations or position of persons, whereas TREC 2003 provides a set of relevant nuggets.

To start with the discussion of the obtained results, table 7 shows the coverage of **Baseline EN-I** and **Mdef-WQA**. AQ stands for the number of questions, for which its response contained at least one nugget (manually checked). **Mdef-WQA** discovered nuggets for all questions in (2), contrary to [1], who found nuggets for solely 42 questions by using external dictionaries and web snippets. In addition, **Mdef-WQA** discovered nuggets within **snippets** for the 133 questions in (1), in contrast to [10], who found a top five ranked snippet that conveys a definition solely for 116 questions within top 50 **downloaded full documents**.

Overall, **Mdef-WQA** covered 94% of the questions, whereas **Baseline EN-I** 74%. This difference is mainly due to the query rewriting step and the more flexible matching of δ . For all questions, in which **Mdef-WQA** and **Baseline EN-I** discovered at least one nugget, the accuracy and the average number of sentences (NS), containing also at least one nugget, was computed. **Mdef-WQA** doubles the number of sentences and achieves a slightly better accuracy. In table 7, AS corresponds to the percentage of sentences within NS, for which the relaxed matching shifted δ to another concept. Some shifts caused interesting descriptive phrases. A good example is: “*neuropathy*” was shifted to “*peripheral neuropathy*” and

Table 8. TREC 2003 results

| | Recall | Precision | Av. len. |
|-----------------|-------------|-------------|----------|
| Baseline | 0.35 ± 0.34 | 0.30 ± 0.26 | 583 |
| Mdef-WQA | 0.61 ± 0.33 | 0.18 ± 0.13 | 1878 |

Table 9. TREC 2003 $F(\beta)$ scores

| β | 1 | 2 | 3 | 4 | 5 |
|----------------------|------|------|------|------|------|
| Mdef-WQA | 0.26 | 0.37 | 0.45 | 0.50 | 0.53 |
| Baseline EN-I | 0.26 | 0.30 | 0.32 | 0.32 | 0.34 |

“*auditory neuropathy*”, conversely, some shifts caused loosely related sentences: “*G7*” to “*Powershot G7*”.

In order to compare our methods with a gold standard for English, we used the assessors’ list provided through the TREC 2003 data. Following the approach of TREC, table 8 displays our current achievement. Given the higher recall 0.61 ± 0.33 obtained by **Mdef-WQA**, it can be concluded that the additional sentences that it selects contain more nuggets seen as vital on the assessor’s list. A key point for the interpretation of the precision is the completeness of the assessor’s list. It is known that systems in TREC are able find valid nuggets, which are judged as not relevant in the list (cf. 5 for details). This is even more likely for web-based system like **Mdef-WQA**, because they will discover many additional nuggets charged as relevant by a user, but will not hit the list. This kind of “it-is-not-on-my-list-evaluation” actually brings about a decrease, because they enlarge the response without increasing precision. In **Mdef-WQA**, this is a critical aspect, because it increases almost twofold the amount of selected descriptive sentences per question (see table 7), and hence, the length of the response.

Given the $F(\beta)$ score achieved for each response by **Mdef-WQA** (see table 9 14), it can be concluded: (a) it is “competitive” with the best systems in TREC 2003, which achieved between 0.5 and 0.56 for $\beta=5$, and (b) additional sentences provided novel nuggets. It is also worth to remark that **Baseline EN-I** obtained a slightly better $F(\beta=5)$ for the following δ s: “*Akbar the Great*”, “*Albert Ghiorso*” and “*Niels Bohr*”. This simply means that these responses were closer to the the assessors’ expectations.

For Spanish, **Mdef-WQA** answered 32 and 22 out of the CLEF 2005 and 2006 questions respectively (see table 10). However, the runs submitted by the best two systems in CLEF 2005 answered 40 out of the 50 definition questions 13 11. Nevertheless, the third best system only answered 26 questions. Additionally,

Table 10. Gold standards

| | Baseline ES-I | Baseline ES-II | Mdef-WQA |
|-----|----------------------|-----------------------|-----------------|
| (4) | 11 | 33 | 32 |
| (5) | 9 | 12 | 22 |

Table 11. Results overview. (TQ = Total number of questions in the question-set).

| Corpus | Baseline ES-I | | | Baseline ES-II | | | |
|--------|---------------|-----------|-----------------|-----------------|----|-------------------|-----------------|
| | TQAQ | NS | Accuracy | AQ | NS | Accuracy | |
| (4) | 50 | 26 | 2.59 ± 2.45 | 0.85 ± 0.23 | 39 | 10.13 ± 10.66 | 0.67 ± 0.31 |
| (5) | 42 | 10 | 3.00 ± 3.13 | 0.61 ± 0.31 | 15 | 3.4 ± 3.31 | 0.65 ± 0.26 |
| Corpus | Mdef-WQA | | | | | | |
| | TQ | AQ | NS | Accuracy | | | |
| (4) | 50 | 47 | 8.6 ± 4.85 | 0.63 ± 0.19 | | | |
| (5) | 42 | 30 | 7.27 ± 6.76 | 0.67 ± 0.25 | | | |

the best system in CLEF 2006 answered 35 out of the 42 definition questions, whereby **Mdef-WQA** found answers for 22 out of the 35 questions answered by this best system. Unfortunately, CLEF 2006 gold standard provides only one nugget for only these 35 questions.

Since the coverage of the gold standards focuses solely on abbreviations and positions of persons, and answers for seven CLEF 2006 questions are missed, we assigned three out of five different assessors to each data-set. Each assessor judged whether or not each output sentence yielded descriptive information. A sentence was considered as descriptive if and only if at least two out of the three assessors agreed (results in table [11](#)). In both data-sets, **Mdef-WQA** outperformed both baselines, in particular, it discovered descriptive phrases for 47 out of the 50 CLEF 2005 questions. Additionally, **Mdef-WQA** returned more descriptive utterances (NS) with a lower level of redundancy. However, the accuracy of the output sentences decreased compared to our English results. We interpret this as a consequence of the lower amount of web redundancy for Spanish, which effects the quality of identifying the most relevant and reliable phrases. Finally, table [10](#) shows that the performance of **Mdef-WQA** can be improved by aligning patterns in [11](#) without necessarily considering them in the rewriting process.

All in all, the substantial difference in the performance between **Baseline EN/ES-I** and **Mdef-WQA** stresses the improvement caused by the query rewriting, and proves that extracting answers to definition questions straightforwardly from web snippets is promising.

Concerning the performance of the sense disambiguation process, **Mdef-WQA** was able to distinguish different potential senses for some δ s, e.g., for “*atom*”, the particle-sense and the format-sense. On the other hand, some senses were split into two separate senses, e.g., “*Akbar the Great*”, where “*emperor*” and “*empire*” indicated different senses. This misinterpretation is due to the independent co-occurrence of “*emperor*” and “*empire*” with δ , and the fact that they are unlikely to share words. In order to improve this, some external sources of knowledge are necessary. This is not a trivial problem, because some δ s can be extremely ambiguous like “*Jim Clark*”, which refers to more than ten different real-world entities. **Mdef-WQA** recognised the pilot and the Netscape founder (Fig. [11](#)). Independently of that, we found that entities and the correlation of

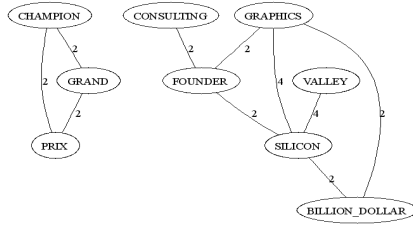


Fig. 1. $\hat{\Phi}_{ij} > 1$ for $\delta = \text{“Jim Clark”}$

highly closed terms in the semantic space provided by LSA can be important building blocks for a more sophisticated strategy for the disambiguation of δ .

4 Conclusions and Future Work

This work presents *Mdef-WQA*, a system that extracts answers for definition questions from web snippets. Our ongoing research focuses on adapting our system to deal with German. This adaptation brings about two challenges: (a) discriminate descriptive phrases in present tense from sentences in perfect tense with “*sein*”, and (b) cope with the orthographical variations caused by umlauts and compounds.

Mdef-WQA pioneers attempts by definitional QAS to disambiguate descriptive utterances. One finding is that web snippets do not provide the necessary information for a complete disambiguation. To overcome this problem, external resources such as full documents, WordNet and/or additional queries might be explored as a source for fetching extra information from the web.

An additional challenge is recognising of relevant morpho-syntactical variations of descriptive sentences, which would help to decrease the redundancy of the output. Anyway, this redundancy can still be useful for discovering answers to definition questions in the context of the TREC/CLEF Question Answering tracks, projecting these redundant utterances to the corresponding corpus.

References

1. Cui, T.S.C.H., Kan, M.Y., Xiao, J.: A comparative study on sentence retrieval for definitional question answering. In: SIGIR Workshop on Information Retrieval for Question Answering (IR4QA), July 29, 2004, Sheffield, UK (2004)
2. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing By Latent Semantic Analysis. *Journal of the American Society For Information Science* 41, 391–407 (1990)
3. Denicia-Carral, C., Montes-y-Gómez, M., Villaseñor-Pineda, L., García Hernández, R.: A Text Mining Approach for Definition Question Answering. In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) *FinTAL 2006*. LNCS (LNAI), vol. 4139, pp. 76–86. Springer, Heidelberg (2006)

4. Goldstein, J., Mittal, V., Carbonell, J., Kantrowitz, M.: Multi-document summarization by sentence extraction. In: NAACL-ANLP 2000 Workshop on Automatic summarization, pp. 40–48 (2000)
5. Hildebrandt, W., Katz, B., Lin, J.: Answering Definition Questions Using Multiple Knowledge Sources. In: HLT-NAACL 2004, pp. 49–56 (2004)
6. Joho, H., Sanderson, M.: Large Scale Testing of a Descriptive Phrase Finder. In: 1st Human Language Technology Conference, San Diego, CA, pp. 219–221 (2001)
7. Joho, H., Sanderson, M.: Retrieving Descriptive Phrases from Large Amounts of Free Text. In: 9th ACM conference on Information and Knowledge Management, McLean, VA, pp. 180–186. ACM Press, New York (2000)
8. Kintsch, W.: Predication. *Cognitive Science* 25, 173–202 (1998)
9. Magnini, B., Giampiccolo, D., Forner, P., Ayache, C., Osenova, P., Peas, A., Jijkoun, V., Sacaleanu, B., Rocha, P., Sutcliffe, R.: Overview of the CLEF 2006 Multilingual Question Answering Track. In: Working Notes for the CLEF 2006 Workshop, 20–22 September, 2006, Alicante, Spain (2006)
10. Miliaraki, S., Androutsopoulos, I.: Learning to Identify Single-Snippet Answers to Definition Questions. In: COLING 2004, pp. 1360–1366 (2004)
11. Montes-y-Gómez, M., Villaseñor-Pineda, L., Pérez-Coutiño, M., Gómez-Soriano, J.M., Sanchis-Arnal, E., Rosso, P.: INAOE-UPV Joint Participation in CLEF 2005: Experiments in Monolingual Question Answering. In: Peters, C., Gey, F.C., Gonzalo, J., Müller, H., Jones, G.J.F., Kluck, M., Magnini, B., de Rijke, M., Giampiccolo, D. (eds.) CLEF 2005. LNCS, vol. 4022, pp. 21–23. Springer, Heidelberg (2006)
12. Soubbotin, M.M.: Patterns of Potential Answer Expressions as Clues to the Right Answers. In: Proceedings of the TREC-10 Conference, NIST (2001), Gaithersburg, Maryland (2001)
13. Vallin, A., Giampiccolo, D., Aunimo, L., Ayache, C., Osenova, P., Peñas, A., Rijke, M., Sacaleanu, B., Santos, D., Sutcliffe, R.: Overview of the CLEF 2005 Multilingual Question Answering Track. In: Peters, C., Gey, F.C., Gonzalo, J., Müller, H., Jones, G.J.F., Kluck, M., Magnini, B., de Rijke, M., Giampiccolo, D. (eds.) CLEF 2005. LNCS, vol. 4022, pp. 21–23. Springer, Heidelberg (2006)
14. Voorhees, E.M.: Evaluating Answers to Definition Questions. In: HLT-NAACL 2003, 109–111 (2003)
15. Wiemer-Hastings, P., Zipitria, I.: Rules for Syntax, Vectors for Semantics. In: Proceedings of the 23rd Annual Conference of the Cognitive Science Society (2001)

A SPARQL Semantics Based on Datalog^{*}

Simon Schenk

Institute for Computer Science,
University of Koblenz,
sschenk@uni-koblenz.de

Abstract. SPARQL is the upcoming W3C standard query language for RDF data in the semantic web. In this paper we propose a formal semantics for SPARQL based on datalog. A mapping of SPARQL to datalog allows to easily reuse existing results from logics for analysis and extensions of SPARQL. Using this semantics we analyse the complexity of query answering in SPARQL and propose two useful extensions to SPARQL, namely binding of variables to results of filter expressions and views on RDF graphs as datasets for queries. We show that these extensions do not add to the overall complexity of SPARQL.

1 Introduction

The Resource Description Framework (RDF) is a language for representing semantic information in the World Wide Web [9], based on a graph model. Concepts and instances are nodes of this graph and their relations are represented as the edges.

SPARQL is the upcoming W3C standard query language for RDF data on the semantic web. SPARQL is based on graph pattern matching. A SPARQL query matches a graph pattern against a dataset consisting of one or more input graphs. The resulting variable bindings are either returned in tabular form ("select queries") or embedded into a template description in order to generate new RDF data ("construct queries").

In this paper we propose a formal semantics for SPARQL based on a mapping to datalog. We prove interesting results about SPARQL using the existing body of knowledge about datalog. Based on the provided mapping we propose two useful extensions to SPARQL and show that they do not add to the complexity of SPARQL query answering.

Unlike for example the SQL query language for relational databases, SPARQL currently does not allow to bind variables to values of functions applied on other bindings. The first extension deals with binding of variables to the results of SPARQL filter expressions. As an example we will use a graph containing information about the price of a product in Euros and the conversion rate from Euro to Dollar. Using the extension, it will be possible to directly ask for the Dollar price, which is not possible in the current SPARQL working draft [12].

^{*} This research was supported by the European Commission under contract FP6-027026, Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content - K-Space. The expressed content is the view of the authors but not necessarily the view of the K-Space project.

The second extension aims at using the result of SPARQL construct queries as part of the dataset of another query, a feature desirable for the formulation of complex queries in a concise way. We will show how to use a virtual graph containing computed dollar prices as dataset for a second query. This second query will search for products of a certain maximum dollar price without first having to compute it.

2 Foundations

We start by giving some foundations we will build our formalisation on. SPARQL is a query language for the RDF data model plus an extension called named graphs. Therefore we will introduce RDF and named graphs before giving a short overview of SPARQL. In some aspects, where the SPARQL specification working draft slightly differs from the RDF specification, we will follow the SPARQL specification.

2.1 RDF

RDF is a graph based knowledge representation language. The nodes in a graph are IRIRRefs¹, blank nodes² or literals. Arcs between the nodes represent their relationships. The arcs are labeled with IRIRRefs, representing the property that holds between the two nodes.

Definition 1. *RDF statement, RDF graph.*

Let I be the set of IRIRRefs, L the set of RDF Literals and B the set of Blank Nodes as defined in [12]. I , L and B are pairwise disjoint. Let $R = I \cup L \cup B$. A statement is a triple in $R \times I \times R$. If $S = (s, p, o)$ is a statement, s is called the subject, p the predicate and o the object of S .

An RDF graph is a set of statements. For every two graphs G_1 and G_2 the sets of blank nodes in statements in G_1 and in G_2 are disjoint.

[9] defines a model theoretic semantics for RDF graphs based on inference rules. They include, for example, transitivity of class membership. We refer the reader to [9] for a detailed description of RDF, which includes a definition of RDF and the RDF Schema language RDFS. We do not discuss RDF semantics in this paper. [14] proposes a mapping of RDFS to logic programming, which could be combined with our approach. Note that in contrast to [9], but analogous to the SPARQL specification working draft defined in [12] we allow literals as subjects of statements.

2.2 Named Graphs

While the RDF recommendation does not allow referring to RDF graphs, *named graphs* introduced in [3] offer means to group a set of statements together into a graph and to refer to this graph using an IRIRRef. SPARQL uses named graphs to declare the dataset a query is evaluated on.

¹ The most common type of IRIRRefs are URLs, e.g. `http://isweb.uni-koblenz.de`

² A kind of existentially quantified variables.

Definition 2. *Named graph.*

A named graph is a pair (n, G) of an IRIRef n and an RDF graph G .

Please note that n is only used to refer to G and can not necessarily be dereferenced to G . One purpose of naming G is the use of n to describe G , for example when tracking provenance information. The following example shows a named graph named `http://ex.org/g1` in N3 syntax [2], which contains four statements. One of these statements is used to describe the graph itself.

Example 1.

```
http://ex.org/g1 {
  http://ex.org/g1 http://ex.org/usedAs "example"^^xsd:string.
  http://ex.org/aproduct rdf:type http://ex.org/product.
  http://ex.org/aproduct http://ex.org/euroPrice "8.15"^^xsd:float.
  http://ex.org/euro http://ex.org/dollarExchRate "1.3319"^^xsd:float.}
```

3 Introduction to SPARQL

A SPARQL query [12] consists of three parts, namely a dataset, a graph pattern and a projection or construct pattern.

The dataset defines the scope of query evaluation: A list of named graphs to be used can be defined. These are matched, whenever a part of the graph pattern is scoped to named graphs. Additionally, a default graph is defined, which is used iff no scoping to named graphs is done. The default graph can be the union of multiple named graphs. For example if the dataset and graph patterns shown in example 2 are used, pattern 1 would be matched against the union of `http://ex.org/g1` and `http://ex.org/g2`. Pattern 2 would be matched against `http://ex.org/g3` and `http://ex.org/g4`, binding `?g` to the name of the named graph containing the matched statement. Pattern 3 would be matched against `http://ex.org/g3` only and pattern 4 would match nothing, because `http://ex.org/g1` is not one of the named graphs declared in the dataset using FROM NAMED. Note that variable names are written with a leading questionmark in SPARQL.

Example 2.

```
# dataset
FROM <http://ex.org/g1>          FROM <http://ex.org/g2>
FROM NAMED <http://ex.org/g3>  FROM NAMED <http://ex.org/g4>

# graph pattern 1:  {?s ?p ?o.}
# graph pattern 2:  GRAPH ?g {?s ?p ?o}}
# graph pattern 3:  GRAPH <http://ex.org/g3> {?s ?p ?o}
# graph pattern 4:  GRAPH <http://ex.org/g1> {?s ?p ?o}
```

The graph pattern of a query is matched against the data in the dataset. Complex graph patterns are composed of simpler ones using conjunction, a kind of disjunctions, a kind of left outer joins and filter expressions, which further constrain variable bindings. We will describe particularities of these connectives later in this chapter.

In the case of a select query, the variable bindings produced by matching the graph pattern are projected to a tabular representation. For example the select projection `?g ?p <http://ex.org/some/iri>` would return a table containing three columns with the bindings of `?g`, `?p` and a constant value. In the result of a select query, null bindings for a subset of the variables used are allowed.

In the case of construct queries, the variable bindings are used to construct new RDF statements. The construct pattern in this case again is a kind of graph pattern. However, it is not matched against existing data but instantiated with the computed variable bindings. All valid RDF statements created during this instantiation are returned as result of the query. In this context, 'valid' means subjects and objects of all statements are in R , predicates in I and all variables are bound.

For a detailed specification of SPARQL the reader is referred to [12].

4 SPARQL Syntax

In this paper we will use a syntactical subset of SPARQL, which however semantically can express all valid SPARQL SELECT and CONSTRUCT queries. We first introduce the syntax used here, which is very close to that of [12]. Then we define a normalisation of graph pattern scoping, which will allow to define our semantics more concisely.

Definition 3. SPARQL query

A SPARQL construct query is a triple (C, D, P) of a construct pattern C , a dataset D and a graph pattern P . A SPARQL select query is a triple (S, D, P) of a select projection S , a dataset D and a graph pattern P .

The dataset definition specifies which graphs are evaluated by the query. A default graph is specified, which is the union of the graphs enumerated using *FROM*. The set of named graph used for query evaluation is listed using *FROM NAMED*.

Definition 4. Dataset.

Let n be an IRIRef which is a name of a named graph. Then *FROM n* is a dataset and *FROM NAMED n* is a dataset. If D and D' are datasets, then $D + D'$ is a dataset.

Definition 5. Construct pattern.

Let V be a set of variable names and R and I defined as above. If $s \in R \cup V$, $p \in I \cup V$ and $o \in R \cup V$, then $\{s, p, o.\}$ is a construct statement pattern. A construct pattern is a list of construct statement patterns. A construct pattern is written as the keyword *CONSTRUCT* followed by a conjunction of construct statement patterns in curly brackets.

Example 3. *CONSTRUCT* $\{\{?s ?p ?o.\}.\{?x ?y ?z.\}\}$ is a *CONSTRUCT* pattern.

Definition 6. Select projection.

Let V and R be defined as above. A select projection is a tuple of elements from $R \cup V$. A select projection is written as the keyword *SELECT* followed by a list of elements from $R \cup V$

Example 4. *SELECT* `?product ?price` is a *SELECT* projection.

Definition 7. *Graph pattern.*

Let V , R and I be defined as above. If $s \in R \cup V$, $p \in I \cup V$ and $o \in R \cup V$, then $\{s, p, o.\}$ is a graph pattern, called statement pattern. If P_1 and P_2 are graph patterns, $\{P_1 . P_2\}$, $\{P_1 \text{ OPTIONAL } P_2\}$ and $\{P_1 \text{ UNION } P_2\}$ are graph patterns. If P is a graph pattern and F is a filter expression as defined below, $\{P \text{ FILTER } F\}$ is a graph pattern. If P is a graph pattern and $n \in I \cup V$ then $\{\text{GRAPH } n P\}$ is a graph pattern.

'.' stands for conjunction of graph patterns. UNION and OPTIONAL are similar to disjunction and left outer join in the relational calculus respectively. They are special, however, in that they may leave variables unbound (UNION in both graph patterns used as parameters, OPTIONAL only on the optional side), resulting in *null* bindings.

Filter expressions are recursively defined as follows:

Definition 8. *Filter expression.*

Let f be an *IRIRef* or one of the builtin operators listed in [L2], section 11. Let $v_1, \dots, v_j \in I \cup R \cup V$ and $j > 0$. Then $f(v_1, \dots, v_j)$ is a filter expression. If F_1, F_2 are filter expressions, $(F_1 \ \&\& \ F_2)$, $(F_1 \ || \ F_2)$ and $(!F_1)$ are filter expressions. In SPARQL queries filter expressions are prepended the keyword *FILTER*.

A select query, asking for all products with a euroPrice of less than 9 could look as follows:

Example 5.

```
SELECT ?product ?price
FROM <http://ex.org/g1>
WHERE {{{?product rdf:type <http://ex.org/product>}.} .
      {?product <http://ex.org/euroPrice> ?price}}
      FILTER (?price < "9.0"^^xsd:float)}
```

4.1 Normalisation of GRAPH Patterns

We apply some preprocessing to transform a generic graph pattern into a more suitable form for our translation. The GRAPH expression in SPARQL "overrides" the current default graph for evaluating a pattern. Hence, in the following example: $\{\text{GRAPH } g_1 \{\{P_1 . \{\text{GRAPH } g_2 P_2 \}\} . P_3\}\}$, P_1 and P_3 are evaluated against g_1 and P_2 is evaluated against g_2 . We can resolve this nesting into a flat structure as follows by simple rewriting of the query string using string pattern substitution. First, we explicitly scope every unscoped statement pattern with the default graph. Rule N1 removes unnecessary GRAPH expressions. Rules N2 to N5 distribute scoping over graph patterns down to the level of statements patterns for each of the operators '.', UNION, OPTIONAL and FILTER.

Definition 9. *Normalisation of graph scoping.*

Let P be the graph pattern of a SPARQL query. Let $d \in I$ be a newly introduced name for the default graph of the query. Let g_1, g_2 be names of named graphs. Let P_1, P_2, P_3 be graph patterns. Let F be a filter expression. Apply the following rules left to right. Apply N0) only once. Afterwards, apply N1) to N5) as long as possible:

| | substitute | by |
|-----|--|--|
| N0) | P | {GRAPH d P} |
| N1) | {GRAPH g_1 {GRAPH $g_2 P_1$ }} | {GRAPH $g_2 P_1$ } |
| N2) | {GRAPH g_1 { $P_1 . P_2$ }} | {{GRAPH $g_1 P_1$ } . {GRAPH $g_1 P_1$ }} |
| N3) | {GRAPH g_1 { P_1 UNION P_2 }} | {{GRAPH $g_1 P_1$ } UNION {GRAPH $g_1 P_1$ }} |
| N4) | {GRAPH g_1 { P_1 OPTIONAL P_2 }} | {{GRAPH $g_1 P_1$ } OPTIONAL {GRAPH $g_1 P_1$ }} |
| N5) | {GRAPH g_1 { P_1 FILTER F}} | {GRAPH $g_1 P_1$ } FILTER F |

5 Semantics

In this section we define a mapping of SPARQL to datalog. The semantics of a SPARQL query is then defined using a query to corresponding datalog program. Our mapping function is called m and defined in the following. We will use functional and relational syntax interchangeably where useful, i.e. for $f(x) = y$ we will write $(x, y) \in f$ and analogous for the inverse: $f^{-1}(y) = x$ and $(y, x) \in f^{-1}$.

We use the following syntax for datalog:

- A datalog program is a set of normal clauses.
- A normal clause has the form $H \leftarrow B$. H is called the head and B the body of the clause. H is an atom and B is a conjunction of literals.
- “,” in the body of a clause stands for conjunction. “ \neg ” and “ \vee ” are used for negation and disjunction.

Please note, that the following mapping rules do not directly generate a datalog program, but are translated to datalog using Lloyd-Topor transformation [7] afterwards. For details on datalog, we refer the reader for example to [4]. We will make use of skolem functions for the translation defined in the following. These skolem functions are used for the translation from SPARQL to datalog programs, not within the programs themselves. Hence, they do not increase the complexity of evaluation of the resulting programs, which are function free. Additionally it can easily be shown, that the skolem functions used can be implemented with linear complexity, so they also do not add to the complexity of query translation.

Let C_I, C_B, C_L be disjoint sets of constants of a datalog program DP . Let C_V be a set of variables of DP . We define the following bijective mappings:

$$m_R : I \rightarrow C_I \subset m,$$

$$m_B : B \rightarrow C_B \subset m,$$

$$m_L : L \rightarrow C_L \subset m, \text{ such that } m_L(xsd:true) = true, m_L(xsd:false) = false$$

$$m_V : V \rightarrow C_V \subset m, m(\text{null}) = \text{null}$$

First we map the dataset of the query. Rule 02) maps all statements in a named graph to facts. We use the predicate $t/4$ to hold the true statements in the dataset and the query results. If $t(g, s, p, o)$ holds, then graph g contains a statement (s, p, o) . Rule 01) composes the default graph from these facts. We define a mapping for the graph pattern connectives (rules 03) to 06)) taking care of null bindings.

Let $deref$ be a function mapping from the name of a named graph (a IRIRef) to the actual graph. Let g be a name of a named graph. We assign the (new) name d to the default graph of the dataset of the query. Let $varsInt$ be a function mapping from

a SPARQL expression E to the set of variables it introduces, i.e. that are not used in statement patterns outside E .

| X | m(X) |
|-------------------------------|---|
| 01) (FROM g) | $t(d, x, y, z) \leftarrow t(m(g), x, y, z)$ |
| 02) (FROM NAMED g) | $\{t(m(g), m(s), m(p), m(o)) \leftarrow \mid (s, p, o) \in \text{deref}(g)\}$ |
| 03) (GRAPH g (s, p, o)) | $t(m(g), m(s), m(p), m(o))$ |
| 04) (P_1 . P_2) | $m(P_1), m(P_2)$ |
| 05) (P_1 OPTIONAL P_2) | $m(P_1), m(P_2) \vee$ $(\neg P_2, \text{isNull}(m(v_1)), \dots, \text{isNull}(m(v_n)))$ where $\{v_1, \dots, v_n\} = \text{varsInt}(P_2)$ |
| 06) (P_1 UNION P_2) | $(m(P_1), \text{isNull}(m(v_1)), \dots, \text{isNull}(m(v_n))) \vee$ $(m(P_2), \text{isNull}(m(w_1)), \dots, \text{isNull}(m(w_m)))$ where $\{v_1, \dots, v_n\} = \text{varsInt}(P_2)$ and $\{w_1, \dots, w_m\} = \text{varsInt}(P_1)$ |

In SPARQL, filter expressions are composed of *filter functions*, mapping a list of parameters to a literal, and three valued logical connectives $\&\&$, \parallel and $!$ for conjunction, disjunction and negation, which work on truth values in $\{\text{true}, \text{false}, \text{error}\}$. Parameters of filter functions again can be filter functions. The return value of a filter function can be any literal or *error*. Hence, when evaluating logical connectives, the literal is mapped to its *effective boolean value*. The effective boolean value is defined analogous to programming languages like C: If a literal is of boolean type, its boolean value is obvious. If it is of numeric type it is false if the value is 0 and true otherwise. If it is of a string type or untyped, it is true if the value is of length larger than zero and false otherwise.

We evaluate filter expressions using predicates defined for every filter function which is listed in [12], chapter 11 (there called filter operators). The results of evaluating a filter function or effective boolean value F are always bound to a skolem variable $\text{var}(F)$, which is the last parameter of the predicate. This variable is used in surrounding filter expressions to access the results of the nested filter expression. As filter expressions can be used as parameters of other filter expressions, mapping rules 07) and 08) handle the cases that a parameter of a filter function is in $V \cup R$ or that it is a filter expression. To avoid modeling all of the filter functions of SPARQL, we assume for now that these predicates are external ones. Note, that of cause this is not necessary.

We provide a theory for computing the effective boolean value and for the three valued boolean connectives of SPARQL using the predicates *and*, *or* and *not*. Here, we only give a mapping from SPARQL filter expressions to datalog. The theory for the predicates *and*, *or*, *not*, *ebv*, *isTrue* and *bound* can be found in the appendix.

Let O be a filter operator, $v_1 \dots v_i \in R \cup V$ and F, F' be filter expressions.

| X | m(X) |
|--------------------------|---|
| 07) (P FILTER F) | $m(P), m(F), \text{ebv}(\text{var}(F), \text{var}(p \text{ FILTER } F)),$ $\text{isTrue}(\text{var}(p \text{ FILTER } F))$ |
| 08) $O(\dots, F, \dots)$ | $m(F), m(O)(\dots, \text{var}(F), \dots)$ |
| 09) $O(v_1, \dots, v_i)$ | $m(O)(m(v_1), \dots, m(v_i), \text{var}(O))$ |

| X | m(X) |
|-------------|--|
| 10) F && F' | m(F), m(F'), ebv(var(F), var(ebv(F))), ebv(var(F'), var(ebv(F'))), and(var(ebv(F)), var(ebv(F')), var(F&&F')) |
| 11) !F | m(F), ebv(var(F), var(!F)), not(var(!F)) |
| 12) F F' | m(F), m(F'), ebv(var(F), var(ebv(F))), ebv(var(F'), var(ebv(F'))), or(var(ebv(F)), var(ebv(F')), var(F F')) |

The filter function *bound* is a special case, which is used here also outside of filter expressions. It checks, whether a variable has been bound by graph pattern matching, i.e. whether it's value is different from *null*. It is the only filter operator accepting unbound variables as operands without resulting in error. Thus, the bound filter is used in combination with optional graph patterns to model negation as failure in SPARQL and to return a query result if a graph pattern could *not* be matched. We also use *bound* in rule 13) to guarantee valid statements as results of a construct query. The bound filter is the only part of the theory, where negation is used:

isNull(null) \leftarrow
bound(x, true) $\leftarrow \neg$ isNull(x)

Now we have everything necessary for translating a whole SPARQL query. A construct query is mapped to a set of program clauses in rule 13); one for every statement pattern in the construct pattern. The last 4 subgoals in 13) are to ensure, that valid RDF statements are produced. In the case of a SPARQL select query, we need a projection instead of a triple generation as the last step. Note that in the case of a select query, null bindings in the result are allowed. Select queries are translated using rules 14) and 15). The result of translating a query always is a set of clauses.

Let C be a construct pattern, r a new name introduced for the resulting graph of a construct query and S be a select pattern.

| X | m(X) |
|---------------|--|
| 13) (C, D, P) | m(D) \cup { c(m(r), m(s), m(p), m(o)) \leftarrow m(P), bound(s), bound(p), isIRI(p), bound(o) (s,p,o) \in C } |
| 14) (S, D, P) | m(D) \cup {m(S) \leftarrow m(P)} |
| 15) S | s(m(v ₁), ..., m(v _n)) where (v ₁ , ..., v _n) = S |

To the program DP resulting from mapping a query, we apply Lloyd-Topor transformation [7], in order to eliminate disjunction in the body of clauses in DP . Using the mapping developed in this section, we can now define the semantics of a SPARQL query based on the corresponding datalog program.

Definition 10. *Semantics of SPARQL queries.*

Let m^{-1} be the inverse of the mapping $m_C \cup m_B \cup m_L$.

If Q is a construct query, the result of Q is the set of statements ($m^{-1}(s)$, $m^{-1}(p)$, $m^{-1}(o)$) obtained from the bindings computed for the goal $\leftarrow c(r, s, p, o)$ from $m(Q)$.

If Q is a select query, the result of Q is the set of tuples ($m^{-1}(v_1)$, ..., $m^{-1}(v_n)$) obtained from the bindings computed for the goal $\leftarrow s(v_1, \dots, v_n)$ from $m(Q)$, where (v₁, ..., v_n) = S and S is the select projection of Q.

Using the mapping defined in this section, example 5 translates to the following program, assuming a literal mapping of IRIRefs and constants.

Example 6.

```

1: t(d, S, P, D) <- t(g1, s, p, o)           % rule 1)
2: t(g1, g1, usedAs, "example") <-         % rule 2)
3: t(g1, aproduct, rdfType, product) <-
4: t(g1, aproduct, euroPrice, 8.15) <-
5: t(g1, euro, dollarExchRate, 1.3319) <-
6: s(PRODUCT, PRICE) <-                     % rules 14) and 15)
    t(d, PRODUCT, rdfType, product),       % rules 3) 4)
    t(d, PRODUCT, euroPrice, PRICE),
    lt(PRICE, 9, F), isTrue(F)             %rules 7) and 8)

```

6 Properties of the Logic Based SPARQL Semantics

First we investigate the size of the logic program resulting from translating a dataset and a query.

6.1 Complexity of the Translation

While the translation introduced above is exponential in the length of the query, we will show that we can optimise it to be linear.

Lemma 1. *Complexity of construct query translation.*

The length of the logic program resulting from the translation of a SPARQL construct query (C, D, P) is $O(|T| + |D| + |C|2^l|P|)$, where T is the size of our background theory shown in appendix A |D| is the number of statements in the graphs in the dataset plus the number of graphs used for generating the default graph, $|C|$ is the number of statement patterns in the construct pattern, l is the maximum level of nestings of OPTIONAL and UNION patterns, and $|P|$ is the length of the graph pattern.

Proof. The mapping of the dataset adds exactly one fact for every statement in the graph declared using FROM or FROM NAMED and a clause for every graph used for generating the default graph. Every query is translated into $|C|$ clauses. These are split using Lloyd Topor transformation: For every logical OR in the body of a clause, Lloyd Topor transformation results in two new clauses. As logical ORs are only introduced by mappings of UNION and OPTIONAL patterns, this step generates at most $|2^l|$ clauses. Statement patterns and filter expressions map to a number of literals of the logic program which is equal to the number of statement patterns in the graph pattern or the length of the filter expression respectively. Unbound OPTIONAL patterns are replaced by a list of *isNull* atoms for the unbound variables, if any. This list has at most three times the length the OPTIONAL graph pattern. Thus the overall length of the resulting clauses is bound by $3|P|$.

Lemma 2. *Complexity of select query translation.*

The length of the logic program resulting from the translation of a SPARQL select query (S, D, P) is $O(|T| + |D| + |2^l||P|)$, where T is the size of our background theory shown in appendix A, $|G|$ is the number of statements in the graphs in the dataset plus the number of graphs used for generating the default graph, l is the maximum level of nestings of OPTIONAL and UNION patterns, and $|P|$ is the length of the graph pattern.

The proof is analogous to that for construct queries, with the only difference being that we translate the query to a single clause. With some simple optimisation, we can do better:

Lemma 3. *Complexity of optimised construct query translation.*

The translation of a SPARQL construct query (C, D, P) to datalog can be optimised to have complexity in $O(|T| + |G| + |C| + |P|)$.

Proof. First we introduce a new predicate s' with an arity of the number of variables used in C . We translate $|P|$ as usual, but generate a single clause with the head $s'(v_1, \dots, v_i)$ where v_1, \dots, v_i are the variables used in C in alphabetical order. Then we translate the construct pattern into clauses using s' as body of the clauses for binding the variables using their known order in s' . This reduces the complexity from $O(|T| + |D| + |C||2^l||P|)$ to $O(|T| + |D| + |C| + |2^l||P|)$.

Additionally, for every two clauses resulting from Lloyd-Topor transformation of a UNION or OPTIONAL pattern we can remove one clause:

If a bound filter is applied to a variable introduced in a nested OPTIONAL or UNION pattern, we can add an atom $bound(x)$ for every optional variable x to the body of the clause binding the optional variables. Now one of the two clauses under consideration is trivially false, as it contains $isNull(x)$ and $bound(x)$ (if bound is used positively) or $bound(x)$ and $\neg bound(x)$ (if bound is negated in the graph pattern).

If no bound filter is used, we only need to consider the clause which does bind the optional variables. The alternative clause is irrelevant, because (1) we can never derive a return value of “true” from a filter on an unbound variable and (2) if no filter is applied to an optional variable, the OPTIONAL pattern can not influence the result. As a result, the complexity is reduced to $O(|T| + |D| + |C| + |P|)$.

We can not do such an optimisation for select queries, as in this case null bindings are allowed in the query result. However, there are variations of Lloyd-Topor transformation, which avoid the exponential blowup of the program (cf. [6]). The reader may note that for construct queries we have also reduced the complexity of the evaluation - not only of Lloyd-Topor transformation, as we completely remove one branch for every disjunction.

6.2 Complexity of Query Evaluation

Now we investigate the overall complexity of SPARQL query evaluation. We assume that all filter expressions can be evaluated in polynomial time, an assumption which is true for all build in SPARQL filter functions defined in [12]. Our results correspond to results by Perez et al. [10] based on an algebraic semantics.

Theorem 1. *Complexity of SPARQL evaluation.*

The combined complexity of SPARQL query evaluation is LOGSPACE.

Proof. (sketch) Our SPARQL syntax is defined by combining simpler graph patterns into more complex ones, such that a tree-structure of graph pattern inclusions represents a query. This structure is preserved by the mapping function m as can easily be deduced from the mapping rules defined in section 5. While for construct queries $t/4$ is defined recursively in our above formalisation, it is easy to see that we can slightly modify the mapping to be hierarchical by introducing a new predicate for the query result. Hence, the resulting datalog program can be translated into a hierarchical one without influencing the result of SPARQL query evaluation. It contains negation in the theory for the bound filter. Thus the resulting program is in non-recursive datalog with negation. The complexity of non-recursive datalog with negation is LOGSPACE (cf. [4]).

Construct queries require all variables used in the construct pattern to be bound. Hence, if UNION or OPTIONAL are used, every translation of a graph pattern ends with an application of the *bound* filter which negatively depends on *isNull*. Construct queries without UNION or OPTIONAL on the other hand do not need negation. (Note that if UNION and OPTIONAL are not used, we can remove the bound filter, because in this case all variables are bound.) Select queries on the other hand allow null bindings in the result, so unless the bound filter is explicitly used, we do not need negation here. In those cases where the bound filter, is not used, we have polynomial complexity of SPARQL evaluation for hierarchical datalog without negation.

7 Extensions

Based on the semantics defined above, we propose two useful extensions to SPARQL.

7.1 Binding Variables to Filter Functions

As a first extension we propose to allow the binding of variables to values computed using filter expressions in SPARQL queries, if the filter expression does not return an error. For example given a graph describing Euro prices and conversion rates we could then compute dollar prices as proposed in the introductory example.

To express this variable binding, we slightly extend the definition of graph patterns:

Definition 11. *Graph patterns with functional bindings.*

Let V , R and I be defined as above.

If $s \in R \cup V$, $p \in I \cup V$ and $o \in R \cup V$, then (s, p, o) is a graph pattern.

If P_1 and P_2 are graph patterns, $(P_1 \ . \ P_2)$, $(P_1 \ \text{OPTIONAL} \ P_2)$ and $(P_1 \ \text{UNION} \ P_2)$ are graph patterns. If P is a graph pattern and F is a filter expression, $(P \ \text{FILTER} \ F)$ is a graph pattern. If P is a graph pattern, $v \in V$ and F is a filter expression, $(P \ \text{LET} \ v \ F)$ is a graph pattern. If P is a graph pattern and $n \in U \cup V$ then $(\text{GRAPH} \ n \ P)$ is a graph pattern.

We also need to slightly extend the normalisation rules. In order to define the semantics of the new *LET* expression, we add a new mapping rule to bind v to the result of F :

$$\text{N6) (GRAPH } g_1 (P_1 \text{ LET } v F) \qquad ((\text{GRAPH } g_1 P_1) \text{ LET } v F)$$

$$\frac{X \qquad m(X)}{15) (P \text{ LET } v F) \qquad m(P), m(F), \neg \text{isError}(\text{var}(F)), \text{eq}(m(v), \text{var}(F))}$$

The resulting program does not contain any constructs not already considered in the complexity discussion in the prior section. Using our introductory example, we can now ask for the dollarPrice of all products with a euroPrice of less than 9:

Example 7.

```
SELECT ?product ?dollarPrice
FROM <http://ex.org/g1>
WHERE {{{{{?product rdf:type <http://ex.org/product> .} .
        {?product <http://ex.org/euroPrice> ?euroPrice.}}} .
        {<http://ex.org/euro> <http://ex.org/dollarExchRate> ?rate}}}.
FILTER (?price < "9.0"^^xsd:float)} .
LET ?dollarPrice (?euroPrice * ?rate)}
```

7.2 Views in Datasets

The second extension allows to use views as parts of a dataset. We extend the dataset definition, such that in addition to existing named graphs also the results of SPARQL construct queries can comprise parts of the dataset. We name such views using newly introduced IRIRefs local to the query. Syntactically this change means that we also allow to write SPARQL CONSTRUCT queries at all places in the dataset, where IRIRefs of named graphs are allowed. Hence, we need to extend the definition of datasets.

Definition 12. *Datasets with views.*

Let n be a IRIRef which is a name of a named graph. Then FROM n is a dataset and FROM NAMED n is a dataset.

Let Q be a SPARQL CONSTRUCT query and b a new IRIRef. Then FROM Q is a dataset and FROM NAMED b Q is a dataset.

If D and D' are datasets, then $D + D'$ is a dataset.

Additionally we need to add two mapping rules to translate the extension into datalog. Let *resultGraph* be a skolem function from SPARQL construct queries to IRIRefs. We use *resultGraph(Q)* as name for a temporal named graph containing the result of the evaluation of Q . This named graph is then used in the dataset analogous to usual named graphs. Rule 11) needs to be extended to use this skolem function instead of a fixed graph. Analogously, we now need to name the default graph using a skolem function we call *defaultGraph*. Let P be the query the Dataset belongs to.

| X | m(X) |
|----------------------|--|
| 01) FROM g | $t(\text{defaultGraph}(P), x, y, z) \leftarrow t(m(g), x, y, z)$ |
| 13) (C, D, P) | $m(D) \cup \{$ $t(m(\text{resultGraph}(C,D,P)), m(s), m(p), m(o)) \leftarrow$ $m(P), \text{bound}(s), \text{bound}(p),$ $\text{isIRI}(p), \text{bound}(o) \mid (s,p,o) \in C\}$ |
| 15) FROM <Q> | $m(Q) \cup \{t(\text{defaultGraph}(P), x, y, z) \leftarrow$ $t(\text{resultGraph}(Q), x, y, z)\}$ |
| 16) FROM NAMED b <Q> | $m(Q) \cup \{t(m(b), s, p, o) \leftarrow$ $t(\text{resultGraph}(Q), s, p, o)\}$ |

With only this slight extension, it is possible to extend the dataset of a SPARQL query with views. The resulting query language is of the same complexity, because P only uses the results of the computation of the views in its dataset. Hence, we could evaluate the views first and then evaluate P itself, both with the known complexity of SPARQL.

We can now formulate an easy to understand query, doing the dollar conversion first in a view and then directly using the `dollarPrice`:

Example 8.

```
SELECT ?product, ?dollarPrice
FROM <http://ex.org/g1>
FROM <CONSTRUCT {?product <http://ex.org/dollarPrice> ?dollarPrice.}
FROM <http://ex.org/g1>
WHERE {{{?product <http://ex.org/euroPrice> ?euroPrice.} .
        {<http://ex.org/euro> <http://ex.org/dollarExchRate> ?rate}}}.
        LET ?dollarPrice (?euroPrice * ?rate)}}
WHERE {{{?product rdf:type <http://ex.org/product>}.
        {?product http://ex.org/dollarPrice ?dollarPrice.}
        FILTER (?dollarPrice < "9.0"^^xsd:float)}}
```

8 Related Work

Semantics of RDF, OWL and SPARQL de Bruijn et al. provide a mapping of RDF and OWL-Lite to first order predicate logic and prove this mapping equal to normative RDF [5]. Based on this mapping they can formulate goals corresponding to simple SPARQL graph pattern matching in first order logic. Volz, Motik et al. [8] describe how to map the web ontology language OWL [1] to datalog. A combination with the work presented here could easily add SPARQL support to these approaches.

Algebraic SPARQL Semantics. Perez et al. provide a formal semantics of a core fragment SPARQL in [10]. Their semantics does not include queries to multiple graphs and the evaluation of arbitrary filter expressions. In contrast our logic based mapping, the mapping in [10] is based on an algebraic evaluation of queries. Depending on the application context, our formalisation allows to build on a larger set of existing work. For example in [13] we need a SPARQL semantics supporting non-monotonic negation, which can not be easily modeled using an algebraic semantics.

Alternative Logic Based SPARQL Semantics. Polleres proposes an alternative mapping of SPARQL queries to logic programs [11]. In addition to our semantics, different possible semantics of joins for SPARQL are discussed. While the formalisation is similar to ours, arbitrary filter expressions are not supported. [11] has been published at the same time as [13] which contains the initial definition of our semantics.

Networked Graphs. In [13] we propose to extend named graphs, such that the content of a graph can be listed extensionally and defined intensionally through SPARQL based views to other graphs. A graph at least partially defined using this view mechanism is called a *networked RDF graph*. Networked RDF graphs allow to easily import, reuse and transform existing RDF data. As in a distributed, uncontrollable and dynamic setting like the semantic web, circular dependencies among networked RDF graphs can not be avoided, we use the logic mapping described here, but evaluated under the well founded semantics, to define the semantics of sets of mutually dependent networked graphs.

9 Conclusion

We have proposed a semantics for the RDF query language SPARQL based on datalog, and two useful extensions of SPARQL, namely the use of views in SPARQL datasets and the binding of variables to results of SPARQL filter functions. We use the semantics proposed here in [13] to define *networked RDF graphs*. Networked RDF graphs are an extension of named graphs such that a named graph can include views on other graphs. As on a web scale recursive dependencies of views can hardly be avoided, a logic based SPARQL semantics evaluated under a suitable semantics allows to deal with non-monotonic negation. Future work will include further useful extensions to SPARQL, for example aggregates, and distributed evaluation of SPARQL queries.

Acknowledgements

The author would like to thank Steffen Staab for his valuable advice and his contribution to the technical report, which this paper is based on, and the anonymous reviewers for their valuable feedback.

References

1. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: Owl web ontology language reference (2004), <http://www.w3.org/TR/owl-ref/>
2. Berners-Lee, T.: Notation 3 (2006), <http://www.w3.org/DesignIssues/Notation3>
3. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: WWW05, pp. 613–622. ACM Press, New York (2005)
4. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. *ACM Computing. Surveys.* 33(3), 374–425 (2001)

5. de Bruijn, J., Franconi, E., Tessaris, S.: Logical reconstruction of normative RDF. In: OWL: Experiences and Directions Workshop, Galway, Ireland, November 2005. CEUR Workshop Proceedings (2005), <http://www.debruijn.net/publications/owl-05.pdf>
6. Decker, S.: Semantic Web Methods for Knowledge Management. Phd thesis, University of Karlsruhe (February 2002)
7. Lloyd, J.W., Topor, R.W.: Making Prolog more expressive. Journal of Logic Programming 1(3), 225–240 (1984)
8. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Universität Karlsruhe (TH) (2006)
9. Hayes, P.: Rdf semantics, <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>
10. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006)
11. Polleres, A.: From SPARQL to rules (and back). Technical report (December 2006), <http://www.polleres.net/publications/GIA-TR-2006-11-28.pdf>
12. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2007), <http://www.w3.org/TR/rdf-sparql-query/>
13. Schenk, S., Staab, S.: Networked RDF Graphs. Technical report (December 2006) <http://uni-koblenz.de/~sschenk/publications/2006/ngtr.pdf>
14. van Gelder, A., Ross, K., Schlipf, J.S.: The Well-Founded Semantics for General Logic Programs. Journal of the ACM 38(3), 620–650 (1991)

A Theory for Handling Filter Expressions

The theory for three valued evaluation of conjunction:

$\text{and}(x, y, \text{error}) \leftarrow \text{isError}(x), \text{isTrue}(y)$
 $\text{and}(x, y, \text{error}) \leftarrow \text{isTrue}(x), \text{isError}(y)$
 $\text{and}(x, y, \text{error}) \leftarrow \text{isError}(x), \text{isError}(y)$
 $\text{and}(x, y, \text{true}) \leftarrow \text{isTrue}(x), \text{isTrue}(y)$
 $\text{and}(x, _ , \text{false}) \leftarrow \text{isFalse}(x)$
 $\text{and}(_ , y, \text{false}) \leftarrow \text{isFalse}(y)$

The theory for three valued evaluation of negation:

$\text{not}(x, \text{error}) \leftarrow \text{isError}(x)$
 $\text{not}(x, \text{true}) \leftarrow \text{isFalse}(x)$
 $\text{not}(x, \text{false}) \leftarrow \text{isTrue}(x)$

The theory for determining the effective boolean value:

$\text{isTrue}(\text{true}) \leftarrow$
 $\text{isFalse}(\text{false}) \leftarrow$
 $\text{isError}(\text{error}) \leftarrow$
 $\text{isEmptyString}(m(\text{""})) \leftarrow$
 $\text{isEmptyString}(m(\text{""}^\wedge \text{xsd:String})) \leftarrow$
 $\text{isNaN}(m(\text{NaN})) \leftarrow$
 $\text{isNonEmptyString}/1$ (external Predicate)

The theory for three valued evaluation of disjunction:

$\text{or}(x, y, \text{error}) \leftarrow \text{isFalse}(x), \text{isError}(y)$
 $\text{or}(x, y, \text{error}) \leftarrow \text{isError}(x), \text{isFalse}(y)$
 $\text{or}(x, y, \text{error}) \leftarrow \text{isError}(x), \text{isError}(y)$
 $\text{or}(x, _ , \text{true}) \leftarrow \text{isTrue}(x)$
 $\text{or}(_ , y, \text{true}) \leftarrow \text{isTrue}(y)$
 $\text{or}(x, y, \text{false}) \leftarrow \text{isFalse}(x), \text{isFalse}(y)$

Handling equality for the extension with LET

$\text{eq}(x, x) \leftarrow$

$\text{ebv}(x, \text{error}) \leftarrow \text{isError}(x)$
 $\text{ebv}(x, \text{true}) \leftarrow \text{isTrue}(x)$
 $\text{ebv}(x, \text{true}) \leftarrow x > 0$
 $\text{ebv}(x, \text{true}) \leftarrow \text{isNonEmptyString}(x)$
 $\text{ebv}(x, \text{false}) \leftarrow \text{isFalse}(x)$
 $\text{ebv}(x, \text{false}) \leftarrow \text{isEmptyString}(x)$
 $\text{ebv}(x, \text{false}) \leftarrow \text{isNaN}(x)$

Negation in Spatial Reasoning

A Computational Approach

Stefan Schleipen, Marco Ragni, and Thomas Fangmeier

Department of Computer Science,
Georges-Koehler-Allee, D-79110, Germany
{ragni,schleipe}@informatik.uni-freiburg.de,
{thomas.fangmeier}@uniklinik-freiburg.de

Abstract. In recent years a lot of research has been done in order to determine factors of complexity in spatial relational reasoning, like the number of models, the wording of conclusion or the influence of relational complexity. But research so far focused on affirmative statements only, i. e. negated expressions have not yet been investigated. In spatial reasoning and in human machine interaction, however, negation plays a fundamental role. Central questions are: How are negated statements represented? What happens in multiple-model cases? Which effects have different reference frames? We conducted three experiments to show that humans (i) negate a relation by using the opposite relation, (ii) construct preferred mental models and use an economic principle, and (iii) have more difficulties in reasoning with negated relations. The goal is to extend our cognitive and computational model – the SRM.

Keywords: Spatial Reasoning, Knowledge Representation and Reasoning, Cognitive modeling.

1 Introduction

There is a vast body of evidence supporting the mental model theory of spatial reasoning. The key idea of this theory is that reasoners translate spatial relations into a mental model and use this representation to solve spatial inference problems. To provide an example [7]:

The spoon is to the left of the knife.
The plate is to the right of the spoon.
The fork is in front of the spoon.
The cup is in front of the knife.

This describes the following two possible models:

| | | | | | |
|-------|-------|-------|-------|-------|-------|
| spoon | plate | knife | spoon | knife | plate |
| fork | | cup | fork | cup | |

Assume a child helps his mother to set the table. The child takes the knife and puts it to the left of the plate. But the mother says to the child ‘The knife

does not belong to the left of the plate'. Where will the child place the knife? Logically, there are three possibilities: the knife can be placed to the right of, in front of, or behind the plate (not considering that the knife could be placed above or under the plate).

Now, the question arises how such problems are processed? Is there a preferred interpretation? The mental model theory (MMT), introduced by [4], suggests that people draw conclusions by constructing and inspecting a spatial array that represents the state of affairs described in the premises. It is a three stage process consisting of a *comprehension*, *description*, and *validation phase*. In the comprehension phase, reasoners construct a mental model that reflects the information from the premises. If new information is encountered during the reading of the premises it is immediately used in the construction of the model. During the description phase, this model is inspected to find new information that is not explicitly given in the premises. Finally, in the validation phase alternative models are searched that refute this putative conclusion. However, some questions remain open with respect to how people deal with multi-model problems. For example, which model is constructed first, and does this model construction adhere to certain principles? Why do reasoners neglect some models?

In contrast, the preferred mental model theory (PMMT) has been developed to explain that humans in general tend to construct a preferred mental model (PMM). The PMM is the starting point for deriving a putative conclusion. In the model variation phase the participants tend to make local and continuous transformations starting from the PMM to search counter-examples [11].

How do humans process a premise like 'A is not to the left of C'? Do we remain in one dimension (by using the opposite relation only)? Which kind of insertion principle is then used? Kaup and colleagues focused on the negation in sentences and contradictory predicates [5]. They conducted a verification experiment in which participants had to verify sentences (e. g. the door is not open) and pictures of situations described in the sentence (e. g. closed door, open door). Reaction times were shorter if the sentence and the picture corresponded. Since there are only two states possible (the door is open or the door is closed) there is only one opposite state left. However, multiple model cases have not been investigated [6].

Hasson and Glucksberg [3] examined the difference in understanding affirmative and negated assertions in natural language. The participants had to make lexical decisions according to terms either to the affirmative or negative meaning. The results suggest that the affirmative assertion continued to facilitate affirmative-related terms, but the negated assertion did not. In the literature no work regarding negation in multiple model cases has been reported.

In artificial intelligence negation is sometimes interpreted as *negation as failure* [12]. It states that a negative literal, *not p*, can be proven true just in case the proof of *p* fails. Another investigation from different formal perspectives has been made by Gabbay and Wansing [1]. The object of our interest is, however, how humans interpret such negations and if they have maybe something in common with formal concepts.

In this paper, we analyze spatial problems with negated relations. The next section contains a formal analysis of negated spatial problems. Then, a representation of empirical data supporting our theory and an algorithmic approach will follow. Finally, we discuss the results presented in the paper and give a short overview of some questions that are left open.

2 Theoretical and Mathematical Approach

Theoretical Approach. First we discuss some representational issues about how negation can be interpreted and later a formal description on negation in different system environments is introduced.

Representation of Negation. There are merely three possibilities how negated relations could be represented in the human mind: The first possibility: A negated relation can be replaced by the opposite relation, i. e. ‘A is not to the left of B’ is replaced by ‘A is to the right of B’. If so, information is lost, but the initially generated model, however, is correct. In the variation phase it is likely that not all consistent models are generated, since variation in a different dimension, which is possible through negation, would not occur.

The second possibility is that the premise with the negated relation will not be altered and the to be inserted object will be placed in the excluded position and then annotated. This representation would generate an incorrect initial model. However, in the variation phase all consistent models can be generated. This strategy, however, would be more time consuming than creating a consistent model in the first place. Additionally, more operations in the validation and variation phase are necessary, since every time a new model is generated the model at hand has to be checked for correctness.

In a third possibility the premise at hand is interpreted by another consistent relation and the to be inserted object is additionally annotated with the negated relation. So a correct model is generated in the construction phase, but through annotation all alternative models (even in different dimensions) can be generated. The annotation prevents the loss of information as in the first possibility.

Connected with the question of representation in multiple model cases is the question, if preferred models are initially constructed and if they adhere to a common principle.

Insertion Principles. What do we know so far about construction principles in reasoning with non-negated descriptions? Assume two premises of the form: (1) ‘A is to the left of B’ and (2) ‘A is to the left of C’ are given. Humans tend to process such premises sequentially, i. e. first a model A B is generated and then object C is inserted into the model. There are two possibilities where C can be inserted: in-between A and B (*first fit principle*, ff-principle) and to the right of B (*first free fit principle*, fff-principle). So the ff-principle places the object directly next to the related object and, if necessary, other objects are displaced. The fff-principle implies that an object will be placed on the first free possible location

(according to the relation). Several empirical investigations have confirmed the latter principal is used by humans to generate the initial model (PMM) [8,9].

Mathematical Approach. How can negated relations mathematically be represented? First of all, an analysis of different interpretations of the base relations ('over, under, left, right') is necessary. This interpretation of a base relation like 'A is over B' can then be extended to an interpretation of a negated expression like 'A is not over B'.

Four systems can be discerned (Fig. 1). The first and the second approach are coordinate based, the third and fourth system are angle based.

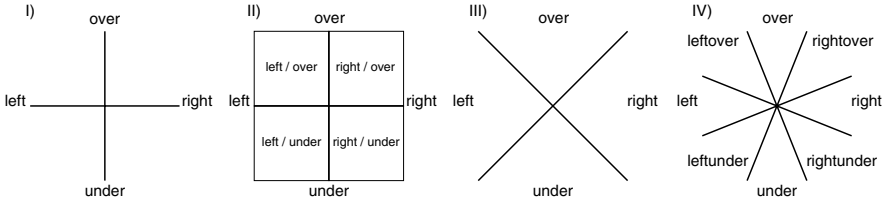


Fig. 1. Four possible systems I) discrete system, II) dense system, III) angle system, IV) system by Gapp [2]

First System. The first system is a discrete approach (Fig. 1, system I). In this case, the relations only permit locations for the related objects that are defined as follows:

$$\begin{aligned}
 \beta \models \text{over}(x,y) &\Leftrightarrow \beta(x') = \beta(y') \text{ and } \beta(x'') > \beta(y'') \\
 \beta \models \text{under}(x,y) &\Leftrightarrow \beta(x') = \beta(y') \text{ and } \beta(x'') < \beta(y'') \\
 \beta \models \text{left}(x,y) &\Leftrightarrow \beta(x') < \beta(y') \text{ and } \beta(x'') = \beta(y'') \\
 \beta \models \text{right}(x,y) &\Leftrightarrow \beta(x') > \beta(y') \text{ and } \beta(x'') = \beta(y'')
 \end{aligned}$$

x is a tuple (x', x'') with x' as the x-coordinate and x'' as the y-coordinate. β is called the truth assignment.

$$\begin{aligned}
 \beta \models \top &\Leftrightarrow (\text{over}(x, y) \models \top) \oplus (\text{under}(x, y) \models \top) \\
 &\oplus (\text{left}(x, y) \models \top) \oplus (\text{right}(x, y) \models \top)
 \end{aligned}$$

The exclusive 'or' \oplus indicates that only one condition can be true, and, therefore, all cases where none or more than one case is true the relation is false. We have to use the exclusive 'or' since the object can only be placed once in the model. \oplus is defined as: $x \oplus y \Leftrightarrow (x \wedge \neg y) \vee (\neg x \wedge y)$. The system only allows variation along the axes of the Euclidean space. The areas between the axes are not defined and therefore not a possible location. The negation of a relation only allows an interpretation along these axes. According to the truth assignment, the negation of one relation can only result in one of the remaining relations. The negated relation is defined by the disjunction of all relations ($r \in R$) except the one that is negated, thus $\neg r := (r | r \in R_{\setminus \{r\}})$.

The negation can be defined over the truth assignment as follows.

$$\begin{aligned}\beta \models \neg \text{over}(x, y) &\Leftrightarrow \neg \text{over}(x, y) \wedge \top \\ &\Leftrightarrow \neg \text{over}(x, y) \wedge ((\text{over}(x, y) \oplus \text{under}(x, y)) \\ &\quad \oplus \text{left}(x, y) \oplus \text{right}(x, y)) \\ &\Leftrightarrow \text{under}(x, y) \oplus \text{left}(x, y) \oplus \text{right}(x, y)\end{aligned}$$

Since $\neg \text{over}(x, y)$ and $\text{over}(x, y)$ cannot be true at the same time, $\text{over}(x, y)$ must be excluded. Exemplary below the negated relations for $\text{over}(x, y)$ and $\text{right}(x, y)$.

$$\begin{aligned}\beta \models \neg \text{over}(x, y) &\Leftrightarrow (\beta(x') = \beta(y') \wedge \beta(x'') < \beta(y'')) \\ &\quad \oplus (\beta(x') < \beta(y') \wedge \beta(x'') = \beta(y'')) \\ &\quad \oplus (\beta(x') > \beta(y') \wedge \beta(x'') = \beta(y'')) \\ \beta \models \neg \text{right}(x, y) &\Leftrightarrow (\beta(x') = \beta(y') \wedge \beta(x'') > \beta(y'')) \\ &\quad \oplus (\beta(x') = \beta(y') \wedge \beta(x'') < \beta(y'')) \\ &\quad \oplus (\beta(x') < \beta(y') \wedge \beta(x'') = \beta(y''))\end{aligned}$$

Second System. The second possible system (Fig. 1, system II) shows a dense approach. This system extends the discrete interpretation and allows objects to be placed in the areas between the axes of the Euclidean space. Thus

$$\begin{aligned}\beta \models \text{over}(x, y) &\Leftrightarrow (\beta(x') \leq \beta(y') \vee \beta(x') > \beta(y')) \wedge \beta(x'') > \beta(y'') \\ \beta \models \text{under}(x, y) &\Leftrightarrow (\beta(x') \leq \beta(y') \vee \beta(x') > \beta(y')) \wedge \beta(x'') < \beta(y'') \\ \beta \models \text{left}(x, y) &\Leftrightarrow \beta(x') < \beta(y') \wedge (\beta(x'') \leq \beta(y'') \vee \beta(x'') > \beta(y'')) \\ \beta \models \text{right}(x, y) &\Leftrightarrow \beta(x') > \beta(y') \wedge (\beta(x'') \leq \beta(y'') \vee \beta(x'') > \beta(y''))\end{aligned}$$

Here, the areas between the axes are not explicitly defined for a certain relation, rather is the definition of the area affected by the relation in the premise. If the premise ‘A is to the left of B’ is given, both areas left of the y-axis are interpreted as left (see Fig. 1, model II). The definition of the area only applies to the premise we actually look at.

As in the discrete model, the negation only excludes the position given by the negated relation. The definition of the negated relations is similar to the negated relations of the discrete model, only the areas between the axes have to be included. According to the truth assignment, the negated relations are defined as follows.

$$\begin{aligned}\beta \models \neg \text{over}(x, y) &\Leftrightarrow (\beta(x') \leq \beta(y') \oplus \beta(x') > \beta(y')) \wedge \beta(x'') < \beta(y'') \\ \beta \models \neg \text{right}(x, y) &\Leftrightarrow \beta(x') < \beta(y') \wedge (\beta(x'') \leq \beta(y'') \oplus \beta(x'') > \beta(y''))\end{aligned}$$

Third System. The third possible system (Fig. 1, system III) is angle based. In this system the interpretation of the relations is given by areas that are spanned between two angles. This system was inspired by the work of Gapp (Fig.1 system IV) [2]. In his work he generated a grid around an object with four different angles (0° , 22.5° , 45° , 67.5°) and relative distances (130, 240, 350, 460 pixels). He tested humans on how they accept or reject certain positions of objects on the grid given a specific relation. Since we have only four distinct

relations and we want to classify the complete space to get a dense model, we do not define subspaces such as leftover, leftover, rightover and rightunder. A reasonable allocation of the space would be four distinct areas of 90° each. According to this, our relations are defined as follows.

ϕ is the angle between the positive x-axis and the straight between the located object and the related object.

$$\begin{aligned} \textit{over}(x, y) &\Leftrightarrow \phi \in [45^\circ, 135^\circ] \\ \textit{under}(x, y) &\Leftrightarrow \phi \in [225^\circ, 315^\circ] \\ \textit{left}(x, y) &\Leftrightarrow \phi \in [135^\circ, 225^\circ] \\ \textit{right}(x, y) &\Leftrightarrow \phi \in [315^\circ, 45^\circ] \end{aligned}$$

The negation of a relation allows a position of the located object in an area excluding the area defined by the relation itself. The resulting definitions of the negated relations are shown below.

$$\begin{aligned} \neg\textit{over}(x, y) &\Leftrightarrow \phi \in [135^\circ, 45^\circ] \\ \neg\textit{right}(x, y) &\Leftrightarrow \phi \in [45^\circ, 315^\circ] \end{aligned}$$

We expect one of these systems to be an approach to the human interpretation. Furthermore, we are convinced that humans will work with PMMs when generating mental models with negated relations. Since the located object can only be positioned in one of the possible locations, we expect that this position will be the converse of the negated relation. So the PMM will be the model which contains a premise with the converse relation instead of the negated. If so, we can use our mathematical system to formalize this PMM.

3 Empirical Data

We present three experiments on how humans generate and inspect mental models out of given premises when the relation of a premise is negated. First, we questioned which relations between two objects were accepted if a relation was negated or not? Second, we are interested in the generation process: (i) How are objects inserted into a model if the relation to another object was negated? (ii) Do participants use certain relations if a model contains a negated relation in a given premise which leads to a preferred mental model during the construction process? (iii) Are the preferred mental models with negated problems different from indeterminate positive problems? Third, we examined the constructed model that participants had in mind: (i) Which influences have different construction directions if a model was built from left to right or from right to left and shapes? (ii) Are there differences between indeterminate and negated problems during the inspection phase?

We assume that the participants interpret the negation of the relation as the logical negation in the same dimension. According to this hypothesis, we expect for the premise ‘A is not to the left of B’ that the participants construct a model in which ‘A is to the right of B’. Another assumption was that models with negated relations are harder to obtain than models without negation. A further

assumption was that the complexity of the model that participants held in mind are higher if a relation was negated in comparison to an indeterminate one.

3.1 First Experiment - Acceptance

In this experiment the participants had to accept or reject a given statement about relations between two objects.

Participants, Materials, Procedure and Design. Thirty six students of the University of Freiburg participated in this experiment (with/without grid: $n = 20/16$, $M = 24.3/24$, $SD = 2.4/2.8$). The participants were presented with pictures of two related objects and a statement. Fig. 2 shows examples with (I) and without (II) an underlying grid. The letter A had a fixed position in the center while the letter B was randomly swapped over the other 48 free cells in the grid. Every possible constellation of A and B was presented with a statement ('B is not over A', 'B is not right of A'). We also asked for 'B is over A' and 'B is right of A' in order to compare the data with positive cases. The last two statements were tested on 16 of the 48 possible cases (see Fig. 3 II). Reaction time and accuracy were recorded for each statement.

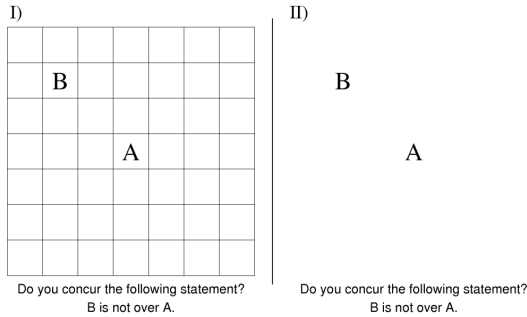


Fig. 2. With underlying grid (*section I*), without underlying grid (*section II*). Underneath the statement with negated relation of the two objects.

Results. The participants made a clear decision for affirmative (right/over) and negated (not right/not over) statements. Fig. 3 indicates that the distinction whether or not B is over/not over A is clear. In both cases (with or without underlying grid) the results are similar for all four statements (over/not over and right/not right). Reaction time for the negation problems with or without underlying grid for 'not over' is significantly longer than for 'not right' (with/without grid: $t = 7.076/5.589$, $df = 19/15$, $p \leq 0.01$), as well as positive problems for 'over' in comparison to 'right' (with/without grid: $t = 3.326/4.062$, $df = 19/15$, $p \leq 0.01$). In most cases the reaction time is significantly shorter (see Fig. 3) if the statement and the actual state of the relation of A and B is true (with/without grid: 'not over' $t = 0.288/4.124$, $df = 19/15$, $p = n.s./p \leq$

0.01; 'not right' $t = 1.717/3.186, df = 19/15, p = n.s./p \leq 0.05$; 'over' $t = 2.810/3.550, df = 19/15, p \leq 0.05/p \leq 0.01$; 'right' $t = 4.157/2.422, df = 19/15, p \leq 0.001/p \leq 0.05$).

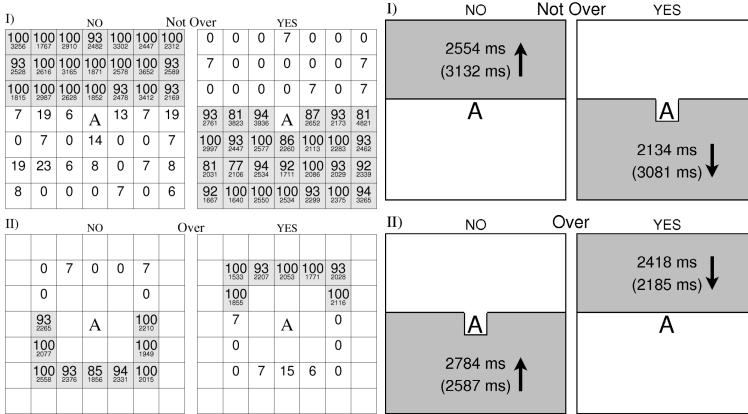


Fig. 3. Left:I) Statement 'not over'. Left square: over-all decisions in percent (bigger numbers) for NO. Right square: reaction time for YES answers (correct answers marked gray). The smaller numbers contain the reaction times for the correct decisions. II) Shows the positive statement 'over' (all other information is similar to I). **Right: I)** Statement 'not over'. Left square: over-all reaction time for NO. Right square: reaction time for YES answers (only correct answers without underlying grid). The numbers in parenthesis contain the reaction time with underlying grid. II) Shows the positive statement 'over' (all other information is similar to I).

3.2 Second Experiment - Simple Generating Experiment

In this experiment we investigated how people construct a model if premises contain negated relation between two objects. Additionally, we analyzed if participants construct a preferred mental model (PMM).

Participants, Materials, Procedure and Design. Twenty three students of the University of Freiburg took part in this experiment (age: $M = 25.8, SD = 4.5$). It was designed as pen and paper experiment consisting of sixteen problems (Table I) in which the participant had to construct a mental model out of four given premises. This model should then be drawn on a sheet of paper. The models were varied in the dimension (one- and two-dimensional), determination (determinate and indeterminate) and negation (affirmative and negated). Every model was presented twice but had different term names (total of 16). All of the 16 problems were constructed in the same way. Four premises arranged five different objects with the relations left, right, over or under. For negation the relation of the third premise was always negated. Note that models with negation were always indeterminate due to the undetermined position of the object. In

order to guarantee that a model was only constructed in working memory, each problem contained three pages thereby delaying the information on premises. The first two premises were given on the first page, premises three and four on page two and page three was blank. The participants were asked to draw only one model even if multiple models could be constructed. Additionally, the participants were instructed not to use any kind of aid (no sketch, etc).

Table 1. Table contains four premises for the positive I) and negative II) problems for one-dimensional (a, b) and two-dimensional (c, d) problems as well as for determinate (a, c) and indeterminate (b, d) problems

| Problem | PMM/alternative models |
|--|---|
| (a) A is to the left of B. B is to the left of C. I) <i>C is to the left of D.</i> II) <i>C is not to the right of D.</i> D is to the left of E. | (1) A B C D E |
| (b) A is to the left of B. B is to the left of C. I) <i>D is to the right of B.</i> II) <i>D is not to the left of B.</i> D is to the left of E. | (1) A B C D E (2) A B D E C (3) A B D C E |
| (c) A is over B. B is to the left of C. I) <i>C is to the left of D.</i> II) <i>C is not to the right of D.</i> D is under E. | (1) A E B C D |
| (d) A is over B. B is to the left of C. I) <i>D is to the right of C.</i> II) <i>D is not to the left of C.</i> D is under E. | (1) A E B C D (1) A E B D C |

Results. The correct answers indicate, that answers to one-dimensional problems are significantly more often correct than two-dimensional problems (Wilcoxon-Test: $Z = 3.109, p = 0.002$). Furthermore, there is a significant difference between affirmative and negative problems (Wilcoxon-Test: $Z = 2.618, p = 0.009$). However, there is no significant difference between determinate and indeterminate problems.

An additional question was how participants understand negated problems. Because if one direction is negated, then all other possible directions are allowed. There was a stable preference for the opposite direction in negated problems. Table 3 shows that except for indeterminate two-dimensional problems the use of the opposite direction was significantly more frequent. A further question was the preference for a model. For both dimensions we found a significant difference from null for the PMM. But we did not find significant differences when we analyzed only the indeterminate problems for affirmative versus negated problems.

Table 2. Table shows the correct responses (in percent) for one- and two-dimensional, affirmative and negative, as well as determinate and indeterminate problems

| | 1-dim | | 2-dim | |
|--------|-------|------|-------|------|
| | Aff. | Neg. | Aff. | Neg. |
| Det. | 87 | 78 | 76 | 52 |
| Indet. | 85 | 78 | 67 | 57 |

Table 3. Table shows the preference for the opposite direction in percent for one- and two-dimensional, as well as determinate and indeterminate negated problems. The numbers divided with colons denote the number of correct answers for the opposite direction in comparison to all correct answers. The last row indicates the proportion of preferred models (fff) in comparison to the other principle (ff). Note that models for determinate negated problems in this task do not provide the discrimination between preferred and alternative models. * $p \leq 0.05$, ** $p \leq 0.01$, *** $p \leq 0.001$.

| | | 1-dim. | 2-dim. |
|--------|---------------|-----------|-----------|
| Det. | opposite | 81%*** | 83%** |
| | opposite: all | 29:36 | 20:24 |
| Indet. | opposite | 75%* | 65% |
| | opposite: all | 27:36 | 17:26 |
| | fff / ff | 24*** / 2 | 14*** / 3 |

3.3 Third Experiment - Complexity of Proof

In the third experiment we were interested in the complexity of proof on PMMs. The participants had to generate a model and then to validate if a given statement holds or not.

Participants, Materials, Procedure and Design. Sixteen students of the University of Freiburg from the age of 21 to 30 ($M = 24.3, SD = 2.4$) participated in this experiment. Two participants were excluded due to the low accuracy rate ($< 50\%$) in determinate problems. We conducted a computer experiment in order to measure reaction time and accuracy as well as reading time for given premises. The experiment contained 20 problems, ten one-dimensional and ten two-dimensional (see Tab. 4).

All four premises were presented simultaneously on the computer screen. After pressing a key the premises disappeared and a statement was presented. One object of the statement was taken from the third premise and the other object from another premise. This guaranteed that the participant had to prove the model with the negated relation and had to infer an implicit relation between two objects. The relation in the statement was always missing so that the participant had to fill in the correct answer or in case of indetermination a relation that seemed the most possible. There were four possible relations for an answer: *left, right, over and under*.

Table 4. Table shows the subject matter of the problems. The italics indicates the three different types of the model negated (3a), indeterminate (3b) and determinate (3c). Half of the problems had the relation 'under' I) in the fourth premise, the other 'over' II).

| Problem | PMM/alternative models |
|--|--|
| (a) A is over B. B is to the left of C. <i>3a D is not to the left of B.</i> <i>3b D is to the right of B.</i> <i>3c C is to the left of D.</i> D is under E. (I) / D is over E. (II) | (I) $A \ E$ $B \ C \ D$ (II) A $B \ C \ D$ E |
| (b) A is over B. B is to the right of C. <i>3a D is not to the right of B.</i> <i>3b D is to the left of B.</i> <i>3c C is to the right of D.</i> D is under E. (I) / D is over E. (II) | (I) $E \ A$ $D \ C \ B$ (II) E $D \ C \ B$ A |

We found no differences in the premise reading times between determinate, indeterminate, and negated problems. The different shapes and dimension were not different either.

Results. Again we found a strong preference for PMM (indeterminate/negated: alternative models = 24%; PMM = 76%; Binomial-Test $p \leq 0.001$). The accuracy of the answers decreased significantly from determinate to indeterminate to negated problems (Page-L Test $N = 14, k = 3, L = 178, p \leq 0.05$).

4 Algorithmic Approach

In the following we outline an example how negated premises are interpreted and resolved. The algorithm parses the premises into an interpreter and then constructs and validates the resolving model.

Different types of premises are to be discerned: Premises of type 1 are determined premises, i. e. the object can be placed directly next to the related object. This is only possible if no other object already occupies this position. Premises of type 2 are called indetermined, i. e. in contrast to premises of type 1, there is already an object on the insertion position. Therefore an insertion principle (like

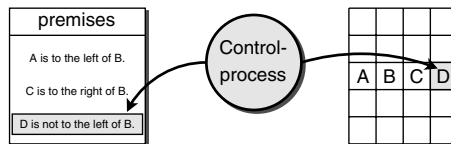


Fig. 4. The SRM processing a negated expression

ff- or fff-principles) has to be applied. Type 3 premises are those premises that contain negated relations. The premise is interpreted by the opposite relation, (e.g. $\neg \text{right} \Rightarrow \text{left}$). In this case the located object is annotated in order to recognize the excluded position. This is necessary for the later validation phase.

In case of premises of type 2 and type 3, only one model is generated. Although in case of indetermination, there can be several consistent models. In both cases, the preferred mental model (PMM) is constructed. During the construction phase, indetermined objects are annotated so that in the variation phase alternative models can be validated without reading the premises again.

In the variation phase the annotated objects will be shifted step by step towards an anchor object (related object) until it is reached. In every shift the model at hand is checked on correctness. Models which contains negated relations are more complex in the variation phase. If we have more than one dimension, it is not enough to just shift the located object towards the anchor.

The processing of the SRM is depicted in Fig. 4. The interpretation of the premises and the model generation is done by a control process using the premise input and a two dimensional array.

The algorithm we use here is a revised version of the algorithm used by [10] for the SRM. The changes apply mostly in the variation phase. In the creation and validation phase only a new type of annotation is introduced. The following pseudo code contains the extensions for handling negated relations. The complete algorithm without negation is described in [10]. In the construction phase (Fig. 5) we included the case of a negated relation. The algorithm checks which object is new to the model. If both objects are new in the model, a new layer is created and both objects are placed in this layer. When the premise contains objects that are in different layers, the layers will be merged. In the case that both objects are already in the model and in the same layer, a model revision step will be inserted. In every case the located object will be annotated with the negated premise. In case that only one object of the premise is already in the model, the missing object is included according to the relation and will be annotated. The annotation depends on which object is missing. If the related object is missing the annotation is the original premise. In the other case the annotation is the negated inverse relation ($\neg \text{left} \rightarrow \neg \text{right}$).

The validation phase is similar to the algorithm of [10], except that we included a function to check if the negated premise is fulfilled in the constructed model. If so, the model is invalid.

In the variation phase the conclusion should be checked if it is consistent in all models. If it is not consistent, a counter-example will be generated by small local transformations [10]. The key difference is that in the variation phase an object can be shifted not only in one direction but in all possible directions which are not prohibited by the annotation. Therefore, the algorithm shifts the object towards the anchor object even, if necessary, on another level but keeping the direction (e.g. if 'C is not left of D' we can shift C in the area over or under D).

```

def processNegatedPremise():
if negPremise():
{ while readnext() do
  { if case1 then      #one object already in layer
    { fmove focus to contained obj
      while not placed do
        { if fread() then
          if contained(RO)
          { fmove(inverse(REL))
            else
              fmove(REL) }
          else
            fwrite missing obj
            annotate missing obj
            placed = true } }
        if case2 then      #both objects new and not in layer
        { l = newLayer()
          fwrite(RO):
          fmove(inverse(REL))
          fwrite(LO)
          annotate LO }
        if case3 then      #both object are in different layers
        { merge(layer(LO), layer(RO))
          annotate LO }
        if case4 then      #both in model and same layer
        { newModel=valConcl(LO,inverse(REL),RO)
          if newModel then
            writeM0del()
            annotate LO } }
    }
}

```

Fig. 5. An algorithm for processing negated premises in the construction phase, RO = related object, LO = located object, REL = relation

5 General Discussion and Outlook

Without negated relations relational reasoning seems to be inherently incomplete. But how do humans reason with negated relations? Our formal analysis revealed that there are at least three possible interpretations. Some previous research has covered the linguistic processing and comprehension. Kaup and colleagues showed that the processing of matching sentence and picture are easier if the sentence and the picture correspond [5]. Hasson and Glucksberg examined the question if negated information entails affirmation [3]. They were able to show that negated metaphors are most likely represented as affirmation. This goes along with the results of our empirical investigations and the third possible representation of negation. In addition, in spatial reasoning multiple model cases are possible. Therefore, the negation of a spatial relation is not necessarily the opposite relation. Though the information about other possible models has to be stored. In this case it seems reasonable to adapt an approach of Vandierendonck, Dierckx, and De Vooght [13] for positive indeterminate model cases, to represent the alternatives by annotations at the object that is related with negation in the initial premises.

Our formal investigation is confirmed by our own empirical data. Our first experiment shows that there is a significant difference in the interpretation of ‘right/over’ versus ‘not right/not over’ and there is no empirical difference with

or without using a grid based environment. This contrasts the results by Gapp [2], who was more interested in the acceptability of spatial relations, where we wanted to know what exact model (i. e. which unique interpretation) participants construct if they are forced to. There are definitively preferred mental models in reasoning with negated assertions, and in indetermined cases, the participants constructed the preferred models by using the fff-principle [8]. The existence of an economic principle has been confirmed and it is persistent for one- and two-dimensional problems. This indicates that the fff-principle is an economic principle of representation for negated relation as well as for indetermined cases. A comparison of indeterminate model cases of non-negative premises with negated premises shows that the accuracy of the answers decreased from indeterminate positive premises to negated premises (Experiment 3).

Finally, the results are implemented into the SRM. Due to the analysis and integration of the role of negation the SRM is enhanced to a more cognitive-adequate computational model. Additionally the new insights gained in Experiment 1 are a help in extending the discrete to a dense structure and representing negation adequately. This results in a more precise computational simulation of the human reasoning process. The SRM constructs the same preferred mental models that were empirically confirmed in Experiment 2. Furthermore, the SRM can explain the result (Experiment 3) by the use of a complexity measure that reasoning with negation is more difficult than reasoning with positive indeterminate cases.

Future work will cover aspects of the understanding of negated expressions in a non-European country, how reasoners find and neglect counter-examples and extending the results to three-dimensional reference frames. It would be also of interest to investigate tasks with several negated premises and connections to belief revision.

Acknowledgments. This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition. We like to thank Bernhard Nebel and Markus Knauff for various helps. We also owe thanks to three anonymous reviewers for their helpful comments.

References

1. Gabbay, D., Wansing, H.: *What is Negation?* Oxford University Press (1999)
2. Gapp, K.P.: Angle, distance, shape and their relationship to projective relations. In: Moore, J.D., Lehman, J.F. (eds.) *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pp. 112–117. Lawrence Erlbaum Associates Inc. (1995)
3. Hasson, U., Glucksberg, S.: Does understanding negation entail affirmation? an examination of negated metaphors. *Journal of Pragmatics* (forthcoming)
4. Johnson-Laird, P.N., Byrne, R.M.J.: *Deduction*. Erlbaum, Hillsdale, NJ (1991)
5. Kaup, B., Luedtke, J., Zwaan, R.A.: Processing negated sentences with contradictory predicates: Is a door that is open mentally closed? *Journal of Pragmatics* 38, 1033–1050 (2006)

6. Knauff, M.: Deduktion und logisches Denken. In: Funke, J. (ed.) Denken und Problemlösen. Enzyklopädie der Psychologie, vol. 8, Hogrefe, Göttingen (2006)
7. Mani, K., Johnson-Laird, P.N.: The mental representation of spatial descriptions. *Memory & Cognition* 10(2), 181–187 (1982)
8. Ragni, M., Fangmeier, T., Webber, L., Knauff, M.: Complexity in spatial reasoning. In: Proceedings of the 28th Annual Cognitive Science Conference, Mahwah, NJ, Lawrence Erlbaum Associates, Mahwah (2006)
9. Ragni, M., Knauff, M., Nebel, B.: A computational model for spatial reasoning with mental models, pp. 1064–1070. Erlbaum, Mahwah, NJ (2005)
10. Ragni, M., Steffenhagen, F.: An implementation of the srm-model. Technical report 011-09/2006, SFB/TR 8 Spatial Cognition (2006), <http://www.sfbtr8.uni-bremen.de>
11. Rauh, R., Hagen, C., Knauff, M., Kuss, T., Schlieder, C., Strube, G.: Preferred and alternative mental models in spatial reasoning. *Spatial Cognition and Computation* 5, 239–269 (2005)
12. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*, 2nd edn. Prentice-Hall, Englewood Cliffs (2003)
13. Vandierendonck, A., Dierckx, V., Vooght, G.D.: Mental model construction in linear reasoning: Evidence for the construction of initial annotated models. *The Quarterly Journal of Experimental Psychology* 57A, 1369–1391 (2004)

Relational Neural Gas

Barbara Hammer and Alexander Hasenfuss

Clausthal University of Technology, Institute of Computer Science,
Clausthal-Zellerfeld, Germany

Abstract. We introduce relational variants of neural gas, a very efficient and powerful neural clustering algorithm, which allow a clustering and mining of data given in terms of a pairwise similarity or dissimilarity matrix. It is assumed that this matrix stems from Euclidean distance or dot product, respectively, however, the underlying embedding of points is unknown. One can equivalently formulate batch optimization in terms of the given similarities or dissimilarities, thus providing a way to transfer batch optimization to relational data. For this procedure, convergence is guaranteed and extensions such as the integration of label information can readily be transferred to this framework.

1 Introduction

Topographic maps such as the self-organizing map (SOM) constitute a valuable tool for robust data inspection and data visualization which has been applied in diverse areas such as telecommunication, robotics, bioinformatics, business, etc. [18]. Alternative methods such as neural gas (NG) [22] provide an efficient clustering of data without fixing a prior lattice. This way, subsequent visualization such as multidimensional scaling [21] can readily be applied, whereby no prior restriction of a fixed lattice structure as for SOM is necessary and the risk of topographic errors is minimized. For NG, an optimum (nonregular) data topology is induced such that browsing in a neighborhood becomes directly possible [23].

In the last years, a variety of extensions of these methods has been proposed to deal with more general data structures. This accounts for the fact that more general metrics have to be used for complex data such as microarray data or DNA sequences. Further it might be the case that data are not embedded in a vector space at all, rather, pairwise similarities or dissimilarities are available.

Several extensions of classical SOM and NG to more general data have been proposed: a statistical interpretation of SOM as considered in [5,14,30,31] allows to change the generative model to alternative general data models. The resulting approaches are very flexible but also computationally quite demanding, such that proper initialization and metaheuristics (e.g. deterministic annealing) become necessary when optimizing statistical models. For specific data structures such as time series or recursive structures, recursive models have been proposed as reviewed e.g. in the article [10]. However, these models are restricted to recursive data structures with Euclidean constituents. Online variants of SOM and NG have been extended to general kernels e.g. in the approaches presented in [27,34] such that the processing of nonlinearly preprocessed data becomes available. However, these versions have been derived for (slow) online adaptation only.

The approach [20] provides a fairly general method for large scale application of SOM to nonvectorial data: it is assumed that pairwise similarities of data points are available. Then the batch optimization scheme of SOM can be generalized by means of the generalized median to a visualization tool for general similarity data. Thereby, prototype locations are restricted to data points. This method has been extended to NG in [3] together with a general proof of the convergence of median versions of clustering. Further developments concern the efficiency of the computation [2] and the integration of prior information if available to achieve meaningful visualization and clustering [6,7,32].

Median clustering has the benefit that it builds directly on the derivation of SOM and NG from a cost function. Thus, the resulting algorithms share the simplicity of batch NG and SOM, its mathematical background and convergence, as well as the flexibility to model additional information by means of an extension of the cost function. However, for median versions, prototype locations are restricted to the set of given training data which constitutes a severe restriction in particular for small data sets. Therefore, extensions which allow a smooth adaptation of prototypes have been proposed e.g. in [8]. In this approach, a weighting scheme is introduced for the points which represents virtual prototype in the space spanned by the training data. This model has the drawback that it is not an extension of the standard Euclidean version.

Here, we use an alternative way to extend NG to relational data given by pairwise Euclidean similarities or dissimilarities, respectively, which is similar to the relational dual of fuzzy clustering as derived in [12,13]. For a given distance matrix or Gram matrix which stems from a (possibly high-dimensional and unknown) Euclidean space, it is possible to derive the relational dual of topographic map formation which expresses the relevant quantities in terms of the given matrix and which leads to a learning scheme similar to standard batch optimization. This scheme provides identical results as the standard Euclidean version if an embedding of the given data points is known. In particular, it possesses the same convergence properties as the standard variants, thereby restricting the computation to known quantities which do not rely on an explicit embedding. Since these relational variants rely on the same cost function, extensions to additional label information or magnification control [6,7,9] become readily available. Further, convergence of the algorithm is guaranteed for every symmetric nonsingular matrix which need not be Euclidean or stem from a metric.

In this contribution, we first introduce batch learning algorithms for neural gas based on a cost function. Then we derive the respective relational dual resulting in a dual cost function and batch optimization schemes for the case of a given distance matrix of data or a given Gram matrix, respectively. We demonstrate the possibility to extend this model to supervised information, and we show the performance in a variety of experiments.

2 Neural Gas

Neural clustering and topographic maps constitute effective methods for data preprocessing and visualization. Classical variants deal with vectorial data $\mathbf{x} \in \mathbb{R}^n$ which are distributed according to an underlying distribution P in the Euclidean plane. The goal of neural clustering algorithms is to distribute prototypes

$\mathbf{w}^i \in \mathbb{R}^n, i = 1, \dots, k$ among the data such that they represent the data as accurately as possible. A new data point \mathbf{x} is assigned to the *winner* $\mathbf{w}^{I(\mathbf{x})}$ which is the prototype with smallest distance $\|\mathbf{w}^{I(\mathbf{x})} - \mathbf{x}\|^2$. This clusters the data space into the receptive fields of the prototypes.

Different popular variants of neural clustering have been proposed to learn prototype locations from given training data [18]. Assume the number of prototypes is fixed to k . Simple k-means directly optimizes the *quantization error*

$$E_{k\text{-means}}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^k \int \delta_{i,I(\mathbf{x})} \cdot \|\mathbf{x} - \mathbf{w}^i\|^2 P(d\mathbf{x})$$

where $\delta_{i,I(\mathbf{x})}$ with Kronecker δ -function indicates the winner neuron for \mathbf{x} . Given a finite set of training data $\mathbf{x}^1, \dots, \mathbf{x}^m$, a batch training algorithm can be directly derived from the cost function, subsequently optimizing the winner assignments, treated as hidden variables of the cost function, and the prototype locations:

```

init  $\mathbf{w}^i$ 
repeat
    compute optimum assignments  $I(\mathbf{x}^j)$  which minimize  $\|\mathbf{x}^j - \mathbf{w}^{I(\mathbf{x}^j)}\|^2$ 
    compute new prototype locations  $\mathbf{w}^i = \sum_j \delta_{i,I(\mathbf{x}^j)} \cdot \mathbf{x}^j / \sum_j \delta_{i,I(\mathbf{x}^j)}$ 
    
```

K-means constitutes one of the most popular clustering algorithms for vectorial data and can be used as a preprocessing step for data mining and data visualization. However, it is quite sensitive to initialization.

Unlike k-means, neural gas (NG) [22] incorporates the neighborhood of a neuron for adaptation. The cost function is given by

$$E_{\text{NG}}(\mathbf{w}) = \frac{1}{2C(\lambda)} \sum_{i=1}^k \int h_\lambda(k_i(\mathbf{x})) \cdot \|\mathbf{x} - \mathbf{w}^i\|^2 P(d\mathbf{x})$$

where

$$k_i(\mathbf{x}) = |\{\mathbf{w}^j \mid \|\mathbf{x} - \mathbf{w}^j\|^2 < \|\mathbf{x} - \mathbf{w}^i\|^2\}|$$

is the rank of the prototypes sorted according to the distances, $h_\lambda(t) = \exp(-t/\lambda)$ scales the neighborhood cooperation with neighborhood range $\lambda > 0$, and $C(\lambda)$ is the constant $\sum_{i=1}^k h_\lambda(k_i(\mathbf{x}))$. The neighborhood cooperation smoothes the data adaptation such that, on the one hand, sensitivity to initialization can be prevented, on the other hand, a data optimum topological ordering of prototypes is induced by linking the respective two best matching units for a given data point [23]. Classical NG is optimized in an online mode. For a fixed training set, an alternative fast batch optimization scheme is offered by the following algorithm, which in turn computes ranks, which are treated as hidden variables of the cost function, and optimum prototype locations [3]:

```

init  $\mathbf{w}^i$ 
repeat
    compute ranks  $k_i(\mathbf{x}^j) = |\{\mathbf{w}^k \mid \|\mathbf{x}^j - \mathbf{w}^k\|^2 < \|\mathbf{x}^j - \mathbf{w}^i\|^2\}|$ 
    compute new prototype locations  $\mathbf{w}^i = \sum_j h_\lambda(k_i(\mathbf{x}^j)) \cdot \mathbf{x}^j / \sum_j h_\lambda(k_i(\mathbf{x}^j))$ 
    
```

Like k-means, NG can be used as a preprocessing step for data mining and visualization, followed e.g. by subsequent projection methods such as Sammon’s mapping or multidimensional scaling.

It has been shown in e.g. [3] that batch optimization schemes of these clustering algorithms converge in a finite number of steps towards a (local) optimum of the cost function, provided the data points are not located at borders of receptive fields of the final prototype locations. In the latter case, convergence can still be guaranteed but the final solution can lie at the border of basins of attraction.

3 Relational Data

Relational data x^i are not explicitly embedded in a Euclidean vector space, rather, pairwise similarities or dissimilarities are available. Batch optimization can be transferred to such situations using the so-called generalized median [3,20]. Assume, distance information $d(x^i, x^j)$ is available for every pair of data points x^1, \dots, x^m . Median clustering reduces prototype locations to data locations, i.e. adaptation of prototypes is not continuous but takes place within the space $\{x^1, \dots, x^m\}$ given by the data. We write w^i to indicate that the prototypes need no longer be vectorial. For this restriction, the same cost functions as beforehand can be defined whereby the Euclidean distance $\|\mathbf{x}^j - \mathbf{w}^i\|^2$ is substituted by $d(x^j, w^i) = d(x^j, x^{l^i})$ whereby $w^i = x^{l^i}$. Median clustering substitutes the assignment of \mathbf{w}^i as (weighted) center of gravity of data points by an extensive search, setting w^i to the data points which optimize the respective cost function for fixed assignments. This procedure has been tested e.g. in [3,6]. It has the drawback that prototypes have only few degrees of freedom if the training set is small. Thus, median clustering usually gives inferior results compared to the classical Euclidean versions when applied in a Euclidean setting.

Here we introduce relational clustering for data characterized by similarities or dissimilarities, using a direct transfer of the standard Euclidean training algorithm to more general settings allowing smooth updates of the solutions. The essential observation consists in a transformation of the cost functions as defined above to their so-called relational dual. We distinguish two settings, similarity data where dot products of training data are available, and dissimilarity data where pairwise distances are available.

3.1 Metric Data

Assume training data x^1, \dots, x^m are given in terms of pairwise distances $d_{ij} = d(x^i, x^j)^2$. We assume that it originates from a Euclidean distance measure, that means, we are always able to find (possibly high dimensional) Euclidean points \mathbf{x}^i such that $d_{ij} = \|\mathbf{x}^i - \mathbf{x}^j\|^2$. Note that this notation includes a possibly nonlinear mapping (feature map) $x^i \mapsto \mathbf{x}^i$ corresponding to the embedding in a Euclidean space. However, this embedding is not known, such that we cannot directly optimize the above cost functions in the embedding space. The key observation is based on the fact that k-means and batch NG optimum prototype locations \mathbf{w}^j can be expressed as linear combination of data points. Therefore, the unknown values $\|\mathbf{x}^j - \mathbf{w}^i\|^2$ can be expressed in terms of known values d_{ij} .

More precisely, assume there exist points \mathbf{x}^j such that $d_{ij} = \|\mathbf{x}^i - \mathbf{x}^j\|^2$. Assume the prototypes can be expressed in terms of data points $\mathbf{w}^i = \sum_j \alpha_{ij} \mathbf{x}^j$ where $\sum_j \alpha_{ij} = 1$. Then

$$\|\mathbf{w}^i - \mathbf{x}^j\|^2 = (D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$$

where $D = (d_{ij})_{ij}$ is the distance matrix and $\alpha_i = (\alpha_{ij})_j$ are the coefficients. This fact can be shown as follows: for $\mathbf{w}^i = \sum_j \alpha_{ij} \mathbf{x}^j$, one can compute

$$\|\mathbf{x}^j - \mathbf{w}^i\|^2 = \|\mathbf{x}^j\|^2 - 2 \sum_l \alpha_{il} (\mathbf{x}^j)^t \mathbf{x}^l + \sum_{l,l'} \alpha_{il} \alpha_{il'} (\mathbf{x}^l)^t \mathbf{x}^{l'}$$

This is the same as

$$\begin{aligned} & (D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i \\ &= \sum_l \|\mathbf{x}^j - \mathbf{x}^l\|^2 \cdot \alpha_{il} - 1/2 \cdot \sum_{l,l'} \alpha_{il} \|\mathbf{x}^l - \mathbf{x}^{l'}\|^2 \alpha_{il'} \\ &= \sum_l \|\mathbf{x}^j\|^2 \alpha_{il} - 2 \cdot \sum_l \alpha_{il} (\mathbf{x}^j)^t \mathbf{x}^l + \sum_l \alpha_{il} \|(\mathbf{x}^l)\|^2 \\ & \quad - \sum_{l,l'} \alpha_{il} \alpha_{il'} \|\mathbf{x}^l\|^2 + \sum_{l,l'} \alpha_{il} \alpha_{il'} (\mathbf{x}^l)^t \mathbf{x}^{l'} \end{aligned}$$

because of $\sum_j \alpha_{ij} = 1$. Because of this fact, we can substitute all terms $\|\mathbf{x}^j - \mathbf{w}^i\|^2$ in batch optimization schemes. The parameters α_i yield

1. $\alpha_{ij} = \delta_{i,I(\mathbf{x}^j)} / \sum_j \delta_{i,I(\mathbf{x}^j)}$ for k-means,
2. $\alpha_{ij} = h_\lambda(k_i(\mathbf{x}^j)) / \sum_j h_\lambda(k_i(\mathbf{x}^j))$ for NG

This allows to reformulate the batch optimization in terms of relational data. We obtain

```

init  $\alpha_{ij}$  with  $\sum_j \alpha_{ij} = 1$ 
repeat
  compute the distance  $\|\mathbf{x}^j - \mathbf{w}^i\|^2$  as  $(D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$ 
  compute optimum assignments based on this distance matrix
   $\tilde{\alpha}_{ij} = \delta_{i,I(\mathbf{x}^j)}$  (for k-means) resp.
   $\tilde{\alpha}_{ij} = h_\lambda(k_i(\mathbf{x}^j))$  (for NG)
  compute  $\alpha_{ij} = \tilde{\alpha}_{ij} / \sum_j \tilde{\alpha}_{ij}$  as normalization of these values.
    
```

Hence, prototype locations are computed only indirectly by means of the coefficients α_{ij} . Initialization can be done e.g. setting initial prototype locations to random data points, which is realized by a random selection of k rows from the given distance matrix. Note that prototypes are represented only indirectly by means of the coefficients α_{ij} . For every prototype, m coefficients are stored, m denoting the number of training points. Hence the space complexity of relational clustering is linear w.r.t. the number of training data and the time complexity of one training epoch is quadratic w.r.t. the number of training points.

Given a new data point x which can isometrically be embedded in Euclidean space as \mathbf{x} , and pairwise distances $d_j = d(x, \mathbf{x}^j)^2$ corresponding to the distance from \mathbf{x}^j , the winner can be determined by using the equality

$$\|\mathbf{x} - \mathbf{w}^i\|^2 = (D(x)^t \cdot \alpha_i) - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$$

where $D(x)$ denotes the vector of distances $D(x) = (d_j)_j = (d(x, \mathbf{x}^j)^2)_j$.

The quantization error can be expressed in terms of the given values d_{ij} by substituting $\|\mathbf{x}^j - \mathbf{w}^i\|^2$ by $(D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$. Interestingly, using the formula for optimum assignments of batch optimization, one can also derive relational dual cost functions for the algorithms. For k-means, we use the shorthand notation $\delta_{ij} = \delta_{i,I(\mathbf{x}^j)}$. It holds $\mathbf{w}^i = \sum_j \delta_{ij} \cdot \mathbf{x}^j / \sum_j \delta_{ij}$, hence the cost function becomes

$$\begin{aligned} & 1/2 \cdot \sum_{ij} \delta_{ij} \|\mathbf{w}^i - \mathbf{x}^j\|^2 \\ &= 1/2 \cdot \sum_{ij} \delta_{ij} \|\sum_l \delta_{il} \mathbf{x}^l / \sum_l \delta_{il}\|^2 \\ &= \sum_i 1/(2 \sum_l \delta_{il}) \cdot \left(\sum_{ll'} \delta_{il} \delta_{il'} \|\mathbf{x}^l\|^2 - \sum_{ll'} \delta_{il} \delta_{il'} (\mathbf{x}^l)^t \mathbf{x}^{l'} \right). \end{aligned}$$

Thus, the relational dual of k-means is

$$\sum_i \frac{1}{4 \cdot \sum_l \delta_{iI(\mathbf{x}^l)}} \cdot \sum_{ll'} \delta_{iI(\mathbf{x}^l)} \delta_{iI(\mathbf{x}^{l'})} d_{ll'}.$$

This measures the pairwise distance of data points assigned to the same cluster.

For NG, we use the abbreviation $k_{ij} = h_\lambda(k_i(\mathbf{x}^j))$. Because of $\mathbf{w}^i = \sum_j k_{ij} \cdot \mathbf{x}^j / \sum_j k_{ij}$, we find

$$\begin{aligned} & 1/2 \cdot \sum_{ij} k_{ij} \|\mathbf{x}^j - \mathbf{w}^i\|^2 \\ &= 1/2 \cdot \sum_{ij} k_{ij} \|\mathbf{x}^j - \sum_l k_{il} \cdot \mathbf{x}^l / \sum_l k_{il}\|^2 \\ &= \sum_i 1/(2 \cdot \sum_l k_{il}) \cdot \left(\sum_{ll'} k_{il} k_{il'} \|\mathbf{x}^l\|^2 - \sum_{ll'} k_{il} k_{il'} (\mathbf{x}^l)^t \mathbf{x}^{l'} \right). \end{aligned}$$

Thus, the relational dual of NG is

$$\sum_i \frac{1}{4 \sum_l h_\lambda(k_i(\mathbf{x}^l))} \cdot \sum_{ll'} h_\lambda(k_i(\mathbf{x}^l)) h_\lambda(k_i(\mathbf{x}^{l'})) d_{ll'}.$$

Obviously, this extends the relational dual of k-means towards neighborhood cooperation.

Note that this relational learning gives exactly the same results as standard batch optimization provided the given relations stem from an Euclidean metric. See e.g. [29] for a characterization of this property. Hence, convergence is guaranteed in this case since it holds for the standard batch versions. If the given distance matrix does not stem from an Euclidean metric, this equality does no longer hold and the terms $(D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t \cdot D \cdot \alpha_i$ can become negative. In this case, one can correct the distance matrix by the γ -spread transform $D_\gamma = D + \gamma(\mathbf{1} - \mathbf{I})$ for sufficiently large γ where $\mathbf{1}$ equals 1 for each entry and \mathbf{I} is the identity [12]. For sufficiently large γ , this correction yields a setting where an interpretation of clustering by means of Euclidean prototypes in a possibly high-dimensional Euclidean space exists.

Alternatively, one can apply the formulas for relational clustering directly to any given matrix D , whereby an interpretation by means of explicit prototypes is no longer possible. Interestingly, one can show that this algorithm converges for every symmetric and nonsingular D in a finite number of steps. We exemplarily present the proof for NG: consider the cost function

$$E(k_{ij}, \alpha_{ij}) = \sum_{ij} h_\lambda(k_{ij}) \left(\sum_l d_{jl} \alpha_{il} - \frac{1}{2} \cdot \sum_{ll'} d_{ll'} \alpha_{il} \alpha_{il'} \right)$$

where $\alpha_{ij} \in \mathbb{R}$ and k_{ij} constitutes a permutation of $0, \dots, k-1, k$ denoting the number of prototypes. In relational NG, this cost function is iteratively optimized with respect to α_{ij} for fixed k_{ij} and k_{ij} for fixed α_{ij} . The latter is obvious. The first can be seen as follows: the derivative of $E(k_{ij}, \alpha_{ij})$ with respect to α_{nm} yields

$$\sum_j h_\lambda(k_{nj})d_{jm} - \sum_j h_\lambda(k_{nj}) \sum_l d_{lm}\alpha_{nl} = \sum_j d_{jm} \left(h_\lambda(k_{nj}) - \sum_l h_\lambda(k_{nl})\alpha_{nj} \right)$$

For nonsingular D , this is 0 for all n and m iff $\alpha_{nj} = h_\lambda(k_{nj}) / \sum_l h_\lambda(k_{nl})$, hence relational NG optimizes α_{nj} in the iterative procedure. Assume $\alpha_{ij}(k_{ij})$ are optimum values α_{ij} for fixed k_{ij} . Assume k_{ij} are given. Assume k'_{ij} are computed in the next iteration of relational NG. Then $E(k_{ij}, \alpha_{ij}(k_{ij})) \geq E(k'_{ij}, \alpha_{ij}(k_{ij}))$ because k'_{ij} is chosen optimum with respect to $\alpha_{ij}(k_{ij})$, and $E(k'_{ij}, \alpha_{ij}(k_{ij})) \geq E(k'_{ij}, \alpha_{ij}(k'_{ij}))$ since $\alpha_{ij}(k'_{ij})$ is chosen optimum with respect to k'_{ij} . Hence the cost function decreases in consecutive steps. Since only a finite number of different assignments k_{ij} exists, the algorithm converges (thereby we assume that potential ties for the choice of the ranks k_{ij} are broken deterministically).

Hence relational NG and variants converge for every nonsingular and symmetric matrix D , whereby the cost function $E(k_{ij}, \alpha_{ij})$ is minimized. Note that for optimum values $\alpha_{ij} = h_\lambda(k_{ij}) / \sum_l h_\lambda(k_{il})$ the cost function yields

$$E(k_{ij}, \alpha_{ij}) = \frac{1}{2} \sum_i \frac{1}{\sum_{l''} h_\lambda(k_{il''})} \sum_{l'} h_\lambda(k_{il}) h_\lambda(k_{il'}) d_{ll'} ,$$

i.e. we arrive at the relational dual of NG also when using this procedure for general (symmetric and nonsingular) D .

3.2 Dot Products

A dual possibility is to characterize data x^1, \dots, x^m by means of pairwise similarities, i.e. dot products. We denote the similarity of x^i and x^j by $k(x^i, x^j) = k_{ij}$. We assume that these values fulfill the properties of a dot product, i.e. the matrix K with entries k_{ij} is positive definite. In this case, a representation \mathbf{x}^i of the data can be found in a possibly high dimensional Euclidean vector space such that $k_{ij} = (\mathbf{x}^i)^t \mathbf{x}^j$.

As beforehand, we can represent distances in terms of these values if $\mathbf{w}^i = \sum_l \alpha_{il} \mathbf{x}^l$ with $\sum_l \alpha_{il} = 1$ yields optimum prototypes:

$$\|\mathbf{x}^j - \mathbf{w}^i\|^2 = k_{jj} - 2 \sum_l \alpha_{il} k_{jl} + \sum_{l'} \alpha_{il} \alpha_{il'} k_{ll'} .$$

This allows to compute batch optimization in the same way as beforehand:

init α_{ij} with $\sum_j \alpha_{ij} = 1$

repeat

compute the distance $\|\mathbf{x}^j - \mathbf{w}^i\|^2$ as $k_{jj} - 2 \sum_l \alpha_{il} k_{jl} + \sum_{l'} \alpha_{il} \alpha_{il'} k_{ll'}$

compute optimum assignments based on this distance matrix

$\tilde{\alpha}_{ij} = \delta_{i,I(\mathbf{x}^j)}$ (for k-means) resp.

$\tilde{\alpha}_{ij} = h_\lambda(k_i(\mathbf{x}^j))$ (for NG)

compute $\alpha_{ij} = \tilde{\alpha}_{ij} / \sum_j \tilde{\alpha}_{ij}$ as normalization of these values.

One can use the same identity for $\|\mathbf{x} - \mathbf{w}^i\|^2$ to obtain a possibility to compute the winner given a point x and to compute the respective cost function. Convergence of this algorithm is guaranteed since it is identical to the batch versions for the Euclidean data embedding \mathbf{x}^i if K is positive definite.

If K is not positive definite negative values can occur for $\|\mathbf{x}^j - \mathbf{w}^i\|^2$. Then the kernel matrix can be corrected by $K_\gamma = K + \gamma \cdot \mathbf{1}$ with large enough γ .

4 Supervision

The possibility to include further information, if available, is very important to get meaningful results for unsupervised learning. This can help to prevent the ‘garbage in - garbage out’ problem of unsupervised learning, as discussed e.g. in [16,17]. Here we assume that additional label information is available which should be accounted for by clustering or visualization. Thereby, labels are embedded in \mathbb{R}^d and can be fuzzy. We assume that the label attached to x^j is denoted by \mathbf{y}^j . We equip a prototype w^i with a label $\mathbf{Y}^i \in \mathbb{R}^d$ which is adapted during learning. For the Euclidean case, the basic idea consists in a substitution of the standard Euclidean distance $\|\mathbf{x}^j - \mathbf{w}^i\|^2$ by a mixture

$$(1 - \beta) \cdot \|\mathbf{x}^j - \mathbf{w}^i\|^2 + \beta \cdot \|\mathbf{y}^j - \mathbf{Y}^i\|^2$$

which takes the similarity of label assignments into account and where $\beta \in [0, 1]$ controls the influence of the label values. This procedure has been proposed in [6,7,32] for Euclidean and median clustering and online neural gas, respectively. One can use the same principles to extend relational clustering.

For discrete Euclidean settings $\mathbf{x}^1, \dots, \mathbf{x}^m$ cost functions and related batch optimization is as follows (neglecting constant factors):

$$E_{k\text{-means}}(\mathbf{w}, \mathbf{Y}) = \sum_{ij} \delta_{i, I(\mathbf{x}^j)} \cdot ((1 - \beta) \cdot \|\mathbf{x}^j - \mathbf{w}^i\|^2 + \beta \cdot \|\mathbf{y}^j - \mathbf{Y}^i\|^2)$$

where $\delta_{i, I(\mathbf{x}^j)}$ indicates the winner for \mathbf{x}^j which is the neuron $\mathbf{w}^{I(\mathbf{x}^j)}$ with smallest $(1 - \beta) \cdot \|\mathbf{x}^j - \mathbf{w}^{I(\mathbf{x}^j)}\|^2 + \beta \cdot \|\mathbf{y}^j - \mathbf{Y}^{I(\mathbf{x}^j)}\|^2$. Besides this slight change in the winner notation, the batch update is extended by the adaptation step $\mathbf{Y}^i = \sum_j \delta_{i, I(\mathbf{x}^j)} \mathbf{y}^j / \sum_j \delta_{i, I(\mathbf{x}^j)}$ for the prototype labels.

Similarly, the cost function of NG becomes

$$E_{\text{NG}}(\mathbf{w}, \mathbf{Y}) = \sum_{ij} h_\lambda(k_i(\mathbf{x}^j)) \cdot ((1 - \beta) \cdot \|\mathbf{x}^j - \mathbf{w}^i\|^2 + \beta \cdot \|\mathbf{y}^j - \mathbf{Y}^i\|^2)$$

where $k_i(\mathbf{x}^j)$ denotes the rank of neuron i measured according to the distances $(1 - \beta) \cdot \|\mathbf{x}^j - \mathbf{w}^i\|^2 + \beta \cdot \|\mathbf{y}^j - \mathbf{Y}^i\|^2$. Again, this change in the computation of the rank is accompanied by the adaptation $\mathbf{Y}^i = \sum_j h_\lambda(\mathbf{x}^j) \mathbf{y}^j / \sum_j h_\lambda(\mathbf{x}^j)$ for the prototype labels for batch optimization

For these generalized cost functions, relational learning becomes possible by substituting the distances $\|\mathbf{x}^j - \mathbf{w}^i\|^2$ using the identity $\mathbf{w}^i = \sum \alpha_{ij} \mathbf{x}^j$ for optimum assignments which still holds for these extensions. The same computation as beforehand yields to the algorithm for clustering dissimilarity data characterized by pairwise distances d_{ij} :

init α_{ij} with $\sum_j \alpha_{ij} = 1$

repeat

compute the distances as $(1 - \beta) \cdot ((D \cdot \alpha_i)_j - 1/2 \cdot \alpha_i^t D \alpha_i) + \beta \cdot \|Y^i - y^j\|^2$

compute optimum assignments $\tilde{\alpha}_{ij}$ based on this distance as before

compute $\alpha_{ij} = \tilde{\alpha}_{ij} / \sum_j \tilde{\alpha}_{ij}$

compute prototype labels $Y^i = \sum_j \alpha_{ij} y^j$

An extension to similarity data given by dot products $\mathbf{x}^i \cdot \mathbf{x}^j$ proceeds in the same way using the distance computation based on dot products as derived beforehand. As beforehand, this version converges in a finite number of steps.

5 Experiments

In the experiments, we focus on the clustering and classification ability of the algorithms rather than the visualization, since these aspects can easily be evaluated by the classification error for given data labels. We demonstrate the performance of the neural gas and k-means algorithms in different scenarios covering a variety of characteristic situations. All algorithms have been implemented based on the SOM Toolbox for Matlab [26]. Note that, for all median versions, prototypes situated at identical points of the data space do not separate in subsequent runs. Therefore constellations with exactly identical prototypes should be avoided. For the Euclidean and relational versions this problem is negligible, presumed prototypes are initialized at different positions. However, for median versions it is likely that prototypes move to an identical locations due to the limited number of different positions in data space, in particular for small data sets. To cope with this fact in median versions, we add a small amount of noise to the distances in each epoch in order to separate identical prototypes. The initial neighborhood rate for neural gas is $\lambda = n/2$, n being the number of neurons, and it is multiplicatively decreased during training. In all runs, relational clustering has been applied directly without any correction of the given matrix.

Wisconsin Breast Cancer Database

The Wisconsin Diagnostic Breast Cancer database (WDBC) is a standard benchmark set from clinical proteomics [33]. It consists of 569 data points described by 30 real-valued input features: digitized images of a fine needle aspirate of breast mass are described by characteristics such as form and texture of the cell nuclei present in the image. Data are labeled by two classes, benign and malignant.

For training we used 40 neurons and 150 epochs per run. The dataset was z-transformed beforehand. The results were gained from repeated 2-fold cross-validations averaged over 100 runs. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 1. Moreover, the data set is contained in the Euclidean space therefore we are able to compare the relational versions introduced in this article to the standard Euclidean methods. These results are shown in Table 1. The effect of a variation of the mixing parameter is demonstrated in Fig. 1. The results are competitive to supervised learning with the state-of-the-art-method GRLVQ as obtained in [28].

Table 1. Classification accuracy on the WDBC database for posterior labeling. The mean accuracy over 100 repeats of 2-fold cross-validation is reported.

| | k-Means | Supervised k-Means | Median k-Means | Relational k-Means | Supervised Relational k-Means |
|----------|----------|---------------------|-----------------|---------------------|--------------------------------|
| Accuracy | | | | | |
| Mean | 93.6 | 93.0 | 93.0 | 93.4 | 93.5 |
| StdDev | 0.8 | 1.1 | 1.0 | 1.2 | 1.1 |
| | Batch NG | Supervised Batch NG | Median Batch NG | Relational Batch NG | Supervised Relational Batch NG |
| Accuracy | | | | | |
| Mean | 94.1 | 94.7 | 93.1 | 94.0 | 94.4 |
| StdDev | 1.0 | 0.8 | 1.0 | 0.9 | 1.0 |

As one can see, the results of Euclidean and relational clustering are identical, as expected by the theoretical background of relational clustering. Relational clustering and supervision allow to improve the more restricted and unsupervised median versions by more than 1% classification accuracy.

Cat Cortex

The Cat Cortex Data Set originates from anatomic studies of cats' brains. A matrix of connection strengths between 65 cortical areas of cats was compiled from literature [4]. There are four classes corresponding to four different regions of the cortex. For our experiments a preprocessed version of the data set from Haasdonk et al. [11] was used. The matrix is symmetric but the triangle inequality does not hold. Nevertheless, relational NG converges as shown beforehand because of a symmetric nonsingular distance matrix.

The algorithms were tested in 10-fold cross-validation using 12 neurons (three per class) and 150 epochs per run. The results presented reveal the mean accuracy over 250 repeated 10-fold cross-validations per method. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 2. Results for different mixing parameters are shown in Figure 2.

Table 2. Classification accuracy on the Cat Cortex Data Set for posterior labeling. The mean accuracy over 250 repeats of 10-fold cross-validation is reported.

| | Median k-Means | Median Batch NG | Relational k-Means | Relational Batch NG | Supervised Median Batch NG | Supervised Relational k-Means | Supervised Relational Batch NG |
|----------|----------------|-----------------|--------------------|---------------------|----------------------------|-------------------------------|--------------------------------|
| Accuracy | | | | | | | |
| Mean | 72.8 | 71.6 | 89.0 | 88.7 | 77.9 | 89.2 | 91.3 |
| StdDev | 3.9 | 4.0 | 3.3 | 3.0 | 3.5 | 3.0 | 2.8 |

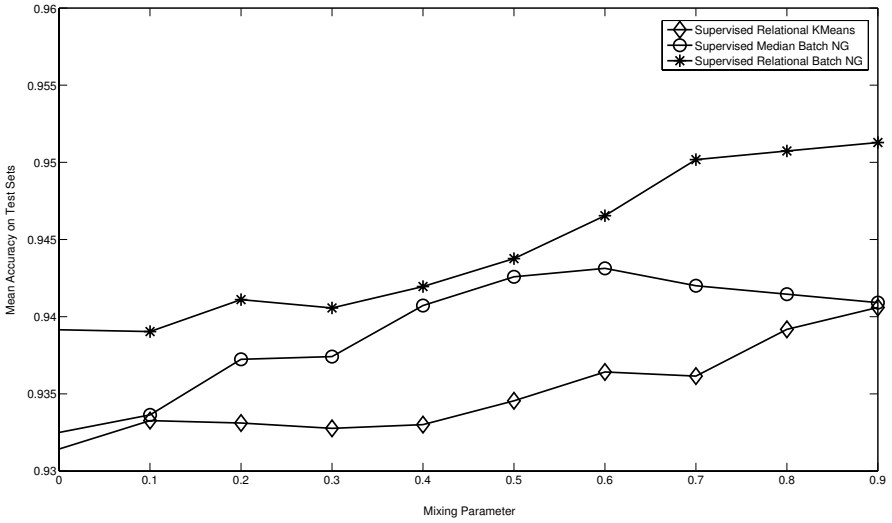


Fig. 1. Results of the supervised methods for the WDBC data set with different mixing parameters applied

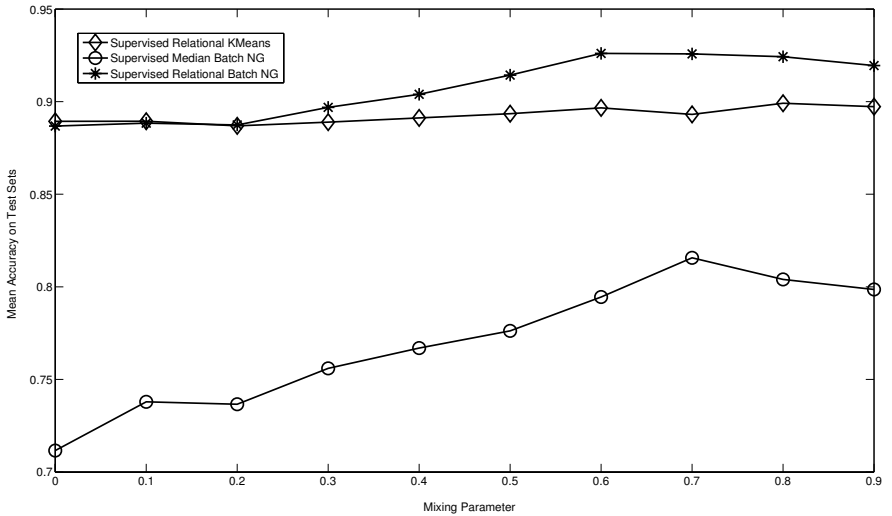


Fig. 2. Results of the supervised methods for the Cat Cortex Data Set with different mixing parameters applied

A direct comparison of our results to the findings of Graepel et al. [4] or Haasdonk et al. [11] is not possible. Haasdonk et al. gained an accumulated error over all classes of at least 10% in leave-one-out experiments with SVMs. Graepel et al. obtained virtually the same results with the Optimal Hyperplane

Table 3. Classification accuracy on the Protein Data Set for posterior labeling. The mean accuracy over 100 repeats of 10-fold cross-validation is reported.

| | Median k-Means | Median Batch NG | Relational k-Means | Relational Batch NG | Supervised Median Batch | Supervised Relational k-Means | Supervised Relational Batch |
|-----------------|-------------------|--------------------|-----------------------|---------------------------|-------------------------------|-------------------------------------|-----------------------------------|
| Accuracy | | | | | | | |
| Mean | 76.1 | 76.3 | 88.0 | 89.9 | 89.4 | 88.2 | 90.0 |
| StdDev | 1.3 | 1.8 | 1.8 | 1.3 | 1.4 | 1.7 | 1.0 |

(OHC) algorithm. In our experiments, the improvement of restricted median clustering by relational extensions by more than 10% classification accuracy can clearly be observed. Note that relational clustering works quite well in this case although an interpretation by means of prototypes is not directly possible.

Proteins

The evolutionary distance of 226 globin proteins is determined by alignment as described in [24]. These samples originate from different protein families: hemoglobin- α , hemoglobin- β , myoglobin, etc. Here, we distinguish five classes as proposed in [11]: HA, HB, MY, GG/GP, and others.

For training we used 45 neurons and 150 epochs per run. The results were gained from repeated 10-fold cross-validations averaged over 100 runs. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 3.

Unlike the results reported in [11] for SVM which uses one-versus-rest encoding, the classification in our setting is given by only one clustering model. Depending on the choice of the kernel, [11] reports errors which approximately add up to 4% for the leave-one-out error. This result, however, is not comparable to our results due to the different error measure. A 1-nearest neighbor classifier yields an accuracy 91.6 for our setting (k-nearest neighbor for larger k is worse; [11] which is comparable to our results.

Chromosomes

The Copenhagen chromosomes database is a benchmark from cytogenetics [19]. A set of 4200 human nuclear chromosomes from 22 classes (the X resp. Y sex chromosome is not considered) are represented by the grey levels of their images and transferred to strings representing the profile of the chromosome by the thickness of their silhouettes. Thus, this data set consists of strings of different length, and standard k-means clustering cannot be used. Median versions, however, are directly applicable. The edit distance is a typical distance measure for two strings of different length, as described in [15,25]. In our application, distances of two strings are computed using the standard edit distance whereby substitution costs are given by the signed difference of the entries and insertion/deletion costs are given by 4.5 [25].

Table 4. Classification accuracy on the Copenhagen Chromosome Database for posterior labeling. The mean accuracy over 10 runs of 2-fold cross-validation is reported.

| | Median k-Means | Median Batch NG | Relational k-Means | Relational Batch NG | Supervised Median Batch | Supervised Relational k-Means | Supervised Relational Batch |
|-----------------|-------------------|-----------------------|-----------------------|---------------------------|-------------------------------|-------------------------------------|-----------------------------------|
| Accuracy | | | | | | | |
| Mean | 82.3 | 82.8 | 90.6 | 91.3 | 89.4 | 90.1 | 91.4 |
| StdDev | 2.2 | 1.7 | 0.6 | 0.2 | 0.6 | 0.6 | 0.6 |

The algorithms were tested in 2-fold cross-validation using 100 neurons and 100 epochs per run (cf. [3]). The results presented are the mean accuracy over 10 times 2-fold cross-validation per method. The mixing parameter of the supervised methods was set to 0.9.

As can be seen, supervised relational neural gas achieves an accuracy of 0.914 for $\alpha = 0.9$. This improves by 8% compared to median variants.

6 Discussion

We have introduced relational neural clustering which extends the classical Euclidean versions to settings where pairwise distances or dot products of the data are given but no explicit embedding into a Euclidean space is known. By means of the relational dual, batch optimization can be formulated in terms of these quantities only. This extends previous median clustering variants to a continuous prototype update which is particularly useful for only sparsely sampled data. The derived relational algorithms have a formal background only for Euclidean distances or metrics; however, as demonstrated in an example for the cat cortex data, the algorithms might also prove useful in more general scenarios, and convergence is guaranteed for fairly general settings. In all experiments presented in this contribution, relational clustering significantly improves the classification accuracy obtained by semi-supervised clustering compared to median clustering using the same underlying cost function. Depending on the data set at hand, results which are competitive to state-of-the-art classification (using dedicated supervised training) could be approximated in our settings, demonstrating the efficiency and robustness of relational clustering. However, being based on the quantization error and related quantities, relational clustering is mainly intended for data inspection whereby additional information can be integrated to achieve meaningful clusters. The general framework as introduced in this article opens the way towards the transfer of further principles of SOM and NG to the setting of relational data: as an example, the magnification factor of topographic map formation for relational data transfers from the Euclidean space, and possibilities to control this factor as demonstrated for batch clustering e.g. in the approach [9] can readily be used.

One very important subject of future work concerns the complexity of computation and sparseness of prototype representation. For the approach as introduced above, the complexity scales quadratic with the number of training

examples and the size of prototype representations is linear with respect to the number of examples. The representation contains a large number of very small coefficients, which correspond to data points for which the distance from the prototype is large. Therefore it can be expected that a restriction of the representation to the close neighborhood is sufficient for accurate results.

References

1. Anderson, J.W.: *Hyperbolic Geometry*, 2nd edn. Springer, Heidelberg (2005)
2. Conan-Guez, B., Rossi, F., El Golli, A.: A fast algorithm for the self-organizing map on dissimilarity data. In: *Workshop on Self-Organizing Maps*, pp. 561–568 (2005)
3. Cottrell, M., Hammer, B., Hasenfuss, A., Villmann, T.: Batch and median neural gas. *Neural Networks* 19, 762–771 (2006)
4. Graepel, T., Herbrich, R., Bollmann-Sdorra, P., Obermayer, K.: Classification on pairwise proximity data. In: Jordan, M.I., Kearns, M.J., Solla, S.A. (eds.) *NIPS*, vol. 11, pp. 438–444. MIT Press, Cambridge (1999)
5. Graepel, T., Obermayer, K.: A stochastic self-organizing map for proximity data. *Neural Computation* 11, 139–155 (1999)
6. Hammer, B., Hasenfuss, A., Schleif, F.-M., Villmann, T.: Supervised median neural gas. In: Dagli, C., Buczak, A., Enke, D., Embrechts, A., Ersoy, O. (eds.) *Intelligent Engineering Systems Through Artificial Neural Networks. Smart Engineering System Design*, vol. 16, pp. 623–633. ASME Press (2006)
7. Hammer, B., Hasenfuss, A., Schleif, F.-M., Villmann, T.: Supervised batch neural gas. In: Schwenker, F., Marinai, S. (eds.) *ANNPR 2006. LNCS (LNAI)*, vol. 4087, pp. 33–45. Springer, Heidelberg (2006)
8. Hasenfuss, A., Hammer, B., Schleif, F.-M., Villmann, T.: Neural gas clustering for dissimilarity data with continuous prototypes. In: *WANN'07* (accepted, 2007)
9. Hammer, B., Hasenfuss, A., Villmann, T.: Magnification control for batch neural gas. *Neurocomputing* 70, 1225–1234 (2007)
10. Hammer, B., Micheli, A., Sperduti, A., Strickert, M.: Recursive self-organizing network models. *Neural Networks* 17(8-9), 1061–1086 (2004)
11. Haasdonk, B., Bahlmann, C.: Learning with distance substitution kernels. In: Rasmussen, C.E., Bühlhoff, H.H., Schölkopf, B., Giese, M.A. (eds.) *Pattern Recognition. LNCS*, vol. 3175, Springer, Heidelberg (2004)
12. Hathaway, R.J., Bezdek, J.C.: Nerf c-means: Non-euclidean relational fuzzy clustering. *Pattern Recognition* 27(3), 429–437 (1994)
13. Hathaway, R.J., Davenport, J.W., Bezdek, J.C.: Relational duals of the c-means algorithms. *Pattern Recognition* 22, 205–212 (1989)
14. Heskes, T.: Self-organizing maps, vector quantization, and mixture modeling. *IEEE Transactions on Neural Networks* 12, 1299–1305 (2001)
15. Juan, A., Vidal, E.: On the use of normalized edit distances and an efficient k-NN search technique (k-AESA) for fast and accurate string classification. In: *ICPR 2000*, vol. 2, pp. 680–683 (2000)
16. Kaski, S., Nikkilä, J., Savia, E., Roos, C.: Discriminative clustering of yeast stress response. In: Seiffert, U., Jain, L., Schweizer, P. (eds.) *Bioinformatics using Computational Intelligence Paradigms*, pp. 75–92. Springer, Heidelberg (2005)
17. Kaski, S., Nikkilä, J., Oja, M., Venna, J., Törönen, P., Castren, E.: Trustworthiness and metrics in visualizing similarity of gene expression. *BMC Bioinformatics* 4, 48 (2003)
18. Kohonen, T.: *Self-Organizing Maps*. Springer, Heidelberg (1995)

19. Lundsteen, C., Phillip, J., Granum, E.: Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes. *Clinical Genetics* 18, 355–370 (1980)
20. Kohonen, T., Somervuo, P.: How to make large self-organizing maps for nonvectorial data. *Neural Networks* 15, 945–952 (2002)
21. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis. *Psychometrika* 29, 1–27 (1964)
22. Martinetz, T., Berkovich, S.G., Schulten, K.J.: Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks* 4, 558–569 (1993)
23. Martinetz, T., Schulten, K.: Topology representing networks. *Neural Networks* 7, 507–522 (1994)
24. Mevissen, H., Vingron, M.: Quantifying the local reliability of a sequence alignment. *Protein Engineering* 9, 127–132 (1996)
25. Neuhaus, M., Bunke, H.: Edit distance based kernel functions for structural pattern classification. *Pattern Recognition* 39(10), 1852–1863 (2006)
26. Neural Networks Research Centre, Helsinki University of Technology, SOM Toolbox, <http://www.cis.hut.fi/projects/somtoolbox/>
27. Qin, A.K., Suganthan, P.N.: Kernel neural gas algorithms with application to cluster analysis. *ICPR 2004* 4, 617–620 (2004)
28. Schleif, F.-M., Hammer, B., Villmann, T.: Margin based Active Learning for LVQ Networks. *Neurocomputing* 70(7-9), 1215–1224 (2007)
29. Schölkopf, B.: The kernel trick for distances, Microsoft TR 2000-51 (2000)
30. Seo, S., Obermayer, K.: Self-organizing maps and clustering methods for matrix data. *Neural Networks* 17, 1211–1230 (2004)
31. Tino, P., Kaban, A., Sun, Y.: A generative probabilistic approach to visualizing sets of symbolic sequences. In: Kohavi, R., Gehrke, J., DuMouchel, W., Ghosh, J. (eds.) *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD-2004*, pp. 701–706. ACM Press, New York (2004)
32. Villmann, T., Hammer, B., Schleif, F., Geweniger, T., Herrmann, W.: Fuzzy classification by fuzzy labeled neural gas. *Neural Networks* 19, 772–779 (2006)
33. Wolberg, W.H., Street, W.N., Heisey, D.M., Mangasarian, O.L.: Computer-derived nuclear features distinguish malignant from benign breast cytology. *Human Pathology* 26, 792–796 (1995)
34. Yin, H.: On the equivalence between kernel self-organising maps and self-organising mixture density network. *Neural Networks* 19(6), 780–784 (2006)

A General Framework for Encoding and Evolving Neural Networks

Yohannes Kassahun¹, Jan Hendrik Metzen¹, Jose de Gea¹, Mark Edgington¹,
and Frank Kirchner^{1,2}

¹ Robotics Group, University of Bremen

Robert-Hooke-Str. 5, D-28359, Bremen, Germany

² German Research Center for Artificial Intelligence (DFKI)

Robert-Hooke-Str. 5, D-28359, Bremen, Germany

Abstract. In this paper we present a novel general framework for encoding and evolving networks called Common Genetic Encoding (CGE) that can be applied to both direct and indirect encoding methods. The encoding has important properties that makes it suitable for evolving neural networks: (1) It is *complete* in that it is able to represent all types of valid phenotype networks. (2) It is *closed*, i.e. every valid genotype represents a valid phenotype. Similarly, the encoding is *closed under genetic operators* such as structural mutation and crossover that act upon the genotype. Moreover, the encoding's genotype can be seen as a composition of several subgenomes, which makes it to inherently support the evolution of modular networks in both direct and indirect encoding cases. To demonstrate our encoding, we present an experiment where direct encoding is used to learn the dynamic model of a two-link arm robot. We also provide an illustration of how the indirect-encoding features of CGE can be used in the area of artificial embryogeny.

1 Introduction

A meaningful combination of the principles of neural networks and evolutionary computation is useful for designing agents that learn and adapt to their environment through interaction. One step towards achieving such a combination involves the design of a flexible genetic encoding that is suitable for evolving networks using both direct and indirect encoding methods. To our knowledge, CGE is the first genetic encoding that tries to consider both direct and indirect encoding of networks under the same theoretical framework. In addition to supporting both types of genetic encodings, CGE has some important properties that makes it suitable for encoding and evolving neural networks.

The paper is organized as follows: First, a detailed review of work in the area of Evolution of Artificial Neural Networks (EANNs) is given. Next, a description of CGE is provided. We then present an experiment in learning the dynamic model of a two-link arm robot, and illustrate how CGE can be used for artificial embryogeny. After this, a comparison of CGE to other genetic encodings is made. Finally, some conclusions and a future outlook is provided.

2 Review of Work in Evolution of Artificial Neural Networks

The field of EANNs can be divided into two major areas of research: the evolution of connection weights, and the evolution of both structure and connection weights. In the first area, the structure of neural networks is fixed before the evolution begins. In the second area, both the structure and the connection weights are determined automatically during the evolutionary process. Since the evolution of connection weights is not interesting in the context of this paper, we will give only a review to relevant work in the second area. For a detailed review of the work in the evolution of neural networks see Yao [19].

Angeline et al. developed a system called GNARL (GeNeralized Acquisition of Recurrent Links) which uses only structural mutation of the topology, and parametric mutations of the weights as genetic search operators [1]. The main problem with this method is that genomes may end up in many extraneous disconnected structures that have no contribution to the solution. The Neuroevolution of Augmenting Topologies (NEAT) [17] evolves both the structure and weights of neural networks. It starts with networks of minimal structures and increases their complexity along the evolution path. The algorithm keeps track of the historical origin of every gene that is introduced through structural mutation. This history is used by a specially designed crossover operator to match genomes which encode different network topologies. Unlike GNARL, NEAT does not use self-adaptation of mutation step-sizes. Instead, each connection weight is perturbed with a fixed probability by adding a floating point number chosen from a uniform distribution of positive and negative values.

Kitano's grammar based encoding of neural networks uses Lindenmayer systems (L-systems) [12] to describe the morphogenesis of linear and branching structures in plants [10]. Sendhoff et al. extended Kitano's grammar encoding with a recursive encoding of modular neural networks [16]. Their system provides a means of initializing the network weights, whereas in Kitano's grammar based encoding, there is no direct way of representing the connection weights of neural networks in the genome. Gruau's Cellular Encoding (CE) method is a language for local graph transformations that controls the division of cells which grow into an artificial neural network [5]. The genetic representations in CE are compact because genes can be reused several times during the development of the network and this saves space in the genome since not every connection and node needs to be explicitly specified in the genome. Defining a crossover operator for CE is still difficult, and it is not easy to analyze how crossover affects the subfunctions in CE encoding since they are not explicitly represented. Vaario et al. have developed a biologically inspired neural growth based on diffusion field modeling combined with genetic factors for controlling the growth of the network [18]. One weak point of this method is that it cannot generate networks with recurrent connections or networks with connections between neurons on different branches of the resulting tree structure. Nolfi and Parisi have modelled biological development at the chemical level using a reaction-diffusion model [14]. This method utilizes growth to create connectivity without explicitly describing each

connection in the phenotype. The complexity of a structure that the genome can represent is limited since every neuron is directly specified in the genome. Other work in indirect encoding have borrowed ideas from systems biology, and simulated Genetic Regulatory Networks (GRNs), in which genes produce signals that either activate or inhibit other genes in the genome. Typical works using GRNs include those of Dellaert and Beer [4], Jakobi [7], Bongard and Pfeifer [3], and Bentley and Kumar [2].

3 Common Genetic Encoding (CGE)

A genotype in CGE is a *sequence of genes* that can take one of three different forms: a *vertex gene*, an *input gene*, or a *jumper gene*. A vertex gene encodes a vertex of a network, an input gene encodes an input to the network, and a jumper gene encodes a connection between two vertices. A particular jumper gene can either be a forward or a recurrent jumper gene. A forward jumper gene represents a connection starting from a vertex gene with higher depth¹ and ending at a vertex with lower or same depth. A recurrent jumper gene represents a connection between two vertices with arbitrary depths. Depending on whether the encoding is interpreted directly or indirectly, the vertex genes can store different information such as weights $w_i \in \mathbb{R}$ (e.g. when the encoded network is interpreted directly as a neural network) or operator type (e.g. when the encoded network is indirectly mapped to a phenotype network).

A genotype $g = [x_1, \dots, x_N] \in \mathcal{G}$ is defined as a sequence of genes $x_i \in \mathcal{X}$, where \mathcal{G} is the set of all valid genotypes, and $\mathcal{X} = \mathcal{V} \cup \mathcal{I} \cup \mathcal{J}_F \cup \mathcal{J}_R$. \mathcal{V} is a set of vertex genes, \mathcal{I} is a set of input genes, and \mathcal{J}_F and \mathcal{J}_R are sets of forward and recurrent jumper genes, respectively. For a gene x and a genotype $g = [x_1, \dots, x_N]$ we say $x \in g$ iff $\exists 0 < i \leq N : x = x_i$. To each vertex gene there is an associated unique identity number $id \in \mathbb{N}_0$ and to each input gene there is an associated label, where input genes with the same label refer to the same input. The set of identity numbers and the set of labels are disjoint. Each vertex gene x_i stores a value $d_{in}(x_i)$, which can be interpreted as the number of expected inputs (i.e., the number of arguments of x_i). A forward or a recurrent jumper gene stores the identity number of its source vertex gene. Two genes $x_i \in g_1$ and $x_j \in g_2$ are considered to be equal if the following condition is satisfied:

$$x_i = x_j \Leftrightarrow \begin{cases} (x_i \in \mathcal{V} \wedge x_j \in \mathcal{V} \wedge x_i.id = x_j.id) \\ \vee (x_i \in \mathcal{I} \wedge x_j \in \mathcal{I} \wedge x_i.label = x_j.label) \\ \vee (x_i \in \mathcal{J}_F \wedge x_j \in \mathcal{J}_F \wedge x_i.source_id = x_j.source_id) \\ \vee (x_i \in \mathcal{J}_R \wedge x_j \in \mathcal{J}_R \wedge x_i.source_id = x_j.source_id) \end{cases} \quad (1)$$

There are different functions defined on the genes of a genotype that can be used for determining properties of the genotypes during the evolutionary run. The first function $v : \mathcal{X} \rightarrow \mathbb{Z}$ defined as

$$v(x_i) = \begin{cases} 1 - d_{in}(x_i), & \text{if } x_i \in \mathcal{V} \\ 1, & \text{if } x_i \notin \mathcal{V} \end{cases} \quad (2)$$

¹ For a formal definition of a gene's depth, see Equation 6.

can be interpreted as the number of implicitly produced outputs (which is always 1) minus the number of expected inputs by the gene x_i . This function allows us to define the sum

$$s_K = \sum_{i=1}^{K-1} v(x_i), \quad (3)$$

where $K \in \{1, \dots, N + 1\}$. Note that this definition implies $s_1 = 0$. Based on this, we define the set of *output vertex genes* as

$$\mathcal{V}_o = \{x_j \in g \mid x_j \in \mathcal{V} \wedge (s_i < s_j \forall i : 0 < i < j)\} \quad (4)$$

and the set of *non-output vertex genes* as $\mathcal{V}_{no} = \mathcal{V} - \mathcal{V}_o$.

We consider a subsequence $g_{l,m} = [x_l, x_{l+1}, \dots, x_{l+m-1}]$ of g to be a *subgenome* of a genotype g if $x_l \in \mathcal{V}$ and $s_{l,m} = \sum_{i=l}^{l+m-1} v(x_i) = 1$. Subgenomes are an important concept in CGE, because they make it possible to treat developed phenotype structures as a composition of phenotype substructures that correspond to the subgenomes, and because of this, they allow the genetic encoding to *inherently* support the evolution of modular neural networks.

We can define a hierarchy-relationship between the genes in a genotype by the function $parent : \mathcal{X} \rightarrow \mathcal{V} \cup \emptyset$

$$parent(x_j) = \begin{cases} \emptyset, & \text{if } (s_i < s_j \forall i : 0 < i < j) \\ x_i, & \text{if } s_i \geq s_j \text{ and } s_k < s_j \forall k : 0 < i < k < j. \end{cases} \quad (5)$$

From equations (4) and (5), it follows that for an output vertex gene x_j , $parent(x_j) = \emptyset$. The output of a gene x_j acts implicitly as an input for $parent(x_j)$. The depth of a vertex gene is defined as the minimal topological distance (i.e. minimal number of connections to be traversed) from an output vertex of the network to the vertex itself, where the path contains only implicit connections. This is defined mathematically by the function $depth : \mathcal{V} \rightarrow \mathbb{N}$

$$depth(x_j) = \begin{cases} 0 & \text{if } parent(x_j) = \emptyset \\ depth(parent(x_j)) + 1, & \text{otherwise} \end{cases}. \quad (6)$$

Table 1 shows an example of a genotype encoding the neural network shown in Figure 1 along with the resulting values of the above-defined functions.

We consider two genotypes g_1 and g_2 to be *equivalent* if and only if there is a one-to-one correspondence between them, i.e. $\forall x_i \in g_1 \exists x_j \in g_2 : x_i = x_j \wedge parent(x_i) = parent(x_j)$, and $\forall x_j \in g_2 \exists x_i \in g_1 : x_i = x_j \wedge parent(x_i) = parent(x_j)$. The equivalence criterion between two genotypes can be used to lessen the competing convention problem [15] that is encountered during the evolution of neural networks. A newly generated genotype is tested against all existing genotypes before it is added to the population. If there is an already existing equivalent genotype, the newly generated genotype will not be added to the population.

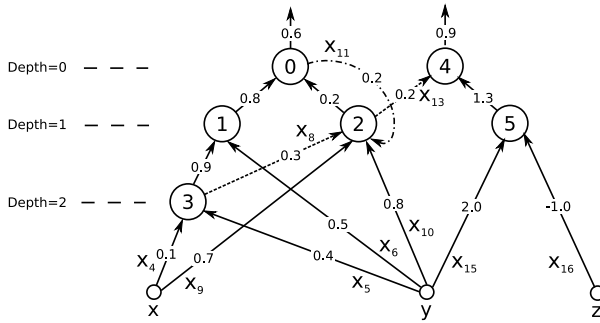


Fig. 1. An example of a valid phenotype with two output vertices (0 and 4), and three input vertices (x, y and z)

The following five *criteria* must be fulfilled for a genotype $g = [x_1, \dots, x_N]$ to be considered a valid genotype, i. e. $g \in \mathcal{G}$:

1. Each vertex gene $x_i \in \mathcal{V}$ must have at least one input: $d_{in}(x_i) > 0$.
2. There can be no closed loops of forward jumper connection genes in g .
3. There is no forward jumper gene whose source vertex *depth* is less than the *depth* of its target vertex.
4. For a gene $x_k \in g$, $s_k < s_{N+1}, \forall k \in \{1, \dots, N\}$.
5. For every $x_k \in g$: $parent(x_k) = \emptyset \Rightarrow x_k \in \mathcal{V}$.

A vertex gene x_i with $d_{in}(x_i) = 0$ has no input and would always yield the same result. Because of this, such a vertex is not allowed (criterion 1). The second and third criteria together guarantee that the evaluation of a phenotype in the direct encoding case or the development process of a phenotype in the indirect encoding case can be completed in a finite amount of time (i. e. there are no infinite loops). The last two criteria together ensure that the sum of outputs produced by all genes in g minus the sum of all expected inputs is equal to the number of outputs of the corresponding phenotype network. We denote the set of phenotypes represented by CGE genotypes with \mathcal{P}_{CGE} . The *development function* $\mathcal{D} : \mathcal{G} \rightarrow \mathcal{P}_{CGE}$ formalizes a process that creates for every valid genotype $g = [x_1, \dots, x_N] \in \mathcal{G}$ a corresponding phenotype $p \in \mathcal{P}_{CGE}$.

We have designed three kinds of *genetic operators* for the use in CGE: parametric mutation, structural mutation and structural crossover. The genetic operators to be used in CGE are designed so that the resulting genotypes they produce fulfill the 5 criteria stated above. A *parametric mutation* $\mathcal{PA} : \mathcal{G} \rightarrow \mathcal{G}$ changes only the values of the parameters included in the genes (e.g. the weights w_i). The order of the genes in g and $\mathcal{PA}(g)$ remains the same. An example of a *structural mutation* operator $\mathcal{ST} : \mathcal{G} \rightarrow \mathcal{G}$ that fulfills the above criteria is defined as follows: when \mathcal{ST} operates on a genotype, it either inserts a recurrent jumper gene, or a subgenome after a vertex gene x_i , and the number of inputs $d_{in}(x_i)$ will be increased by one. The source vertex of a recurrent jumper can be chosen arbitrarily. The subgenome consists of a vertex gene x_k followed by

Table 1. The phenotype in Figure 1 is encoded by the genotype shown in this table. For each gene x_i of the genotype, the gene’s defined properties and the values of various functions which operate on the gene are summarized. In the allele row, V denotes a vertex gene, I an input gene, JF a forward jumper gene, and JR a recurrent jumper gene. The source row shows the *id* of the source vertex of a jumper gene and the *parent* row shows the id of the parent gene.

| gene | x_1 | x_2 | x_3 | x_4 | x_5 | x_6 | x_7 | x_8 | x_9 | x_{10} | x_{11} | x_{12} | x_{13} | x_{14} | x_{15} | x_{16} |
|----------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|-------------|----------|----------|----------|----------|
| allele | V | V | V | I | I | I | V | JF | I | I | JR | V | JF | V | I | I |
| id | 0 | 1 | 3 | - | - | - | 2 | - | - | - | - | 4 | - | 5 | - | - |
| source | - | - | - | - | - | - | - | 3 | - | - | 0 | - | 2 | - | - | - |
| label | - | - | - | x | y | y | - | - | x | y | - | - | - | - | y | z |
| weight | 0.6 | 0.8 | 0.9 | 0.1 | 0.4 | 0.5 | 0.2 | 0.3 | 0.7 | 0.8 | 0.2 | 0.9 | 0.2 | 1.3 | 2.0 | -1.0 |
| d_{in} | 2 | 2 | 2 | - | - | - | 4 | - | - | - | - | 2 | - | 2 | - | - |
| v | -1 | -1 | -1 | 1 | 1 | 1 | -3 | 1 | 1 | 1 | 1 | -1 | 1 | -1 | 1 | 1 |
| s | 0 | -1 | -2 | -3 | -2 | -1 | 0 | -3 | -2 | -1 | 0 | 1 | 0 | 1 | 0 | 1 |
| parent | \emptyset | 0 | 1 | 3 | 3 | 1 | 0 | 2 | 2 | 2 | 2 | \emptyset | 4 | 4 | 5 | 5 |
| depth | 0 | 1 | 2 | - | - | - | 1 | - | - | - | - | 0 | - | 1 | - | - |

an arbitrary number $M > 0$ of inputs or forward jumper genes. The number of inputs d_{in} to x_k is set to M and its depth is set to $depth(x_i) + 1$. The depth of the source vertex of a forward jumper gene connected to x_k is not allowed to have a depth less than the depth of x_k . A good example of a *crossover operator* $\mathcal{CR} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}$ that can be used with CGE is the operator introduced by Stanley [17]. This operator aligns two genomes encoding different network topologies, and creates a new structure that combines the overlapping parts of the two parents as well as their differing parts. The *id*’s that are stored in vertex and jumper genes, and the labels that are stored in input genes, are used to align genomes.

4 Properties of the Encoding

In this section, we list some of the properties of the genetic encoding that makes it suitable for evolving neural networks. Formal proofs of these properties are given in [9]. The first property given by Proposition 1 reinforces the fourth and the fifth criterion listed in Section 3.

Proposition 1. *For a valid genotype $g \in \mathcal{G}$, the number of expected inputs by all vertex genes $\sum_{x_i \in \mathcal{G} \wedge x_i \in \mathcal{V}} d_{in}(x_i)$ is equal to $|\mathcal{V}_{no} \cup \mathcal{I} \cup \mathcal{J}_F \cup \mathcal{J}_R|$, i. e. the number of non-output vertex genes.*

The second property given by Proposition 2 relates the sum $s_{N+1} = \sum_{i=1}^N v(x_i)$ to the number of output vertex genes in a valid genotype.

Proposition 2. *For $g = [x_1, \dots, x_N] \in \mathcal{G}$ with N genes, s_{N+1} is equal to the number of output vertex genes $|\mathcal{V}_o|$ in g .*

This property can be used as a *checksum* while performing an implicit evaluation of a direct encoded phenotype, or during the development process of an indirect encoded phenotype.

The following three important properties of the genetic encoding make it suitable for evolving neural networks.

Proposition 3. (Completeness of \mathcal{G} with respect to \mathcal{D}) *Every valid phenotype $p \in \mathcal{P}_{CGE}$ can be represented by a genotype, i. e. \mathcal{D} is surjective: $\forall p \in \mathcal{P}_{CGE} \exists g \in \mathcal{G} : \mathcal{D}(g) = p$.*

This proposition conveys that for every valid phenotype, there is a valid genotype that represents this phenotype (with respect to the development function \mathcal{D}).

Proposition 4. (Closure of \mathcal{D}) *The development function maps every valid genotype to a valid phenotype: $\forall g \in \mathcal{G} : \mathcal{D}(g) \in \mathcal{P}_{CGE}$.*

The closure of \mathcal{D} guarantees the generation of genotypes whose evaluation strategy (in the case of direct encoding) or development process (in the case of indirect encoding) terminates in a finite amount of time.

Proposition 5. (Closure of \mathcal{G} under genetic operators) *The set of genotypes \mathcal{G} is closed under the mutation operators \mathcal{PA} and \mathcal{ST} : $\mathcal{PA}(g) \in \mathcal{G}$ and $\mathcal{ST}(g) \in \mathcal{G} \forall g \in \mathcal{G}$. Furthermore, it is closed under the crossover operator \mathcal{CR} : $\mathcal{CR}(g_1, g_2) \in \mathcal{G} \forall g_1, g_2 \in \mathcal{G}$.*

Proposition 5 emphasizes that the genetic operators are designed so that their output genotypes satisfy the validity criteria listed in Section 3.

5 CGE for Direct Encoding Case

In the direct encoding case, the *phenotypes* which can be represented by the valid genotypes are defined as follows: each valid phenotype $p \in \mathcal{P}_{CGE}$ is a directed graph structure $p = (V, E)$ consisting of a set of vertices V and a set of directed edges E . The set of edges E is partitioned into two subsets: the set of forward connections E_F , and the set of recurrent connections E_R . For each $p = (V, E_F \cup E_R) \in \mathcal{P}_{CGE}$, the subgraph $p_F = (V, E_F)$ is always a directed acyclic graph (DAG). The set E_R can be an arbitrary subset of $V \times V$.

The *development function* $\mathcal{D} : \mathcal{G} \rightarrow \mathcal{P}_{CGE}$ creates for every valid genotype $g = [x_1, \dots, x_N] \in \mathcal{G}$ a corresponding phenotype $p \in \mathcal{P}_{CGE}$. In the direct encoding case, for each $x_i \in \mathcal{V}$, p contains exactly one vertex \hat{x}_i , which has the same identity number as x_i , and for each recurrent jumper gene x_i , there is an edge $e \in E_R$ from a vertex whose id is equal to that of x_i 's source vertex id to the vertex in p whose id is equal to that of $\text{parent}(x_i)$. In the same way, for each $x_i \in \mathcal{J}_{\mathcal{F}}$ there is a corresponding forward connection in E_F . For each $x_i \in \mathcal{I}$, E_F contains a forward connection from the vertex having x_i 's label as id² to the

² There may be several labels possessing the same value for different input vertices, but for each unique label, there exists only one vertex in p whose id corresponds to that label.

vertex with the same id as $parent(x_i)$. Additionally, there are connections in E_F that are not explicitly represented in g . Each non-output vertex gene $x_i \in \mathcal{V}_{no}$ has an *implicit forward connection* with its parent vertex $parent(x_i)$.

The evaluation function evaluates the developed phenotype $p \in \mathcal{P}_{CGE}$. $\mathcal{D}(g)$ can be interpreted as an artificial neural network in the following way: all input vertices of $\mathcal{D}(g)$ are considered as inputs of the network and all other vertices as neuron nodes. The vertices corresponding to an output vertex gene in g are the output neurons of the network. Each forward and recurrent connection causes the output of its source neuron to be treated as an input of its target neuron. Each artificial neuron stores its last output $o_i(t-1)$. Let \hat{x}_i be a neuron with incoming forward connections from the inputs $\hat{x}_1, \dots, \hat{x}_k$ and the neurons $\hat{x}_{k+1}, \dots, \hat{x}_l$, and the incoming recurrent connections from neurons $\hat{x}_{l+1}, \dots, \hat{x}_m$. For an arbitrarily chosen transfer function φ , the current output $o_i(t)$ of the neuron \hat{x}_i is computed using

$$o_i(t) = \varphi\left(\sum_{j=1}^k w_j I_j(t) + \sum_{j=k+1}^l w_j o_j(t) + \sum_{j=l+1}^m w_j o_j(t-1)\right), \quad (7)$$

where the values of $I_j(t)$ represent the inputs of the neural network. If the network has p inputs and q output neurons, we can define \mathcal{E} as a function which takes the phenotype $\mathcal{D}(g)$ and p real input values, and produces q real output values, i.e. $\mathcal{E} : \mathcal{P}_{CGE} \times \mathbb{R}^p \rightarrow \mathbb{R}^q$. A nice feature of CGE in the direct encoding case is that it allows an *implicit evaluation* of the encoded phenotype without the need to decode this phenotype from the genotype via \mathcal{D} [8]. For this purpose, we consider the ordering of the genes in the CGE encoding to be inverted (i. e. from right to left) and evaluate it according to the Reverse Polish Notation (RPN) scheme, where the operands (input genes and jumper genes) come before the operators (vertex genes).

5.1 Exploitation and Exploration of Structures

The evolution of neural networks starts with the generation of the initial genomes. The complexity of the initial genomes is determined by the domain expert and is specified by the maximum depth that can be assumed by the genomes. It then exploits the structures that are already in the system. By exploitation, we mean optimization of the weights of the structures. This is accomplished by an evolutionary process that occurs at smaller time-scale. The evolutionary process at smaller time-scale uses parametric mutation as a search operator. An example of the exploitation process is shown in Figure 2. Exploration of structures is done through structural mutation and crossover operator. The structural selection operator that occurs at larger time-scale selects the first half of the structures (species) to form the next generation. Since sub-networks that are introduced are not removed, there is a gradual increase in the number of structures and their complexity along the evolution path. This allows the meta-level evolutionary process to search for a solution starting from a neural network with minimum structural complexity specified by the domain expert. The search stops when a neural network with the necessary optimal structure

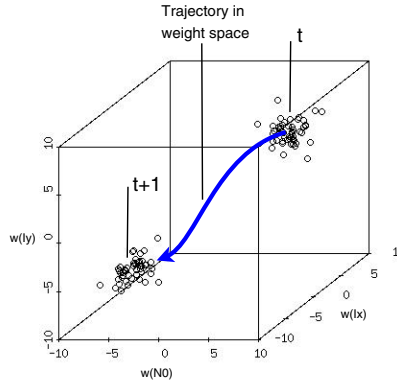


Fig. 2. The weight trajectory of a genome while it is being exploited. The quantities t and $t + 1$ are time units with respect to the larger time-scale. The weights of the existing structures are optimized between two consecutive time units with respect to the larger time-scale. The point clouds at t and $t + 1$ show populations of individuals from the same structure.

that solves a given task is obtained. The details of the exploitation and exploration of structures can be found in [8].

5.2 Learning the Dynamic Model of a Robot Manipulator

The purpose of this experiment is to demonstrate the flexibility of a CGE encoding in solving a learning task. We will illustrate how the modular property of the encoding can be exploited in solving a given task in a divide and conquer strategy manner. Given an initial state of a mechanical structure (i.e. displacements $q(0)$ and velocities $\dot{q}(0)$ of the joints) and the time history of torques $\tau(t)$ acting at joints, the direct dynamic model allows one to predict the resulting motion $q(t)$ in joint space. With this information and the direct kinematic model, a prediction of the trajectory $x(t)$ in Cartesian coordinates can be performed. For our experiment, the two-link planar arm shown in Figure 3 was used. The dynamic equation of the two-link arm [11] is used to simulate the robot. The learning system can observe the initial state $s(0) = [q(0), \dot{q}(0)]$ and $q(t)$ for t between 0 and 1 sec. For a given initial state $s(0)$, the learning system sends the robot arm the torque pair (τ_1, τ_2) for the time between 0 and 1 sec, and records the resulting motion parameters $q_1(t)$ and $q_2(t)$. For a given torque pair (τ_1, τ_2) , the resulting motion parameters are approximated by polynomials of degree 4 given by $q_1(t) = \sum_{k=0}^4 a_k t^k$ and $q_2(t) = \sum_{k=0}^4 b_k t^k$. The polynomial approximation allows the velocities to be directly calculated, where the velocities are given by $\dot{q}_1(t) = \sum_{k=1}^4 k * a_k t^{k-1}$ and $\dot{q}_2(t) = \sum_{k=1}^4 k * b_k t^{k-1}$. The genotype that represents the solution $g = [g_1, g_2]$ is made up of two subgenomes g_1 and g_2 each representing the motion parameters. To get an idea of how the genotype look like, we will explain the the subgenome g_1 in detail corresponding to the first

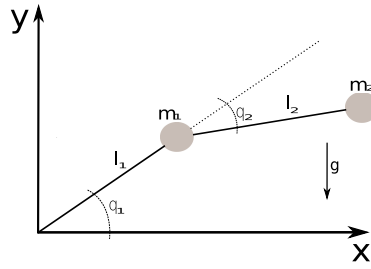


Fig. 3. A two-link planar arm robot used for our experiment

motion parameter $q_1(t)$. The polynomial approximation of $q_1(t)$ can be written as

$$q_1(t) = \sum_{k=0}^4 a_k t^k = a_0 + t(a_1 + t(a_2 + t(a_3 + a_4 t))), \tag{8}$$

where each of the coefficients a_i is represented by a neural network $M_i(\tau_1, \tau_2, q(0), \dot{q}(0))$ whose output can be computed by equation (7). If we introduce two additional vertex genes V^* and V^+ , which take the product and the sum of their arguments respectively, we can represent the polynomial approximation in CGE genotype easily. Table 2 shows the first subgenome g_1 , where M_i is a subgenome dedicated to coefficient a_i . Note that a subgenome M_i is assigned $v(x_i) = 1$ since the sum $s_{l,m}$ for a subgenome is always one. A depth is also assigned to the subgenome M_i since by definition subgenomes start with a vertex gene.

Table 2. A genotype representing the first subgenome g_1

| gene | x_1 | x_2 | x_3 | x_4 | x_5 | x_6 | x_7 | x_8 | x_9 | x_{10} | x_{11} | x_{12} | x_{13} | x_{14} | x_{15} | x_{16} | x_{17} |
|----------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|----------|----------|----------|----------|----------|
| allele | V^+ | M_0 | V^* | I | V^+ | M_1 | V^* | I | V^+ | M_2 | V^* | I | V^+ | M_3 | V^* | I | M_4 |
| id | 0 | - | 1 | - | 2 | - | 3 | - | 4 | - | 5 | - | 6 | - | 7 | - | - |
| source | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| label | - | - | - | t | - | - | - | t | - | - | - | t | - | - | - | t | - |
| weight | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| d_{in} | 2 | - | 2 | - | 2 | - | 2 | - | 2 | - | 2 | - | 2 | - | 2 | - | - |
| v | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | 1 |
| s | 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 |
| parent | \emptyset | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 6 | 7 | 7 |
| depth | 0 | 1 | 1 | - | 2 | 3 | 3 | - | 4 | 5 | 5 | - | 6 | 7 | 7 | - | 8 |

The learning process evolves each subgenome M_i independently using the meta-level evolutionary process discussed in Section 5.1. For the exploitation of structures the CMAES [6] algorithm developed by Hansen and Ostermeier is used. The parameters of the evolutionary process are set as follows: (1) Torque

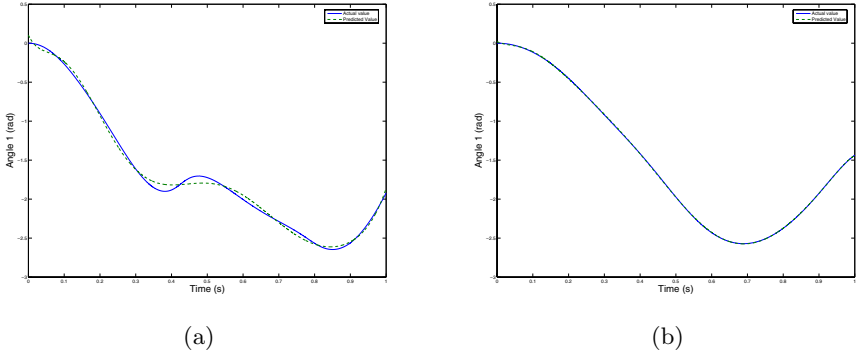


Fig. 4. Actual and predicted values for $q_1(t)$. (a) $q_1(t)$ for $\tau_1 = 0.05$, $\tau_2 = 0.05$, $q_1(0) = 0$, and $\dot{q}_1(0) = 0$. (b) $q_1(t)$ for $\tau_1 = 0.05$, $\tau_2 = 0$, $q_1(0) = 0$, and $\dot{q}_1(0) = 0$.

values are kept between -0.05 and 0.05 Nm. (2) Robot parameters are set to $m_1 = 0.05$ Kg, $m_2 = 0.05$ Kg, $l_1 = 0.25$ m and $l_2 = 0.25$ m. (3) Crossover operator is turned off. (4) Structural mutation is turned on with probability 0.3 (5) Minimal initial structure for each subgenome M_i is set to have one output vertex gene connected to inputs τ_1 , τ_2 , $q(0)$ and $\dot{q}(0)$. After learning the dynamic model of the robot, we tested it on unseen data. The performance of the learned model in predicting the motion parameters $q_1(t)$ and $q_2(t)$ is satisfactory. Figure 4 shows sample comparisons between actual and predicted values for $q_1(t)$.

6 CGE for Artificial Embryogeny

The term embryogeny refers to the growth process which defines how a genotype maps onto a phenotype. Bentley and Kumar [2] identified three different types of embryogenies that have been used in evolutionary systems: external, explicit and implicit. *External* means that the developmental process (i. e. the embryogeny) itself is not subjected to evolution but is hand-designed and defined globally and externally to the genotypes. In *explicit* (evolved) embryogeny the developmental process itself is explicitly specified in the genotypes, and thus it is affected by the evolutionary process. Usually, the embryogeny is represented in the genotype as a tree-like structure following the paradigm of genetic programming. The third kind of embryogeny is *implicit* embryogeny, which comprises neither an external nor an explicit internal specification of the growth process. Instead, the embryogeny "emerges" implicitly from the interaction and activation patterns of the different genes. This kind of embryogeny has the strongest resemblance to the process of natural evolution. A popular example of an implicit embryogeny is the Genetic Regulatory Network (GRN) [3,4]. In this section, we illustrate how CGE can be used to encode an explicit embryogeny.

In explicit embryogeny schemes, a genotype contains a program that describes the developmental process. Most of these programs are represented as tree-like

structures, where each node of the tree contains an elementary instruction (like adding/removing an entity to the phenotype, conditional statements, iteration, a subroutine call, etc.) and (optionally) parameters for these instructions. During the development process, a tree is traversed (usually either in a breadth-first or depth-first manner) and the instruction contained in the current tree node is executed. Thus, while performing the tree traversal, the phenotype is grown step-by-step. Alternatively, the instructions can be contained in the edges of the tree and be carried out when the corresponding edge is traversed. This alternative is equivalent to the case in which instructions are contained in the nodes. For the following example, therefore, we consider only the case in which the instructions are contained in the nodes.

For an encoding to be used for an explicit embryogeny scheme, it should possess the following features: (1) The encoding should be able to encode a tree structure. (2) A gene which encodes a node should contain an instruction as well as a set of parameters for this instruction. (3) The genetic operators must produce only offspring-genotypes which encode tree structures. Since the structures which can be encoded by a CGE-genotype are a superset of the set of all tree structures, one can fulfill the three conditions stated above by slight simplifications (modifications) of a CGE genotype. The first simplification is to do away with the need for input and jumper genes. In the original definition of CGE, each gene contains a weight. If CGE shall be used for explicit embryogeny, one must replace that weight by an arbitrary number of other parameters, which need not to be restricted to the domain of real numbers. The structural mutation operator must be changed in the following manner: instead of introducing recurrent jumper genes, forward jumper genes or input genes, it simply adds a new vertex gene in the genotype and increases the number of inputs of the vertex gene preceding the newly added gene. The parametric mutation operator itself remains largely unchanged - only the fields on which it operates are different: instead of modifying weights, it modifies now all parameters included in a gene, choosing the values from the domain which is associated with this kind of parameter. The crossover remains unchanged since it produces offspring which remains in the domain of tree structures.

Kassahun et al. [9] have shown a way of encapsulating the edge encoding of Luke and Spector [13] into a CGE genotype. Thus, the basic ability of CGE to perform explicit (evolved) embryogeny has already been presented. However, since the edge encoding is an encoding scheme for neural networks, the phenotype remains in the domain of neural networks. In the following, we present a simple way of evolving phenotypes from other domains. For illustration purpose we use binary images $I = \{0, 1\}^{128 \times 128}$ as phenotypes. Each vertex gene of a genotype to be used contains one of the instructions $\{LEFT, RIGHT, UP, DOWN\}$ and a binary parameter $f \in \{0, 1\}$. The growth process is as follows: Initially, all pixels of I are equal to 0 (i. e. white) and a virtual cursor points to the pixel with coordinates $x = 0, y = 0$. Then, a depth first traversal is performed and the instructions are executed as follows: If the instruction is *LEFT*, set $x = x - 1 \text{ MOD } 128$ and $I[x][y] = f$. If the instruction is *RIGHT*, set $x = x + 1 \text{ MOD } 128$

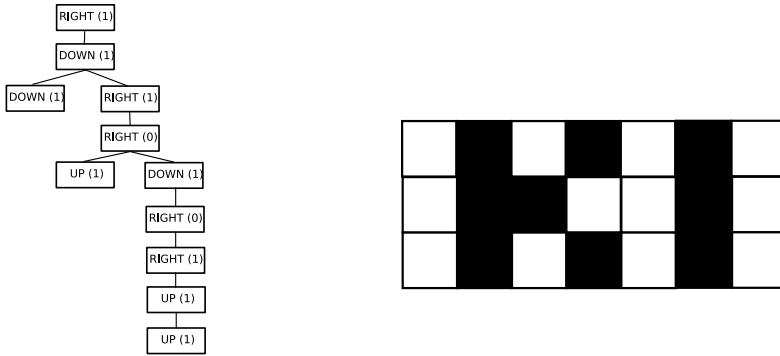


Fig. 5. The figure shows a genotype and the corresponding phenotype showing the letters "KI". The genotype is shown as a tree-like structure, which can be easily represented as a string of genes if the tree is traversed in a depth-first manner. The value of f is shown in parentheses. The phenotype is a binary image, where the pixel with a coordinate $x = 0, y = 0$ is in the upper left corner of the image.

and $I[x][y] = f$. If the instruction is *UP*, set $y = y - 1 \text{ MOD } 128$ and $I[x][y] = f$. If the instruction is *DOWN*, set $x = y + 1 \text{ MOD } 128$ and $I[x][y] = f$. When traversing the instruction in the other direction (on the way back), the original cursor is restored: For example when traversing a *LEFT* instruction on the way back, we set $x = x + 1 \text{ MOD } 128$. Figure 5 shows a genotype and the corresponding phenotype *KI*.

7 Comparison of CGE to Other Genetic Encodings

In this section, a comparison among some genetic encodings developed so far and CGE with respect to the completeness, closure, modularity properties and some additional features is given. Table 3 shows comparison among some representative genetic encodings developed so far. For the direct encoding case, the "eval-

Table 3. Comparison among some representative genetic encodings and CGE. G, N, CE, and E stand for GNARL, NEAT, Cellular Encoding, and Edge Encoding, respectively.

| Property | G | N | CE | E | CGE |
|--|---|---|----|---|-----|
| Completeness | √ | √ | √ | √ | √ |
| Closure | × | √ | √ | √ | √ |
| Modularity | × | × | √ | √ | √ |
| Support both direct and indirect encoding | × | × | × | × | √ |
| Evaluation without decoding (direct encoding case) | × | × | × | × | √ |

uation without decoding” feature of CGE eliminates a step in the phenotype-development process that would otherwise require a significant amount of time, especially for large and complex phenotype networks.

8 Conclusion and Outlook

A flexible genetic encoding that is both complete and closed, and which is suitable for both direct and indirect genetic encoding of networks has been presented. Since the encoding’s genotypes can be seen as having several subgenomes, it inherently supports the evolution of modular networks in both direct and indirect encoding cases. Additionally, in the direct encoding case, the genotype has the added benefit of being able to evaluate a phenotype without the need to first decode it from the genotype.

In the future, we will investigate the design of indirect encoding operators which can achieve compact representations and significantly reduce the search space. We also believe that there is much work to be done in designing genetic operators. In particular, there is a need for genetic operators whose offspring remain in the locus of similarity to their parents in both structural and parametric spaces. More efficient evolution of complex structures would be facilitated by such operators.

References

1. Angeline, P.J., Saunders, G.M., Pollack, J.B.: An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks* 5, 54–65 (1994)
2. Bentley, P., Kumar, S.: Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, 13-17 July, 1999*, vol. 1, pp. 35–43. Morgan Kaufmann, San Francisco (1999)
3. Bongard, J.C., Pfeifer, R.: Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pp. 829–836 (2001)
4. Dellaert, F., Beer, R.D.: A developmental model for the evolution of complete autonomous agents. In: *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pp. 393–401 (1996)
5. Gruau, F.: *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Supérieure de Lyon, Laboratoire de l’Informatique du Parallélisme, France (January 1994)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
7. Jakobi, N.: Harnessing morphogenesis. In: *Proceedings of Information Processing in Cells and Tissues*, pp. 29–41 (1995)
8. Kassahun, Y.: *Towards a Unified Approach to Learning and Adaptation*. PhD thesis, Technical Report 0602, Institute of Computer Science and Applied Mathematics, Christian-Albrechts University, Kiel, Germany (February 2006)

9. Kassahun, Y., Edgington, M., Metzen, J.H., Sommer, G., Kirchner, F.: A common genetic encoding for both direct and indirect encodings of networks. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2007 (accepted, July 2007)
10. Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. *Complex Systems* 4, 461–476 (1990)
11. Lewis, F.L., Dawson, D.M., Abdallah, C.T.: *Robot Manipulator Control: Theory and Practice*. Marcel Dekker, Inc., New York, Basel (2004)
12. Lindenmayer, A.: Mathematical models for cellular interactions in development, parts I and II. *Journal of Theoretical Biology* 18, 280–315 (1968)
13. Luke, S., Spector, L.: Evolving graphs and networks with edge encoding: Preliminary report. In: Late-breaking papers of Genetic Programming 1996, Stanford, CA (1996)
14. Nolfi, S., Parisi, D.: Growing neural networks. Technical Report PCIA-91-15, Institute of Psychology, Rome (1991)
15. Schaffer, J., Whitley, L.D., Eshelmann, L.J.: Combination of genetic algorithms and neural networks: A survey of the state of the art. In: Proceedings of COGANN92 International Workshop on the Combination of Genetic Algorithm and Neural Networks, pp. 1–37. IEEE Computer Society Press, Los Alamitos (1992)
16. Sendhoff, B., Kreutz, M.: Variable encoding of modular neural networks for time series prediction. In: Congress on Evolutionary Computation (CEC'99), pp. 259–266 (1999)
17. Stanley, K.O.: Efficient Evolution of Neural Networks through Complexification. PhD thesis, Artificial Intelligence Laboratory. The University of Texas at Austin, Austin, USA (August 2004)
18. Vaario, J., Onitsuka, A., Shimohara, K.: Formation of neural structures. In: Proceedings of the Fourth European Conference on Artificial Life, ECAL97, pp. 214–223 (1997)
19. Yao, X.: Evolving artificial neural networks. *Proceedings of the IEEE* 87(9), 1423–1447 (1999)

Making a Robot Learn to Play Soccer Using Reward and Punishment

Heiko Müller², Martin Lauer¹, Roland Hafner¹, Sascha Lange¹, Artur Merke²,
and Martin Riedmiller¹

¹ Neuroinformatics Group, Institute of Computer Science and Institute of Cognitive Science, University of Osnabrück, 49069 Osnabrück, Germany

² Lehrstuhl Informatik 1, University of Dortmund, 44221 Dortmund, Germany

{Martin.Lauer,Roland.Hafner,Sascha.Lange,Martin.Riedmiller}@uos.de,
{heiko.mueller,artur.merke}@udo.edu

Abstract. In this paper, we show how reinforcement learning can be applied to real robots to achieve optimal robot behavior. As example, we enable an autonomous soccer robot to learn intercepting a rolling ball. Main focus is on how to adapt the Q-learning algorithm to the needs of learning strategies for real robots and how to transfer strategies learned in simulation onto real robots.

1 Introduction

Although various machine learning techniques have been used successfully in robotics, the idea of reinforcement learning [16] has not been applied very often on real robots so far. This is surprising since the basic idea of learning how to control autonomous robots just by rewarding them for good behavior and punishing them for bad behavior is very promising for solving complex tasks for which no optimal solution yet exists.

In simulation environments, reinforcement learning techniques have already been applied successfully (e.g. [6]), but the step from simulation to real world application has posed many problems. While in simulation, all state variables of a control task are perfectly known, on real robots these values have to be estimated using noisy and unreliable sensory input. Hence, state estimation shows a much larger error.

Secondly, state spaces are typically very large. While the theory of reinforcement learning is based on the assumption of finite state spaces, in practice we are often faced with infinite vector-valued state spaces with ten or more dimensions. Hence, approximations [2] have to be used in order to learn efficiently and to represent the control strategy. These techniques might even disturb the convergence of the learning algorithms [12].

Furthermore, high dimensional state spaces require millions of training examples to be able to learn an optimal policy. These millions of examples typically cannot be generated with a real robot since the robot would breakdown. Again, simulators and model assumptions must be used to overcome this problem.

Finally, real applications typically violate the Markov property which is the basis for all learning algorithms. Due to latencies in sensory processing and actuator execution, we are faced with time gaps of more than $100ms$, in many applications even much more [3]. In contrast, control cycles of many software frameworks are typically below $50ms$, so that an action is not executed completely when the next action must be selected.

Asada et al. have learned successfully different tasks on real robots using reinforcement learning [117,119]. In particular, they evaluated the automatic construction of higher-order state descriptions in order to solve the delay problem [19]. But due to several simplifications in their problem formulations, the resulting strategies have not been competitive and often are far from optimal control.

Speaking more generally, all the problems mentioned above have prevented reinforcement learning approaches so far from being competitive on real autonomous robots. Within this paper, we present a reinforcement learning approach to an autonomous soccer playing robot. The learning task is to intercept a rolling ball. This task has been discussed for a simulated robot before [6] and has now been transferred to the special needs of real robots. The intercept policy finally learned was integrated into the soccer robot strategy of the RoboCup [9] middle size league team *Brainstormers Tribots*, which became world champion in 2006.

In section 2 we first introduce the basic concepts of reinforcement learning which are relevant for this work and then show how reinforcement learning can be used to learn strategies for real robots in section 3. There, we discuss how to overcome the problems that have prevented the application of reinforcement learning on real robots until now. Section 4 shows the experimental results of learning the intercept scenario in simulation and on the real robots. The paper closes with a summary of the main results in section 5.

2 Reinforcement Learning

2.1 Markov Decision Process

Reinforcement learning is based on the idea of an autonomous agent interacting with an unknown environment. The agent observes the environment and decides to choose one out of several possible actions for interacting with the world. Depending on the agent's choice, the environment will change its state. Furthermore, the agent gets a reward for each state-transition. This reward can also be negative, i.e. the agent is punished instead of rewarded. The agent's goal is to achieve as much accumulated reward as possible over time.

This basic idea of reinforcement learning is modeled as a Markov decision process (MDP) [16]. It consists of a finite set of states S of the environment, a finite set of actions A that the agent can choose, a probabilistic state-transition kernel $P = (p_{s,s'}^a)$ where $p_{s,s'}^a$ describes the probability of a transition from state s to s' if the agent chooses action a , and a reward function $r : S \times A \rightarrow \mathbb{R}$ that defines the reward provided to the agent, depending on the current state and action. An important property of MDPs is that the transition probabilities

are independent of past states and actions. This property simplifies theoretical analysis of reinforcement learning.

The behavior of an agent is described in terms of a policy $\pi : S \rightarrow A$. $\pi(s)$ is the action chosen by the agent in state s . Applying policy π in state s , the agent will get an immediate reward of $r(s, \pi(s))$ and the environment will randomly change to another state s' with probability $p_{s,s'}^{\pi(s)}$. If we continue applying policy π , we will observe a sequence of states and rewards. The goal of reinforcement learning is to find a policy that maximizes the expected sum of rewards over time:

$$\underset{\pi}{\text{maximize}} \quad E\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\right] \quad (1)$$

where r_t denotes the reward that is achieved in the t -th step and $\gamma \in [0, 1)$ is a discount factor that is introduced to guarantee convergence of the infinite sum in (1). It is typically chosen close to 1¹.

Using the Markov property of the MDP, we can unroll the sum in (1) and get a fixed point equation which is known as Bellman equation:

$$E\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\right] = r(s, \pi(s)) + \gamma \sum_{s' \in S} \left(p_{s,s'}^{\pi(s)} E\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s', \pi\right]\right) \quad (2)$$

It has been shown [?] that for each MDP there exist policies π^* that maximize $E\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right]$ for all states $s_0 \in S$ at the same time. These policies are called *optimal policies*. Hence, the goal of reinforcement learning is to find an optimal policy.

2.2 Value Iteration

One way to calculate an optimal policy is first to determine the optimal expected reward and derive an optimal policy afterwards. The optimal expected reward is described in the form of a *value function* $V^* : S \rightarrow \mathbb{R}$. $V^*(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi^*\right]$ denotes the accumulated expected reward if we apply an optimal policy π^* starting in state s .

Rewriting the Bellman equation (2) and taking into account that the optimal policy always chooses the action having the maximal expected reward, we get a fixed point equation for V^* :

$$V^*(s) = \max_{a \in A} \left(r(s, a) + \gamma \sum_{s' \in S} (p_{s,s'}^a V^*(s')) \right) \quad (3)$$

The right hand side of (3) can be interpreted as an operator that maps value functions onto value functions. It turns out that this operator is a contraction mapping in the space of value functions [4]. Thus, applying Banach's fixed point theorem, we can approximate the optimal value function V^* starting with an

¹ Variants of this optimization goal exist, see [4].

arbitrary initial value function V_0 and perpetually applying the operator defined by the right hand side of (3).

Furthermore, having once calculated V^* , we can derive an optimal policy π^* from V^* using a greedy evaluation scheme for the value function:

$$\pi^* : s \mapsto \arg \max_{a \in A} \left(r(s, a) + \gamma \sum_{s' \in S} (p_{s, s'}^a V^*(s')) \right) \quad (4)$$

The idea of using an iterative process to calculate the optimal value function and afterwards greedily deriving an optimal policy is implemented by a reinforcement learning algorithm called *value iteration* [4]. The value function is stored in the form of a table. In order to perform the update steps of value iteration, the transition probabilities $p_{s, s'}^a$ and the reward function $r(s, a)$ must be known.

2.3 Q-Learning

The main disadvantage of the value iteration approach is the fact that the state transition probabilities and the reward function must be known in advance. In real world applications of reinforcement learning, e.g. in autonomous robot control, these values are typically not known so that value iteration cannot be applied. To overcome this problem, a learning algorithm named Q-learning has been developed that is not based on this knowledge but that is able to implicitly estimate these values while interacting with the environment and observing transitions.

The main idea is to introduce value functions Q^* that depend on both the current state and the action that potentially might be chosen by the agent. $Q^*(s, a)$ models the expected reward of an agent that starts in state s , chooses action a first and acts optimally later on, i.e.:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in S} (p_{s, s'}^a V^*(s')) \quad (5)$$

Using this definition, we can rewrite the Bellman equation (3) as:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in S} (p_{s, s'}^a \max_{a' \in A} Q^*(s', a')) \quad (6)$$

which yields a fixed point equation for Q^* instead of V^* . The optimal policy can be derived directly from the Q^* -function using greedy evaluation:

$$\pi^* : s \mapsto \arg \max_{a \in A} Q^*(s, a) \quad (7)$$

If we knew the transition probabilities and the reward function, we could apply a dynamic programming-like algorithm similar to value iteration to approximate Q^* . In contrast, the *Q-learning* algorithm [20] does not assume these values to be known in advance. Instead, by observing state transitions of the environment, the algorithm collects quadruples of predecessor state, action, reward and successor

state and it approximates the Q^* function using a stochastic approximation scheme. The Q -function is updated incrementally every time a state transition is observed.

After having observed a transition from state s to state s' using action a and obtaining reward r , the Q -function is updated using the following rule:

$$Q(s, a) \leftarrow \alpha(r + \gamma \max_{a' \in A} Q(s', a')) + (1 - \alpha)Q(s, a) \quad (8)$$

where $\alpha > 0$ is a learning rate decreasing over time. Q -learning is guaranteed to converge towards the optimal Q^* -function as long as all combinations of states and actions are observed repeatedly and the decrease of the learning rate fulfills certain conditions [4].

The advantage of Q -learning in comparison to value iteration is the fact that the optimal policy can be learned only by interacting with the environment. No knowledge of the true transition probabilities or the reward function is necessary. Hence, it can also be applied to real-world tasks with unknown system dynamics. Similar to value iteration, the Q -function is typically stored in a table-based representation which can become very large if S and A are large.

3 Learning on a Real Robot

3.1 Robot Learning Task

The task discussed in this paper is the problem of an autonomous mobile soccer robot intercepting rolling ball, see Fig. 1. Intercepting means that the robot is moving to a certain point where it can interrupt the current ball movement and can get control of the ball. In order to do this, the robot must touch the ball with its front side and the difference in robot and ball velocities must be small, i.e. less than $0.6 \frac{m}{s}$. In this study, we are only interested in situations like the one depicted in Figure 1 in which the ball moves towards the robot. Situations, in which the ball rolls away from the robot are of no interest.

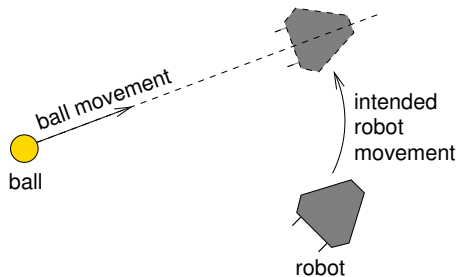


Fig. 1. Intercepting a rolling ball. The robot must move to a point on the line that is given by the ball movement and must turn to face the ball. To intercept the ball, the difference in velocity between the robot and the ball must also be small.

The robot is assumed to be able to recognize its environment with a camera so that it can determine the ball's position and the ball velocity. The interpretation of the camera images and the estimation of the ball velocity are assumed to be done by existing sensor processing [11] and are not subject of this work. Furthermore, we assume that the robot is able to move in any direction without turning in advance.

For our experiments, we used an already existing soccer robot of the RoboCup middle size league team *Brainstormers Tribots* [7]. The robot is 80cm large, 40cm wide (see Fig. 2) and has a holonomic drive with three degrees of freedom, so that it can move in all directions and turn simultaneously. Its maximal velocity is $2.5 \frac{m}{s}$. The robot is driven by three electric motors that are individually controlled by PID-controllers. To recognize the ball, the robot is equipped with a catadioptric camera that allows a 360° view of its surroundings. With the help of this camera system, it can recognize objects up to a distance of 6m, e.g. a ball rolling on the ground.



Fig. 2. A robot of the *Brainstormers Tribots* team that was used for our experiments

3.2 Modeling the Learning Task

In order to apply reinforcement learning algorithms to the intercept problem, we must describe the task in terms of an MDP. The state space of the intercept problem consists of the following variables:

- position of the robot (2-dimensional)
- orientation of the robot (1-dimensional)
- linear velocity of the robot (2-dimensional)
- angular velocity of the robot (1-dimensional)
- ball position (2-dimensional)
- linear velocity of the ball (2-dimensional)

In summary, the state space is described by 10 variables. Since the pose of the robot with respect to some global coordinate system is irrelevant for the intercept task, we can omit the robot pose variables and represent the ball position relative to the robot position. Hence, we are left with seven state variables. Since even this number of variables is still too large for reinforcement learning algorithms, we further simplified the problem using a standard orientation controller that always turns the robot such that it faces the ball. Therefore, this orientation controller becomes part of the environment. By aligning the coordinate system with the direction of the ball movement, we can omit the state variables that describe the direction of the ball movement and the angular velocity of the robot so that we are left with a five-dimensional state space. However, the state space remains infinite.

Due to the holonomic drive of the robot, the complete action space contains accelerations between 0 and a maximal value in any direction. However, reinforcement learning approaches only allow finite and – typically – very small sets of actions. Therefore, we use a discretization of the complete action space, i.e. the robot is allowed to accelerate to one of eight directions organized in 45° angles, see Fig. 3. The amount of acceleration used is maximal with respect to the acceleration capabilities of the robot. After selecting one of the eight actions, the setpoints of the motor controllers are updated appropriately to achieve the desired acceleration. As mentioned above, the orientation of the robot is controlled by a standard controller so that we do not need to learn actions in order to turn the robot.

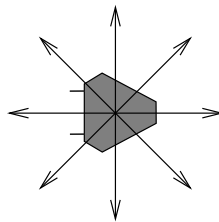


Fig. 3. The eight directions of acceleration

To achieve an MDP we need to model the temporal processing of the robot control task carefully, since we have to guarantee that the Markov property is met. Due to delays in the motor controllers, inertia, and delays in camera image processing, an action is not executed immediately but with a certain delay, and its consequences can only be observed after a certain amount of time. Measurements have shown that the execution of an action needs approximately $240ms$, which is much longer than the control cycle of our software framework which amounts to $40ms$.

If we had decided on a new action every $40ms$, the subsequent actions would not have been executed completely and the Markov property would have been violated. This problem has also been described in [35]. While there, a state

prediction has been used based on physical motion models of the robot and the ball [10,11] to close the time gap between selection and execution of an action, here, we extended the time between two subsequent actions from $40ms$ to $240ms$. Hence, when a subsequent action is selected, the consequences of the prior action have already become part of the state variable. By doing so, we again meet the Markov property.

The reward function for the learning task was designed to achieve a robot behavior that is optimal in an intuitive sense, i.e. the robot should get possession of the ball as quickly as possible. Therefore, the reward function yields a reward of -0.2 for every step to punish the robot for wasting time and a reward of $+500$ if the robot gets control of the ball, i.e. it touches the ball at its front side and the difference between the velocity of the robot and that of the ball is no larger than $0.6 \frac{m}{s}$. The discount factor is set to $\gamma = 0.92$ throughout all experiments.

3.3 Value Function Approximation

The state space of the intercept problem has been defined as a five-dimensional space. Hence, there are infinitely many states, and a table-based representation of value functions or policies is no longer possible. Moreover, observing transitions from each possible state is also not possible. Thus, we need to generalize over subsets of states to obtain policies for all states and use approximators to represent the value functions [2].

Although it has been shown that the convergence properties of reinforcement learning algorithms might be lost in some cases [12] and that only for very special situations can convergence be guaranteed [18], this approach has been investigated in many studies on reinforcement learning. In particular, linear function approximators with nonlinear features have been applied and have shown good performance.

Hence, we used two kinds of linear function approximators within this study, grid maps and lattice maps. Grid maps implement a piecewise constant function

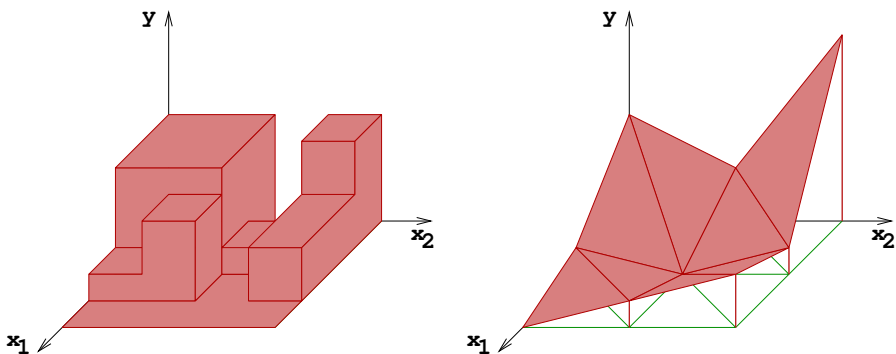


Fig. 4. Example of a grid map (left) and a lattice map (right) over a two-dimensional feature space

on the basis of some regular tessellation of the input space (see Fig. 4 left). The resulting function is a step-function. The height of each step is determined by the average value of all training examples located within the respective cell of the input space. To achieve an appropriate degree of precision, the number of cells must be very large and training examples must be located in each cell. Using grid maps is equivalent to a discretization of the input space.

Lattice maps are based on a tessellation of the input space into simplices using Kuhn triangulation [13]. The resulting function is defined individually for each vertex of the simplices and is interpolated between the vertices. By doing so, the resulting function is continuous and piecewise linear (see Fig. 4 right). Compared to grid maps, the number of simplices used for lattice maps might be much smaller than the number of cells in grid maps of the same precision. Hence, the memory requirements are smaller, the generalization performance is better and the number of training examples necessary is smaller. However, it has been shown that lattice maps may become instable using temporal difference updates and have a higher risk of not converging when compared to grid maps [12]. Fortunately, it has been possible to derive constraints under which the lattice maps remain stable and eventually converge [?]. To train the lattice maps, we used the Kaczmarz update rule [?] which turns out to be more stable than a gradient descent update.

3.4 Training the Robot

Since reinforcement learning is based on learning from experience, we needed to collect data from experiments with our robots. Unfortunately, all reinforcement learning algorithms need millions of examples to perform adequately. Collecting these examples on real robots is not possible since the robots would break. Therefore, we built a simulator tool that implements a simple physical model of the soccer scenario. It allowed us to generate a huge amount of training samples. However, since the simulation is only a crude image of the real soccer scenario, it is necessary to evaluate the results on real robots, as well.

Several experiments have been done on learning how to intercept a rolling ball. First, the policies were learned in simulation and afterwards evaluated in a simulation and on the real robot. Experiments were made using Q-learning combined with grid maps and lattice maps for approximation of the value function.

4 Experimental Results

4.1 Grid Map

In the first experiment, we analyze whether it is even possible to learn the intercept problem with Q-learning. We have chosen the grid map in this experiment to get rid of problems of non-convergence of function approximators. Unfortunately, it turns out that with a five-dimensional feature space the size of the grid map chosen must be very large in order to achieve an approximation with sufficient precision. Therefore, we restricted the experiments to a very limited

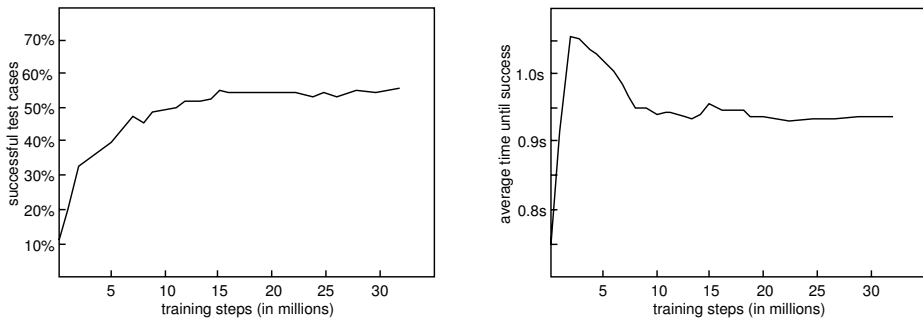


Fig. 5. Left: success rate for Q-learning with grid map on test cases dependent on the number of training steps. Right: average time needed to get control of the ball in the successful test cases.

working area: the ball had to be located within a wedge of $1.5m$ radius and 60° angle in front of the robot. The robot velocity was restricted to $1 \frac{m}{s}$, the ball velocity to $0.5 \frac{m}{s}$. However, the resulting grid partitioned the state space into 225,000 cells. Considering each of the eight possible actions, the Q-function was represented by 1.8 million cells in total. For each cell, we had to train one parameter of the function approximator.

The training was done in the simulator described in section 3.4. If the ball left the working area, a trajectory was finished and a new starting point for the ball within the working area was randomly chosen. The robot followed an ϵ -greedy exploration strategy by selecting with a probability of 0.8 the action that seemed to be optimal concerning the Q-function learned yet and with probability $\epsilon = 0.2$ a random action (exploration rate). The learning rate α decreased over time from 0.4 to 0.01.

Periodically, the performance of the learned policy was evaluated on a test set of 50,000 starting points uniformly distributed in the working area. The robot had a maximal time of 3 seconds to get control of the ball.

Figure 5 shows the percentage of successfully solved test cases dependent on the number of training steps. After 15 million training steps, Q-learning reached its best performance with a success rate of 55%. From other experiments, we know that for at least 70% of all test cases it is possible to intercept the ball. However, since the test cases are generated randomly, there are cases in which no robot control strategy can be successful, i.e. a success rate of 100% is not possible. The right hand plot in Fig. 5 shows the time needed to get control of the ball in all successful test cases. By continuous optimization of the learned strategy the robot is able to decrease the necessary time from 1.05s to 0.95s on average.

4.2 Lattice Map

Although the experiments with the grid map showed the general possibility of learning the intercept problem using reinforcement learning techniques, the results are not convincing. Neither the measured performance nor the limited

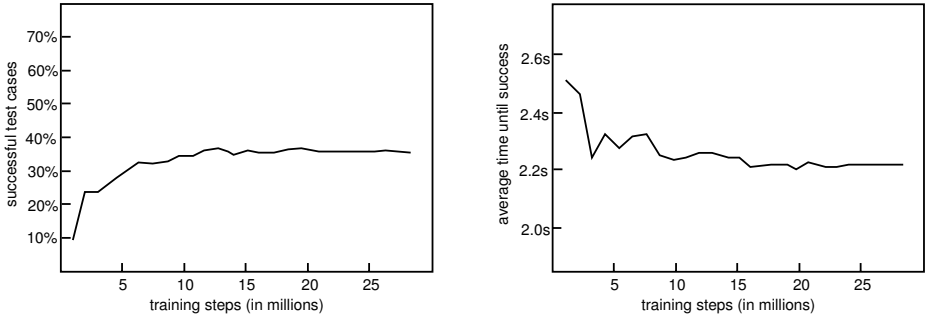


Fig. 6. Left: success rate for Q-learning with lattice map on test cases dependent on the number of training steps. Right: average time needed to get control of the ball in the successful test cases.

working area of this approach is suitable for the application of the learned strategy on a real robot. Therefore, we repeated the experiments with a lattice map function approximator instead of a grid map. Due to the better approximation performance of lattice maps, we could extend the working area to a wedge in front of the robot with radius $5m$ and an angle of 90° . The maximal robot velocity was increased to $2.5\frac{m}{s}$, the maximal ball velocity to $3.5\frac{m}{s}$.²

The lattice map had 161,568 grid points per action, with a total of 1.3 million parameters for all actions, which is smaller than in the grid map case although the working area is larger by a factor of 87.5. To avoid non-convergence of the lattice map during training, we did not consider whole trajectories but only single transitions from one state to another where only grid points were used as starting state. Using a simulator, this could be achieved easily.³

Figure 6 shows the results of the lattice map approach. The success rate is smaller than in the grid map case and achieves at most 36% after 13 million training steps. However, this can be explained by the fact that a larger working area was used with ball velocities up to $3.5\frac{m}{s}$ that are very hard to intercept for any strategy. To illustrate that the performance of the learned strategy is good, we introduced a second criterion to measure whether a strategy is able to touch the ball. Figure 7 shows the learning results using this ball contact criterion. The policy learned achieves a 70% success rate. In contrast, a simple strategy that always drives the robot into the direction of the ball with maximal velocity only achieves a success rate of 63%.

4.3 Learning for the Real Robot

The experiments described in section 4.1 and 4.2 have been made in a simulation environment without considering any delay in sensors and actuators which

² Please note, although the ball initially may roll faster than the robot can move its velocity constantly decreases, thus possibly allowing a soft interception.

³ When *exploiting* the learned strategy, e.g. on the real robot, no such constraint has to be fulfilled.

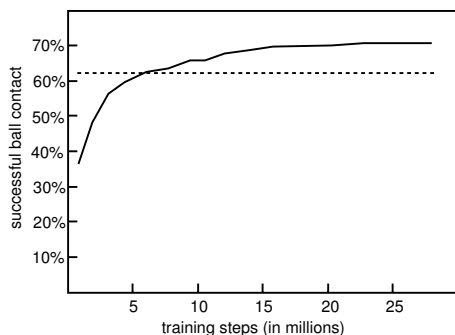


Fig. 7. Success rate in test cases with respect to the ball contact criterion. The solid line shows the success rate for the robot behavior learned using Q-learning and a lattice map, the dashed line shows the success rate of a simple robot that always drives the robot into the direction of the ball with maximal velocity.

typically can be found on real robots (cf. section 3.2). To obtain results that are closer to reality and that can be transferred to a real robot, we repeated the experiments with the lattice map incorporating an artificial delay in the simulator.

Compared to the experiments without delay, the success rates of intercepting decreased from 36% to 18%, the success rate of ball contacts decreased from 70% to 66%, while the average time needed to intercept the ball remained almost the same. The tremendous influence of delays also is illustrated by the fact that the success rate in ball contacts of the simple reference strategy that always drives the robot into the direction of the ball decreased from 63% to 31%. A lot of these failures were caused by the robot bumping too hard into the ball, thus violating the “soft interception” constraint and causing the ball to bounce away.

Using the simulator with delays enabled us to learn in simulation a strategy that can be transferred to the real robot. In order to do this, we integrated the learned strategy into the software framework of our real robots. Information about the environment such as robot pose, robot velocity, ball position, and ball velocity are estimated and provided within a central world model by the existing software framework. The control of the motors is implemented there as well.

In contrast to simulation, the information available on the real robot is much noisier and less reliable. Figure 8 shows position estimates of the robot and the ball in simulation and on real robots for comparable situations. Certainly, the high noise level disturbs the intercepting strategy and increases the task complexity.

To test the performance of the intercept strategy learned, we performed 30 experiments in our laboratory where the ball was rolling down a ramp to achieve a certain velocity. Three different ball velocities ($0 \frac{m}{s}$, $1 \frac{m}{s}$ and $2 \frac{m}{s}$) and three different geometric configurations of initial robot and ball position were used. In 10 out of 30 experiments the robot succeeded in intercepting the ball according to the “soft interception” condition, constraining the difference of ball velocity and robot velocity in the moment of the contact. In most of the other experiments it moved towards the right direction but bounced the ball away.

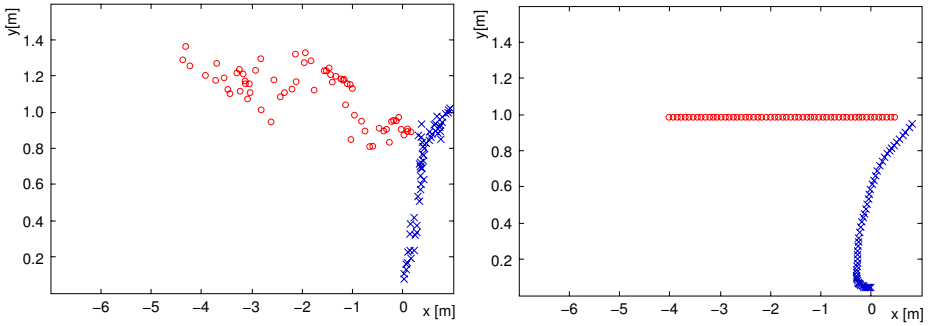


Fig. 8. Comparison of the estimated robot (crosses) and ball (circles) positions in simulation (right) and on the real robot (left) for two comparable situations. The noise level is much higher on real robots than in simulation.

To gain experience from the performance of the learned behavior in more complex and more realistic situations, we integrated it into the competition code of our RoboCup middle size league team the *Brainstormers Tribots*. The strategy of the robots is realized in a behavior-based architecture fusing principles of BDI-like architectures with ideas from the subsumption architecture. The whole strategy of the robots is realized by a number of several smaller submodules, named *behaviors*, and a multi-level hierarchical arbitration mechanism. Within a level of the hierarchy the arbitrator uses a priority ordering of the available behaviors and logical constraints connected to each of the behaviors to decide which behavior should become active in a particular situation.

The intercept behavior learned has been integrated into a larger submodule realizing all approaches to the ball. Whereas the learned behavior is activated only in situations where the ball rolls with a certain velocity and angle *towards* the robot, several other handcoded strategies solve the easier situations where the ball moves away from the robot or lays still on the ground.

We used the embedded learned strategy throughout the world championships 2006 in Bremen. On average, during a game of 30 minutes duration, the intercept behavior was used approximately 30 times per robot. The embedding handcoded behaviors were able to quickly get control of the slowly moving ball in many of the cases the interception strategy failed and bounced the ball away. Although statistics on the success rate of a single behavior in a complex real game is misleading, the entire robot behavior and especially the ability to approach the ball faster and more reliable than any other team was the most important factor in winning the RoboCup World Championship 2006 in the middle size league.

5 Discussion

So far, reinforcement learning has been applied so far primarily in simulated environments, while its application to real autonomous robots has been limited

by a set of factors in which a simulated world differs from a real application, e.g. sensor and actuator delays which are in conflict with the Markov property, limited possibilities for training on the real robot due to hardware constraints, large state spaces and noisy state estimation procedures.

By means of the ball intercepting problem, we have exemplified how these problems can be tackled and how strategies for real robots can be learned successfully. In order to do so, we have generated a modeling of the problem that is consistent with the theoretical needs of reinforcement learning but also covers the real world demands of autonomous robots. Using lattice maps as function approximators to represent the Q-function and combining them with the Q-learning algorithm and a simulator to generate training examples, successful policies could be learned.

In experiments using the simulator, we showed the principle applicability of reinforcement learning to the intercept problem and improved the modeling using lattice maps and an extended working area of the learned intercept routine. Finally, by transferring the intercept strategy learned to the real robot and integrating it into a larger software framework, we were able to test the behavior learned on a real robot and compare the results to the simulated test results. Moreover, we integrated the learned intercept strategy into the tournament code of our soccer robots and became world champion.

Acknowledgments

This work was supported by the German Research Foundation DFG SPP 1125.

References

1. Asada, M., Noda, S., Tawaratsumida, S., Hosoda, K.: Vision-based reinforcement learning for purposive behavior acquisition. In: Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 146–153. IEEE Computer Society Press, Los Alamitos (1995)
2. Baird, L.C.: Residual algorithms: Reinforcement learning with function approximation. In: Proceedings of the 12th International Conference on Machine Learning, pp. 30–37 (1995)
3. Behnke, S., Egorova, A., Gloye, A., Rojas, R., Simon, M.: Predicting away robot control latency. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 712–719. Springer, Heidelberg (2004)
4. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific (1996)
5. Gabel, T., Hafner, R., Lange, S., Lauer, M., Riedmiller, M.: Bridging the gap: Learning in the robocup simulation and midsize league. In: Proc. 7th Portuguese Conference on Automatic Control (Controlo 2006) (2006)
6. Gabel, T., Riedmiller, M.: Learning a partial behavior for a competitive robotic soccer agent. *Künstliche Intelligenz* 20(2), 18–23 (2006)
7. Hafner, R., Lange, S., Lauer, M., Riedmiller, M.: Brainstormers Tribots team description. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup-2006. LNCS(LNAI), vol. 4434, Springer, Heidelberg (2006)

8. Howard, R.A.: Dynamic programming and Markov processes. MIT Press, Cambridge (1960)
9. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: RoboCup: A challenge problem for AI. *AI Magazine* 18(1), 73–85 (1997)
10. Lauer, M.: Ego-motion estimation and collision detection for omnidirectional robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) *RoboCup 2006: Robot Soccer World Cup X. LNCS(LNAI)*, vol. 4434, Springer, Heidelberg (2006)
11. Lauer, M., Lange, S., Riedmiller, M.: Motion estimation of moving objects for autonomous mobile robots. *Künstliche Intelligenz* 20(1), 11–17 (2006)
12. Merke, A., Schoknecht, R.: A necessary condition of convergence for reinforcement learning with function approximation. In: *Proceedings of the 19th International Conference on Machine Learning*, pp. 411–418 (2002)
13. Munos, R., Moore, A.: Variable resolution discretization for high-accuracy solutions of optimal control problems. In: *International Joint Conference on Artificial Intelligence*, pp. 1348–1355 (1999)
14. Pareigis, S.: Adaptive choice of grid and time in reinforcement learning. *Advances in Neural Information Processing Systems* 10, 1036–1042 (1997)
15. Schoknecht, R., Merke, A.: Convergent combinations of reinforcement learning with linear function approximation. *Advances in Neural Information Processing Systems* 15 (2003)
16. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge (1998)
17. Suzuki, S., Kato, T., Asada, M., Hosoda, K.: Behavior learning for a mobile robot with omnidirectional vision enhanced by an active zoom mechanism. In: *Proc. of Intelligent Autonomous System 5(IAS-5)*, pp. 242–249 (1998)
18. Tsitsiklis, J.N., Van Roy, B.: Analysis of temporal-difference learning with function approximation. In: *Advances in Neural Information Processing Systems 1996*, pp. 1075–1081 (1996)
19. Uchibe, E., Asada, M., Hosoda, K.: Behavior learning for a mobile robot with omnidirectional vision enhanced by an active zoom mechanism. In: Birk, A., Demiris, J. (eds.) *Learning Robots. LNCS (LNAI)*, vol. 1545, Springer, Heidelberg (1998)
20. Watkins, C.J.C.H., Dayan, P.: Q-learning. *Machine Learning* 8, 279–292 (1992)

Perception and Developmental Learning of Affordances in Autonomous Robots

Lucas Paletta¹, Gerald Fritz¹, Florian Kintzler², Jörg Irran², and Georg Dorffner²

¹ Joanneum Research Forschungsgesellschaft mbH,
Institute of Digital Image Processing, Computational Perception Group,
Wastiangasse 6, Graz, Austria

² Österreichisches Forschungsinstitut für Artificial Intelligence (OFAI),
Neural Computation and Robotics, Freyung 6, Vienna, Austria

Abstract. Recently, the aspect of visual perception has been explored in the context of Gibson's concept of affordances [1] in various ways. We focus in this work on the importance of developmental learning and the perceptual cueing for an agent's anticipation of opportunities for interaction, in extension to functional views on visual feature representations. The concept for the incremental learning of abstract from basic affordances is presented in relation to learning of complex affordance features. In addition, the work proposes that the originally defined representational concept for the perception of affordances - in terms of using either motion or 3D cues - should be generalized towards using arbitrary visual feature representations. We demonstrate the learning of causal relations between visual cues and associated anticipated interactions by reinforcement learning of predictive perceptual states. We pursue a recently presented framework for cueing and recognition of affordance-based visual entities that obviously plays an important role in robot control architectures, in analogy to human perception. We experimentally verify the concept within a real world robot scenario by learning predictive visual cues using reinforcement signals, proving that features were selected for their relevance in predicting opportunities for interaction.

1 Introduction

The concept of affordances has been coined by J.J. Gibson in his seminal work on the ecological approach to visual perception [1]. In the context of ecological perception, visual perception would enable agents to experience in a direct way the opportunities for action. However, Gibson remained unclear about both how this concept could be used in a technical system and which representation to use. Neisser [2] replied to Gibson's concept of direct perception with the notion of a perception-action cycle that shows the reciprocal relationship of the knowledge (i.e., a schema) about the environment directing exploration of the environment (i.e., action), which samples the information available for pick up in the environment, which then modifies the knowledge, and so on. This cycle describes how knowledge, perception, action, and the environment all effectively interact in order to achieve goals.

Our work on affordance-like perception is in the context of technical, i.e., robotic systems, based on a notion of affordances that ‘*fulfill the purpose of efficient prediction of interaction opportunities*’. We extend Gibson’s ecological approach under acknowledgment of Neisser’s understanding that *purposive* visual feature representation on various hierarchies of abstraction are mandatory to appropriately respond to environmental stimuli. We take advantage of a refined concept of affordance perception by representing (i) an interaction component (*affordance recognition*: recognizing relevant events in interaction via perceptual entities) and (ii) a predictive aspect (*affordance cueing*: predicting interaction via perceptual entities). This conceptual step enables firstly to investigate the functional components of perception that make up affordance-based prediction, and secondly to lay a basis to identify the causal relation between predictive features and predicted event via machine learning technology.

The particular contribution of this work is to propose a novel framework for the developmental learning of affordances, and to frame the outline of basic affordances in terms of reinforcement learning. In this context, the work is in line with the concept to enable purposive - in particular, *affordance based* - perception which is consequently structured into cueing, behavior, and outcome related components. Learning is mandatory to enable agents to autonomously develop their characteristic embodied perception through interaction with the environment. Reinforcements guide the development through exploration without external supervision, and a theory for estimating delayed reward is the appropriate framework to extract early cues.

The outline of this paper is as follows. Section 2 describes the relevance of structured affordance-like representations in robot perception and argues for the importance to learn the features of perceptual entities. Section 3 describes the framework for developmental learning of affordances. Section 4 presents the concept of reinforcement learning of basic affordances, and how predictive features are extracted within Markov Decision Processes. Section 5 illustrates the experimental results that strongly support the proposed hypothesis on the relevance of generalized features that can be learned using reinforcement for successful affordance-like cueing in robot control systems. Section 6 concludes with an outlook on future work.

2 Computational Models on Affordances

Affordance-like perception aims at supporting control schemata for perception-action processing in the context of rapid and simplified access to agent-environment interactions. In this Section we argue for the relevance of learning in cue selection, and present a framework on the outline of components that enables to identify relevant visual features.

2.1 Related Work

Previous research on affordance-like perception focused on heuristic definitions of simple feature-function relations to facilitate sensor-motor associations in robotic agents. The MIT humanoid robot Cog was involved in object poking and proding experiments that investigate the emergence of affordance categories to choose actions

with the aim to make objects roll in a specific way [7]. The research of Stoytchev [8] analyzed affordances on an object level, investigating new concepts of object-hood in a sense of how perceptions of objects are connected with visual events that arise from action consequences related to the object itself. However, these experiments involve computer vision still on a low level, and do not consider complex sensor-motor representation of an agent interaction in less constrained, even natural environments. In the biologically motivated cognitive framework of Cos-Aguilera et al. [15], object based affordances are set in the context of motivation driven behavior selection. In contrast to our work, they do not learn visual feature extraction in a purposive manner (Section 2.2) but rather match sensory input with stored object features in a classical sense and then associate object identities with appropriate interaction patterns.

Affordance based visual object representations are per se function based representations. In contrast to classical object representations, functional object representations (Stark and Bowyer [9], Rivlin et al. [10]) use a set of primitives (relative orientation, stability, proximity, etc.) that define specific functional properties, essentially containing face and vertex information. These primitives are subsumed to define surfaces from the functional properties, such as '*is sit-able*' or '*provides stable support*'. However, so far function based representations were basically defined by the engineer, while - in contrast - it is particularly important in affordance based recognition to *learn* the structure and the features themselves *from experience* (Section 4).

2.2 Affordance Based Perception and Learning of Affordances

Fig. 1 depicts the concept of feature based affordance perception as outlined in detail in [17,22]. We first identify the component of *affordance recognition*, i.e., the recognition of the affordance related visual event that causally anticipates a relevant interaction, e.g., the capability of lifting (*lift-ability*) an object using an appropriate robotic actuator. The recognition of this event should be performed in identifying a process of evaluating spatio-temporal information that leads to an outcome entity. This outcome entity should be unique in perceptual feature/state space, i.e., it should be characterized by the observation of specific feature attributes that are abstracted from the stream of sensory-motor information.

The second functional component of *affordance cueing* encompasses the key idea on affordance based perception, i.e., anticipating the opportunity for interaction from causally relevant features, i.e., the *predictive features*, that can be extracted from the incoming sensory processing stream. In particular, this component is embedded in the perception-action cycle of the robotic agent. The agent is receiving sensory information in order to build upon arbitrary levels of feature abstractions, for the purpose of recognition of perceptual entities. In contrast to classical feature and object recognition, this kind of recognition is *purposive* in the sense of selecting exactly those features that efficiently support the evaluation of identifying an affordance, i.e., the perceptual entities that possess the capability to predict an event of affordance recognition in the feature time series that is immediately following the cueing stage of affordance based perception. The outcome of affordance cueing is in general a probability distribution P_A on all possible affordances (Section 4.1), providing evidence for a most confident affordance cue by delivering a hypothesis

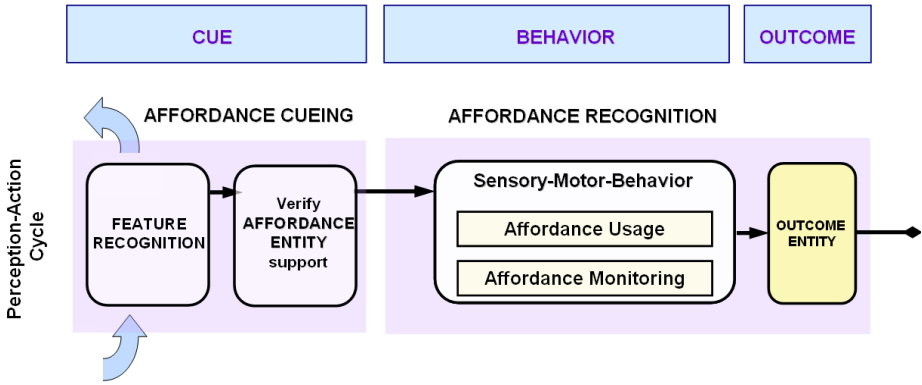


Fig. 1. Concept of affordance perception, depicting the key components of affordance cueing and recognition embedded within an agent’s perception-action cycle (most left). While affordance cueing (left) provides a prediction on future opportunities of interaction on the basis of related predictive features, affordance recognition (right) identifies the convergence of a sensory-motor behavior towards the identified outcome of the overall interaction pattern.

that favors the future occurrence of a particular affordance recognition event. This cue is *functional* in the sense of *associating* to the related feature representation a specific *utility* with respect to the capabilities of the agent and the opportunities provided by the environment, thus representing *predictive features* within the affordance based perception system. An overall consistent formal theory on affordances, describing agents with the capability to perceive functionalities for interaction, has been proposed by Doherty et al [24].

In contrast to previous work on functional feature and object representations [9,10], we stress the fact that functional representations must necessarily contain *purposive features*, i.e., represent perceptual entities that refer to interaction patterns and thus must be selected from an existing pool of generic feature representations. Feature selection (and, in a more general sense, feature extraction) must be performed in a machine learning process and therefore avoid heuristic engineering which is always rooted in a human kind understanding of the underlying process, a methodology which is necessarily both, firstly, error prone due to failing insight into statistical dependencies and, secondly, highly impractical for autonomous mobile systems. Recent work on the learning of affordance features has focused on methodologies to estimate direct mappings between cues and actions from experience [17,23]; in the presented work we motivate from a developmental point of view (Section 3) and outline a mathematical framework to extract affordance cues from arbitrary action sequences, i.e., from delayed rewards.

3 Developmental Learning of Affordances

An agent embedded in its habitat is able to perceive the environment with its sensors and is able to move and manipulate this environment with its actuators. A structure enabling the robot to act on its perceptions by using its actuators, is called control

architecture. In case that the architecture causes actions, depending on the perceived state of the environment, a closed loop control emerges. The design of this control is essential for enabling the robot to use affordances, the proposed approach uses principles from the reactive control approach as well as from the subsumption based approach.

ABACUS. We present ABACUS (Affordance Based Adaptive Control Using Self-Experience; Fig. 2 [22]), a multi layered conceptual framework, which enables the robot to use the concepts of affordances by taking it through several learning stages. In a first phase, *Phase 0*, the robot is starting with pure reactive behavior, proceeding to *Phase 1* which deals with learning affordances through basic interaction (*basic affordances*), and *Phase 2* on the learning of affordances through action sequences (*complex affordances*), furthermore, to the *Final Phase* where the robot is able to use the affordances it gained so far for planning of goal driven behavior.

The fundamental functioning of the developmental learning is as follows. In *Phase 0*, a control layer, implementing reactive behavior, is added to the structures named sensor layer, filter layer, and actuator layer, and thus a basic reactive control is built. The reactive control of Phase 0 is then refined in Phase 1, where an adaptive structure learns to perceive and use basic affordances that are directly related to single action possibilities of the agent (e.g. gripping an object) and are thus mostly related to the object as a whole. The control developed in phase 1 is refined by Phase 2. Phase 2 is designed to learn to perceive and use more complex affordances that are related to sequences of actions (e.g. stacking, which consists of several interactions with 2 objects like gripping, lifting, driving and releasing) and are mostly related to object parts (e.g. a flat surface). The enhancement and refinement of each Phase n through a succeeding Phase $n+1$ form a subsumption like affordance based control. The final scenario is realized by incrementally extending the Phase structures to gain an incrementally more complex affordance based architecture.

Phase 0. The sensor layer consists of physical sensors and software modules that are interfaces between software and hardware to enable the agent to receive raw data about the state of the robots environment, and the state of the robot itself. The filter layer is designed to reduce computing complexity and fault sensitivity within the control layer. Instead of using the original time series from the cameras, the laser-scanner or the positions of the robotic arm, the control layer can use more complex data extracted from the sensoric input space, e.g. by filters for detecting simple geometric forms like ellipsoides or rectangles, or complex SIFT filters (Section 5). Other examples for modules within the filter layer are simple bandpass-filters, motion detectors or more complex novelty-detectors. The filter modules can also be cascaded, so that a hierarchy of filters is formed. Hence careful design of these filters is imperative for achieving the desired reduction of computing complexity without losing essential information. Within the filter modules attention mechanisms can be used to reduce the amount of data transferred to the next layer. By cascading the filters, attention mechanisms can also be applied to sets of filters, e.g. to focus attention to one sensor modality or one special filter.

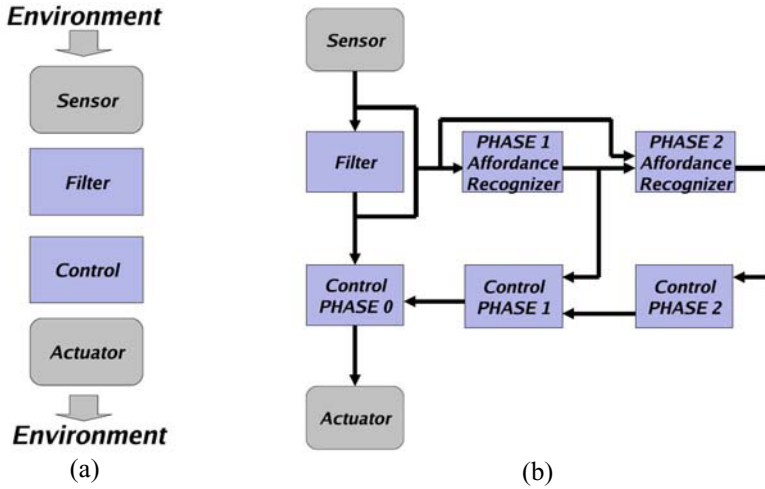


Fig. 2. (a) Sketch of the basic control structure underlying ABACUS (Affordance Based Adaptive Control Using Self-Experience). (b) ABACUS is a multi layered conceptual framework, which enables the robot to use the concepts of affordances by taking it through several learning stages. Phase 2 of ABACUS is designed to learn to perceive and use complex affordances by using the information received from the filter layer and the phase 1 affordance recognizer. Complex affordances are related to sequences of actions (e.g. stacking, which consists of several interactions with 2 objects like gripping, lifting, driving and releasing) and are mostly related to object parts (e.g. a flat surface enables the robot to stack something on it).

Phase 1. Phase 1 is a structure that is designed to learn to perceive and use basic affordances that are directly related to single action possibilities of the agent (e.g. gripping an object) and are thus mostly related to the object as a whole. These affordances will also be called category 1 affordances. Phases 1 utilizes the described sensor layer, filter layer, and actuator layer and refines the perception and control of phase 0. Phase 1 consists of an affordance recognizer and an affordance based control module (see Fig. 6). The Phase 1 affordance recognizer learns to recognize basic affordances on the basis of the output of the filter layer and sensor layer. The module learns what the outcome of an action is and learns what the cues to detect affordances without interacting with its environment are. For a detailed description of an algorithm that can be used to realize the adaptive Phase 1 affordance recognizer see section 4. The information, extracted by the affordance recognizer is used within the Phase 1 affordance based control to trigger Phase 1 actions, or trigger or inhibit the basic actions that are implemented within the modules of the Phase 0 control layer.

Phase 2. Phase 2 is a structure that is designed to learn to perceive and use complex affordances that are related to sequences of actions (e.g. stacking, which consists of several interactions with two objects like gripping, lifting, driving and releasing) and are mostly related to object parts (e.g. a flat surface enables the robot to stack something on it). These complex affordances are also called category 2 affordances. Phase 2 extends the affordance recognition capabilities of Phase 1 and refines the

affordance based control of Phase 1 and Phase 0. Like Phase 1, Phase 2 is subdivided in an affordance recognizer and an affordance based control module. By using data from the filter layer and the Phase 1 affordance perception output as an abstract and highly complex sensor, the Phase 2 affordance recognizer is enabled to learn complex affordances that require basic affordances to be present. The main focus of phase 2 lies on action sequences, e.g. behaviors lifting, stacking, or turning objects etc. that can be composed out of simple basic behaviors or motion primitives. Like in the case of basic actions, the outcome of sequenced actions is categorized and perceptual cues are searched that indicate the presence of affordances. For a detailed description of a possible learning algorithm to realize the Phase 2 affordance recognizer modules see Section 4. The information extracted by the affordance recognizer is used within the Phase 2 affordance based control to revise Phase 0 control and Phase 1 control by triggering and inhibiting actions and action sequences as well as triggering Phase 2 actions.

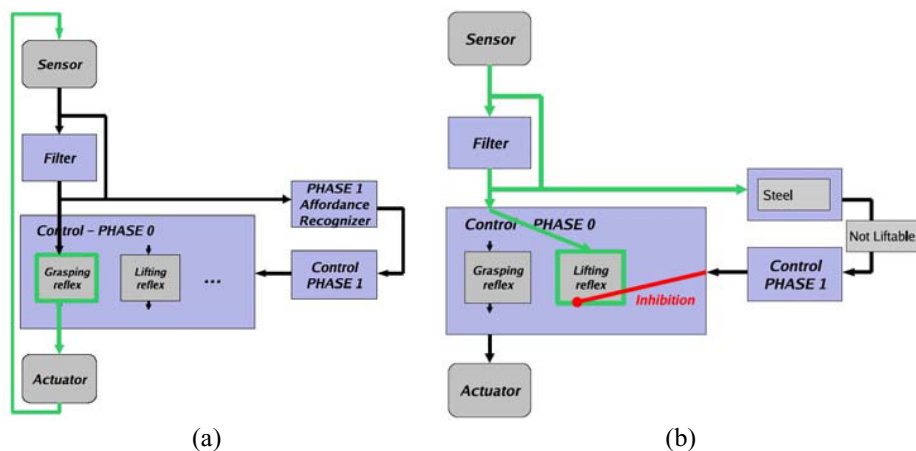


Fig. 3. Example for the revision of Phase 0 based control by Phase 1 based control. (a) shows the general structure of a Phase 1 instantiation. The basic actions within control phase 0 are activated by environmental triggers, e.g., of the grasping reflex. The reflex like action of the agent changes the state of the agent and its environment; this change leads to a change within the sensor data measured by the agent. The new state of the environment/state of the robot could trigger another action, e.g. the state of javing an object in the gripper could cause a lifting reflex. Once the Phase 1 affordance recognizer has learned to discriminate *liftable* from *non-liftable* environmental entities, the Phase 1 based control can inhibit the lifting reflex (b), and in a later stage even inhibit all actions leading to the inhibited action.

4 Reinforcement Learning of Basic Affordances

4.1 Affordance Based Cueing

Early awareness of opportunities for interaction is highly relevant for autonomous robotic systems. Visual features are one out of multiple modalities from sensory

processing that operate perception via optical rays and therefore support early awareness about the environment of a robot agent from rather remote locations. Although the necessity of affordance perception from 3D information recovery, such as optical flow, has been stressed in previous work, we do not restrict ourselves to any specific cue modality and intend to generalize towards the use of arbitrary features (2D, 3D) that can be derived from visual information with the only constraint that they enable reliable prediction of the opportunity for interaction processes from an early point in time.

Scenario. The scenario for the experiments consists of a mobile robotic system (Kurt2 from Fraunhofer IAIS, Germany [17]), equipped with a camera stereo pair and a magnetizing effector, and some can-like objects with various top surfaces, colors and shapes. The purpose of the magnetizing effector is to prove the nature of the individual objects by lowering its rope-end effector down to the top surface of the object, trying to magnetize the object (only can bodies are magnetizable) and then to lift the object. Test objects with well magnetizable geometry (with slab like top surfaces, in contrast to those with spherical top surface) are subject to a lifting interaction, while the others are not able to be lifted from the ground. This interaction process is visualized for several test objects and sampled in a sequence of image frames which are referenced with multimodal sensor information, e.g., size of magnetizing and motor current of the robot.

Visual Features. From the viewpoint of a technical system using computer vision for digital image interpretation, we selected local descriptors, such as the Scale Invariant Feature Transform [13], to support well the generation of visual feature abstractions. We first segment the color based image information and then associate classified histograms of descriptor responses - sampled within the regions - to the region feature vector. The histograms integrate responses from SIFT descriptors that were trained to discriminate either rectangular or circular surface shapes [17].

Affordance Hypotheses. The outcome of the affordance cueing system is in general expected to be – given a perceptual entity in the form of a multimodal feature vector - a probability distribution over affordance hypotheses,

$$P_{A_i} = P(A_i | F_t),$$

with affordance hypothesis A_i , and feature vector F_t at time t . It is then appropriate to select an affordance hypothesis $A_{max} = \arg \max_i (P(A_i))$, with Maximum A Posteriori (MAP) confidence support for further processing.

Cue-Feature Value Matrix. Fig. 4 shows a sample cue-feature value matrix in the context of the experiments - depicting attribute values of 2D features (color G=green, R=red, M=magenta, etc.), or SIFT category (R=rectangular, C=circular, etc.) and interaction results (left column, bottom) in dependence on various types of visual regions (top row) - that visualizes dependencies between feature attributes of the region information and a potential association to results of the affordance recognition process. We can easily see that the SIFT category information (rectangular=R and circular=C region characterization) together with a geometric feature (top=T region, i.e., representing a region that is located on top of another region) provides the

discriminative feature that would allow to predict the future outcome (e.g., lift-able or non lift-able) of the affordance recognizer. The latter therefore represents the identification of the affordance and thereby the nature of the interaction process (and its outcome entity) itself.

| | G | R | M | R | Y | B | Bl | Gr |
|---------------|---|---|---|---|---|---|----|----|
| colour | G | R | M | R | Y | B | Bl | Gr |
| SIFT category | R | R | C | C | R | R | R | N |
| shape L/W | L | L | L | L | P | P | P | L |
| T/B | T | T | T | T | B | B | B | N |
| LIFTABLE | Y | Y | N | N | Y | Y | N | N |
| NOT LIFTABLE | N | N | Y | Y | Y | Y | Y | N |

Fig. 4. Cue-feature value matrix depicting attribute values of 2D features (color G/green, R/red, M/magenta, etc., or SIFT category R/rectangular, C/circular, etc.) and interaction results (left column, bottom) in dependence on various types of visual regions (top row). From this we conclude a suitable feature value configuration (i.e., SIFT categories to discriminate *lift-able/non lift-able* predictions) to support the hypothesis on *lift-able* object information.

4.2 Reinforcement Learning

In the following, we describe an implementation of developing control Phase 1 (Section 3) of affordance learning, by encapsulating Phase 0 in terms of a reactive behavior that is modeled within the reinforcement learning of the overall affordance *liftability*.

Markov decision processes [18] have already been introduced in a perception-action context of visual recognition, e.g., in [19], selecting foci of attention for optimal integration of visual information in sequential object recognition, or in sequential object recognition [20]. In this paper, the MDP will provide the general framework to outline a multi-step behavioral task under the viewpoint of state based prediction, i.e., cueing, of future outcomes of that task. Fig. 5b shows a schematic outline of closed-loop learning of the behavioral task within the robot scenario, together with the extraction of early cues (feature recognition) from a selection of relevant attributes.

Markov Decision Processes. An MDP is defined by a tuple $(S;A;\delta;\mathfrak{R})$ with state recognition set S , action set A , probabilistic transition function δ , and reward function $\mathfrak{R}: S \times A \rightarrow \Pi(S)$ describes a probability distribution over subsequent states, given action $a \in A$ executable in state $s \in S$. In each transition, the agent receives reward according to $\mathfrak{R}: S \times A \rightarrow \mathbb{R}$, $\mathfrak{R}_t \in \mathbb{R}$. In our experimental scenario, the agent must act to maximize the utility $Q(s,a)$, i.e., the expected discounted reward

$$Q(s, a) \equiv U(S, a) = E \left[\sum_{n=0}^{\infty} \gamma^n R_{t+n}(s_{t+n}, a_{t+n}) \right]$$

where $\gamma \in [0,1]$ is a constant controlling contributions of delayed reward. We formalize a sequence of action selections a_1, a_2, \dots, a_n as an MDP and are searching

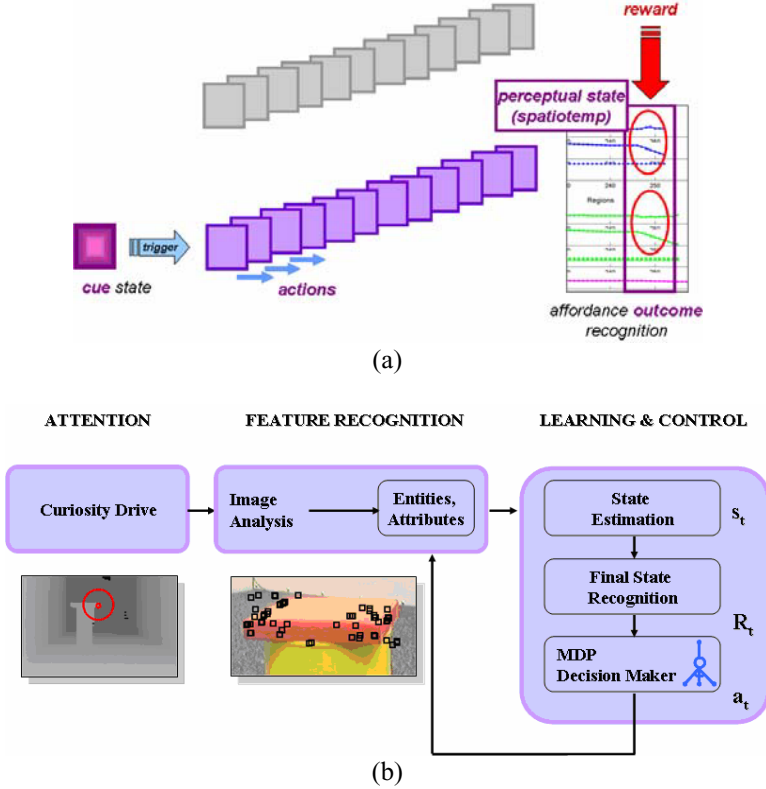


Fig. 5. Markov Decision Process (MDP): (a) Perceptual states that anticipate the outcome (effect) status are traced back via perception-action trajectories. (b) Closed-loop processing in affordance-based feature recognition. On the basis of attentive image segmentation (curiosity drive, in 2D or 3D), feature entities are recognized, then build up a perceptual state which feeds into the decision maker. Perceptual states may anticipate different trajectories in state space; highly rewarded states represent *cues* for anticipating targeted outcome states (events).

for optimal solutions with respect to finding action selections so as to maximizing future reward with respect to the affordance task. With each action, an estimate on the cumulative reward gives feedback about the direction towards the goal of the task. With each action, the reward is received per action by $R(s, a) := \Omega$, with $\Omega=1$ if the goal event is reached (object lifted into goal image zone), and $\Omega=0$ if not (Fig. 8). Since the probabilistic transition function $\Pi(\cdot)$ cannot be known beforehand, the

probabilistic model of the task is estimated via reinforcement learning, e.g., by Q-learning [20] which guarantees convergence to an optimal policy applying sufficient updates of the Q-function $Q(s; a)$, mapping recognition states s and actions a to utility values. The Q-function update rule is

$$Q(s, a) \equiv (1 - \alpha)Q(s, a) + \alpha \left[R + \gamma \left(\max_{a'} Q(s', a') \right) \right]$$

where α is the learning rate, γ controls the impact of an action on future policy returns. The decision process is determined by the sequence of actions. The agent selects then the action with largest $Q(s, a)$, i.e.,

$$a_T = \arg \max_{a'} Q(s_T, a')$$

so as to maximize the cumulative expected reward $Q(s, a)$.

Actions and States. In the selected scenario, actions are defined by discrete steps of gripper motion (up, down) and magnetization (on, off). Each perceptual state is defined by a discrete feature configuration

$$S = (c \in \{C_i\}, d \in \{D_{circ}, D_{rect}\}, e \in \{0, 1\}, h_r \in \{r_j\}, h_g \in \{r_j\}, m \in \{0, 1\}),$$

with region color class c , region shape d , configuration type e , region elevation level h_r , gripper elevation level h_g , and magnet state m .

The affordance based, purposive selection of features is here represented by relevance weighting of perceptual states in terms of associated expected rewards. Each extracted image region is attributed by a perceptual state. Perceptual states that anticipate a complete trajectory of perception-action transition leading to the affordance outcome event with high certainty will be associated to high rewards and consequently constitute cue like states. Affordance based perceptual states are intrinsically related to the reward function, an associated estimator on affordance hypotheses P_{Ai} (see above), and the affordance classifier that discriminates corresponding Q -values into affordance cues and irrelevant states.

5 Proof of Concept

The experiments were performed in a real world robot environment with the purpose of providing a proof of concept on the successful learning of predictive 2D features, i.e., affordance based cues, and on characterizing affordance recognition processes.

Scenario. Robot operations are discriminated into two phases (a) a cueing phase where the robot is moving to the object, and (b) a recognition phase, where the robot tries to lift an object (Fig. 1). In both phases, parts of the objects are described by their regions. Any region has different features like color, center of mass, top/bottom location and the shape description (rectangular, circular) already described above, the features are extracted from the robot camera imagery. Additional information, such as, effector position, are provided by the robot. Regions are the entities used in the experiments, no explicit object model is generated for the can-like objects.

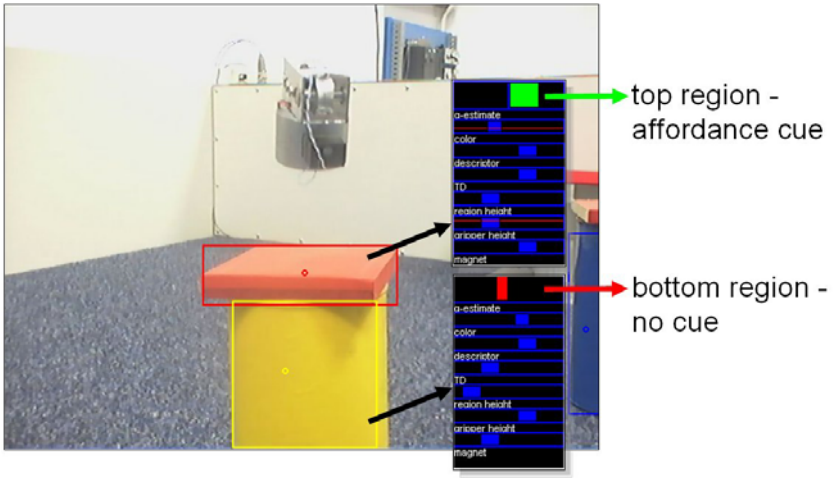
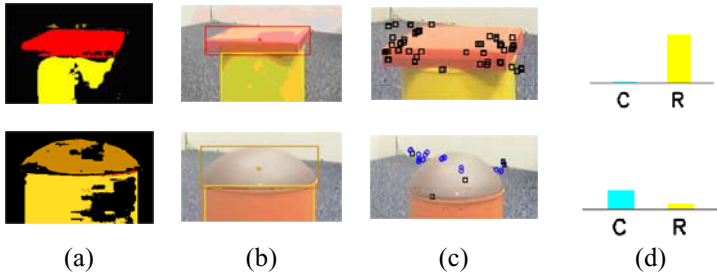


Fig. 6. Affordance based cueing of region determined perceptual states from learned predictive cumulative rewards. (a) Color segmentation, (b) color region bounding boxes, (c) classified SIFTs, (d) histogram on rectangular (R) and circular (C) descriptors. (e) Analyzed top and bottom regions are correspondingly classified as cues for lift-ability or non-lift-ability, visualized in terms of green and red bars with bar sizes correlating to positive or negative reward, respectively (monitoring boxes, top), anticipating a lift-able event.

Affordance Recognition and Cueing. The recognition of an affordance is crucial for verifying a hypothesis about an affordance A associated with an entity feature F . These entities are specifically extracted out of the images as follows. Firstly, a watershed algorithm is used to segment regions of similar color together. After merging of smaller parts, every entity is represented by the average color value, the position in the image and the relation to adjacent regions (top/bottom). This information is also used for tracking entities over time. To verify whether or not an entity becomes ‘lift-able’, the magnetizable effector of the robot is lowered until the top region of the object under investigation is reached, the magnet is switched on and the effector is lifted up. If the entity is lift-able, a common motion between effector and region can be recognized, and both can and gripper regions are undergoing a vertical transition (direction up) in the field of view. Additionally the magnet has to

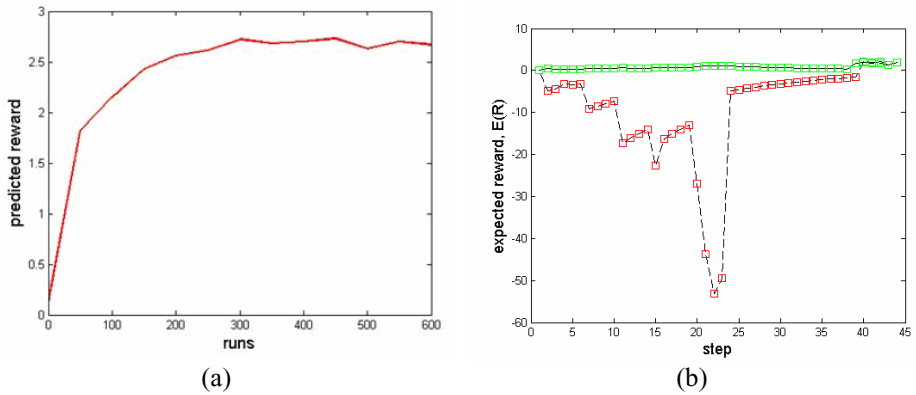


Fig. 7. (a) Reinforcement learning curve depicting increase of predicted cumulative reward associated to a predictive perceptual state over time, verifying its statistically correct anticipation of successive ‘lift’ events. (b) Step-wise classification of regions of interest under predictive cueing (green=cue, red=no cue). In the first phase, the gripper moves down to grip the object. The downward peak is at the turning point where the gripper lifts up the object.

be switched on and the effector has to be placed in the center of the top region. These rules build up the affordance recognizer looking for lift-able entities in the recognition phase of the experiment.

Reinforcement Learning of Predictive Features. Affordance cueing and recognition may require different kinds of feature extraction. For cueing, some structural description of the top region is required to separate the unequal shape of the top regions. In order to get structural information about an entity a histogram over prototypical SIFT descriptors is used to discriminate between circular and rectangular regions.

Structural Classification. All local SIFT descriptors extracted in the region of the entities are clustered using unsupervised clustering (*k-means*, $k=100$). For each specific entity, we generate a histogram over cluster prototypes, using a nearest neighbor (NN) approach to get the cluster label for each SIFT descriptor in that region. In a supervised learning step, every histogram is labeled whether it is or not associated with a rectangular or circular entity. A C4.5 decision tree [14] of size 27 is then able to distinguish between these two classes. The error rate on a test set with 353 samples is $\approx 1.4\%$.

Q-learning, decisive states, and affordance based cueing. Images about the objects that were tested for the affordance ‘lift-able’ in the recognition phase collect positive rewards that trace back to early perceptual states due to the Q-learning update rule. As mentioned earlier, there exists no object model yet, therefore only entities exist for the system, and the learning of cueing states is with respect to the region extraction determining the perceptual states. In our experiment 30 frames are used from the beginning of the affordance recognition back, that means a recall of ~ 2.5 seconds from the past (12 fps are captured by the robot during the experiment). The entity

representation for the cueing phase contains the following features: (a) average color value of the region in the image, (b) top/bottom information, (c) the result of the structure classification, (d) the size of the segmented region. Fig. 7a depicts the learning curve resulting from the reinforcement learning phase, with respect to the predicted cumulative reward associated to an early perceptual state that thereby is verified to represent a ‘cueing’ state. Fig. 6e depicts results for predicted cumulative rewards regarding observed regions (top, bottom) reflecting different evaluation of perceptual states towards ‘cueing’ (green bar) and ‘non-cueing’ (red bar) states. Fig. 6b shows the dynamic view on step-wise classification of regions under predictive cueing, e.g., predictive features (green) were detected all through the sequence of 45 successive frames.

Extraction of Cue Features. The experimental result that proves that the reinforcement learner identified the actually relevant features was finally received from a statistical analysis of perceptual states. A C4.5 decision tree [14] was learned to estimate the decisive attributes that enable classification of perceptual states into affordance cues ($Q(s,a) > 0$) and irrelevant states ($Q(s,a) \leq 0$). Features found were $CR < 0.5$ (‘rectangular’) and $TD < 0.5$ (‘top region’) which exactly defines the decisive features as outlined in the affordance cue matrix depicted in Fig. 4 (‘SIFT category’ and ‘TB’).

6 Conclusions

This work presented the framework of reinforcement learning for perceptual cueing to opportunities for interaction of robotic agents. The framework for cueing and recognition of affordance-like visual entities is verified with a concrete implementation using state-of-the-art visual descriptors on a real world robot scenario and proved that features are successfully selected that are relevant for prediction towards affordance-like control in interaction. The real world robot environment was chosen to enable a proof of concept in terms of learning to select exactly those features that are relevant for the prediction of the interaction outcome.

Future work will focus on extending the feature based representations towards object driven affordance-based interaction, grounding the work on the visual descriptor information presented here, and demonstrating the generality of the concept. In this line of research, we think that the presented reinforcement learning provides the appropriate methodology to motivate the learning of functional object recognition, grounding thereby the object notion in a concept of predictive feature abstractions.

Acknowledgments

This work is funded by the European Commission’s projects MACS (FP6-004381) and by the FWF Austrian National Research Network ‘Cognitive Vision’ under sub-project S9104-N04.

References

- [1] Gibson, J.J.: *The Ecological Approach to Visual Perception*, Boston, Houghton Mifflin (1979)
- [2] Neisser, U.: *Cognition and Reality*. In: *Principles and Implications of Cognitive Psychology*, Freeman & Co., San Francisco (1976)
- [3] Gibson, E.J.: Exploratory behavior in the development of perceiving, acting and the acquiring of knowledge. *Annual Review of Psychology* 39, 1–41 (1988)
- [4] Faillenot, I., Toni, I., Decety, J., Grégoire, M.-C., Jeannerod, M.: Visual pathways for object-oriented action and object recognition: functional anatomy with PET. *Cerebral Cortex* 7, 77–85 (1997)
- [5] Fagg, A.H., Arbib, M.A.: Modeling parietal-premotor interaction in primate control of grasping. *Neural Networks* 11(7-8), 1277–1303 (1998)
- [6] Wheeler, S.D., Fagg, H.A., Grupen, R.A.: Learning Prospective Pick and Place Behavior. In: *Proc. 2nd International Conference on Development and Learning*, June 2002, pp. 197–202. IEEE Computer Society, Cambridge, MA (2002)
- [7] Paul, F., Metta, G., Natale, L., Rao, S., Sandini, G.: Learning About Objects Through Action - Initial Steps Towards Artificial Cognition. In: *Proc. IEEE International Conference on Robotics and Automation, ICRA 2003, Taipei, Taiwan, May 12-17 (2003)*
- [8] Stoytchev, A.: Behavior-Grounded Representation of Tool Affordances. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, April 18-22, 2005, Barcelona, Spain (2005)
- [9] Stark, L., Bowyer, K.W.: Function-based recognition for multiple object categories. *Image Understanding* 59(10), 1–21
- [10] Rivlin, E., Dickinson, S.J., Rosenfeld, A.: Recognition by functional parts. *Computer Vision and Image Understanding* 62, 64–176 (1995)
- [11] Bogoni, L., Bajcsy, R.: Interactive Recognition and Representation of Functionality. *Computer Vision and Image Understanding* 62(2), 194–214 (1995)
- [12] Edwards, M.G., Humphreys, G.W., Castiello, U.: Motor facilitation following action observation: a behavioural study in prehensile action. *Brain Cognition* 53, 495–502 (2003)
- [13] Lowe, D.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2), 91–110 (2004)
- [14] Quinlan, J.R.: *C4.5 Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA (1993)
- [15] Cos-Aguilera, I., Cañamero, L., Hayes, G.M., Gillies, A.: Ecological integration of affordances and drives for behaviour selection. In: Bryson, J., et al. (eds.) *Proc. Workshop on Modeling Natural Action Selection*, pp. 225–228. AISB Press (2005)
- [16] Cos-Aguilera, I., Cañamero, L., Hayes, G.M.: Using a SOFM to learn Object Affordances. In: Cos-Aguilera, I. (ed.) *Proc. Workshop of Physical Agents, WAF'04, March 2004, Girona, Catalonia, Spain (2004)*
- [17] Fritz, G., Paletta, L., Kumar, M., Dorffner, G., Breithaupt, R., Rome, E.: Visual Learning of Affordance based Cues. In: Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J.C.T., Marocco, D., Meyer, J.-A., Miglino, O., Parisi, D. (eds.) *SAB 2006. LNCS (LNAI)*, vol. 4095, pp. 25–29. Springer, Heidelberg (2006)
- [18] Puterman, M.: *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, New York (1994)

- [19] Paletta, L., Fritz, G., Seifert, C.: Q-Learning of Sequential Attention for Visual Object Recognition from Informative Local Descriptors. In: Proc. 22nd International Conference on Machine Learning, ICML 2005, Bonn, Germany, August 7-11, 2005, pp. 649–656 (2005)
- [20] Draper, B.A.: Modeling Object Recognition as a Markov Decision Process. In: Proc. 13th International Conference on Pattern Recognition 4, 95
- [21] Watkins, C., Dayan, P.: Q-learning. *Machine Learning* 8, 279–292 (1992)
- [22] Irran, J., Kintzler, F., Pözl, P.: Grounding Affordances. In: Trapp, R. (ed.) *Cybernetics and Systems*. Austrian Society for Cybernetic Studies, Vienna (2006)
- [23] Ugur, E., Dogar, M.R., Cakmak, M., Sahin, E.: The learning and use of traversability affordance using range images on a mobile robot. In: Proc. Internat. Conference on Robotics and Automation, ICRA 2007, pp. 1721–1726 (2007)
- [24] Doherty, P., Merz, T., Rudol, P., Wzorek, M.: Tentative proposal for a formal theory of affordances. Technical Report MACS/4/2.1, Linköpings Universitet, IDA Group, Linköping, Sweden (August 2005)

A Computational Model of Bistable Perception-Attention Dynamics with Long Range Correlations

Norbert Fürstenau

German Aerospace Center, Institute of Flight Guidance, Lilienthalplatz 7
D-38108 Braunschweig, Germany
norbert.fuerstenau@dlr.de

Simulation results of bistable perception due to ambiguous visual stimuli are presented which are obtained with a nonlinear dynamics model using perception–attention–memory coupling. Percept reversals are induced by attention fatigue and noise, with an attention bias which balances the relative percept duration. The dynamics of the attention parameter exhibits qualitative agreement with the eye blink rate variation [4]. Coupling of an attention bias to the perception state introduces memory effects leading to significant long range correlations of perceptual duration times as quantified by the Hurst parameter ($H > 0.5$). This prediction is in agreement with recent experimental results [1]. Deviations of the reversal time statistics from the Γ -distribution increase with decreasing memory time constant and attention noise. Mean perceptual duration times of 2 – 5 s are predicted in agreement with experimental results [7] if a feedback delay of ca. 40 ms is assumed which is typical for cortical reentrant loops.

Keywords: cognitive bistability, modelling, nonlinear dynamics, perception, attention, Hurst parameter.

1 Introduction

In the present work new simulation results of an extended version of a nonlinear dynamics model of cognitive multistability [3] are presented. They specifically predict long range correlations of the perceptual duration times which have recently been found experimentally by Gao et al. [1] via determination of the self similarity (Hurst) parameter $H (> 0.5)$ [2] of the reversal time series.

Bistable perception is the spontaneous involuntary switching of conscious awareness between the different percepts of an ambiguous stimulus. It is excited with different methods and stimuli such as binocular rivalry [5], perspective reversal, e.g. with the famous Necker cube [6][7], and ambiguous motion displays as induced by moving groups of crossed lines (plaids) [8]. Bistability provides a unique approach to fundamental questions of perception and consciousness because it allows for the direct measurement of the switching of subjective perception under constant external stimulus (e.g. [9][10][11][12] [13]).

The basic model couples the dynamics of a macroscopic (behavioral) perception state order parameter with an adaptive attention (feedback gain) control parameter with additive noise [3]. Memory and learning effects are introduced by allowing for the adaptation of the originally constant attention bias parameter which balances the subjective preference of one of the two percepts. Determination of the Hurst parameter of perception reversal time series with two different methods [1][14] yields significant long range correlations when coupling the bias parameter to the perception state for simulating short term memory.

Concerning theoretical modeling there are ongoing discussions on the predominance of a stochastic [15][16][38] versus deterministic [3] [17][18][39] background of multistability, on the importance of neural or attentional fatigue or adaptation [6][18] versus memory effects [1], and on the dominance of bottom-up [32] versus top-down [3][10][11] processing. The combined deterministic-stochastic approach of the present model is comparable to the synergetic model of Ditzinger & Haken [18]. The latter is based on two separate coupled nonlinear dynamics (polynomial) equations for the perception state order parameters. According to the experimentally supported satiation (neuronal fatigue) hypothesis [6], spontaneous transitions between different attractor states of the perception order parameters are induced by a slow time variation of associated attention (control) parameters due to perception–attention coupling. Recently published experimental results of Nakatani et. al. [24] support the perception–attention coupling approach. On a microscopic neural basis the attention fatigue corresponds to the well known spike rate adaptation in short term memory networks used for modeling the deterministic limit cycle oscillation of perception reversals and rivalry [39]. By including an additive attention noise term the present model like [18] explains the experimental finding that deterministic as well as stochastic dynamics determines the measured reversal time statistics for different multistability phenomena [37].

Following ideas proposed by von der Malsburg [40] the perception state P is assumed to arise from superimposed coherent fields of synchronously firing neuronal assemblies as excited by the ambiguous stimulus. In agreement with the widely accepted view of reentrant synchronous interactions between distant neuronal groups within the thalamo-cortical system leading to conscious perception (e.g. [10][20][21][24]), phase feedback of the superimposed coherent fields determines the multistable perception dynamics. Like [18] the model utilizes perception-attention coupling, however within a delayed reentrant loop and attention identified with adaptive feedback gain [3][27]. In contrast to [18] an adaptive attention bias balances the preference between percepts via learning and memory. In this kind of minimum reentrant model P is represented by the dynamic phase difference between the superimposed fields with a recursive cosinoidal mapping function. The approach is also motivated by the mean field phase oscillator theory of coupled neuronal columns in the visual cortex [22]. It describes the synchronization of neuronal oscillations as the physiological basis of dynamic temporal binding which in turn is thought to be crucial for the selection of perceptually or behaviorally relevant information [10][11][12]. Within the present model the difference of the perceptual duration time statistics between binocular rivalry and perception reversal with regard to the stochastic and deterministic character [37] can be explained via different memory time constants of the dynamic attention bias which determines the self similarity (Hurst) parameter.

In section 2 I describe the theoretical approach. Results of computer experiments with simulated perception time series and a statistical analysis of the reversal time series are presented in section 3, followed by a conclusion in section 4.

2 Theory

2.1 The Recursive Interference Model

As a kind of minimum architecture allowing for the emergence of discontinuous state transitions, a reentrant perception-attention dynamics with attention fatigue [6][18] and delayed phase feedback interference is employed [3][27]. Here I will motivate this architecture by a closer look at the thalamo-cortical reentrant loops as proposed within the dynamical core hypothesis of consciousness [13][21] and within the discussion of bottom-up and top-down aspects of visual attention [25]. Figure 1 depicts within a block diagram important modules of the attentionally modulated visual perception system. The diagram is based on simplified brain circuit schematics (e.g. [26]) including the attentional top-down modulation of the dorsal ("where") and ventral ("what") streams of information[25].

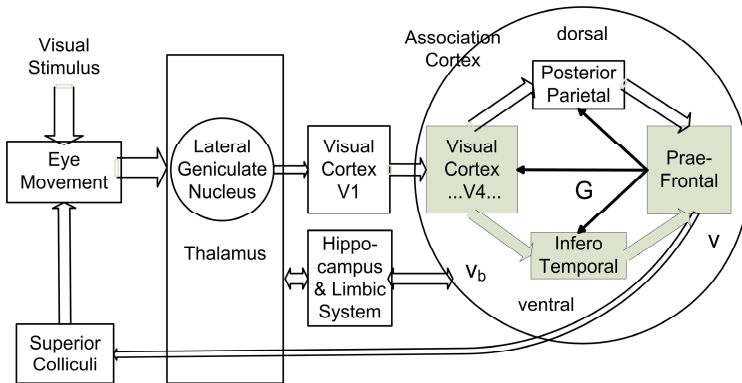


Fig. 1. Schematic of visual information flow within the thalamo-cortical system, with indication of bottom-up streams and attentional top-down modulation (black arrows) of ventral (what) and dorsal ("where") pathways within the association cortex (based on [25][26]). Perception state order parameter v , attention control parameter G , and attention bias or preference parameter v_b are placed at neurobiologically relevant positions. For details see text.

It is assumed that under bistable switching between conscious percepts, after feed-forward preprocessing of the stimulus up to the Primary Visual Cortex V1, the main processing takes place within recurrent perception-attention loops of the association cortex (e.g. [9][20][21][25]). For the stationary stimuli under consideration here the loop via the Superior Colliculi for control of eye movements, i.e. overt attention is neglected. On the other hand, recent experimental results indicate that binocular rivalry involves a more automatic, stimulus-driven form of competition than ambiguous figure reversal and is less easily biased by selective attention [41]. Without

considering early feedforward processing, in the present approach this effect may be modeled by the dynamic bias parameter v_b which determines relative percept duration times [3] and long range correlations via the v_b -memory time constant τ_M (see below). The model architecture is suggested to basically represent the ventral ("what") V4–InferoTemporal–PraeFrontal–V4 loop as target structure. Consequently it favors a top-down view assuming the usage of prior knowledge according to [42], and for simplicity neglects possible early spatial attention modulation of perception reversals as indicated by recent neurophysiological measurements [32]. Experimental evidence on perception–attention coupling with ambiguous stimuli was presented by Nakatani & van Leeuwen [29] using EEG recording of frontal theta and occipital alpha bands and eye blink rate measurement [4]. According to Hillyard et.al. [28] stimulus-evoked neuronal activity can be modified by an attentional induced additive bias or by a true gain modulation (present model parameters $v_b(t)$ and $g(t)$). Increase of gain $g(t)$ is correlated with increased blood flow through the respective cortical areas. Consequently in the present model the feedback gain serves as adaptive control parameter ($g \sim$ attention parameter G) which induces the rapid transitions between the alternative stationary perception states P1 and P2 through attention fatigue [6], like in [18]. An overdamped feedback system (time constant τ) is assumed leading to a first order dynamical equation. Formally this is achieved analogous to multistable optical systems [30]. The resulting phase oscillator equation (1) is similar to the phase attractive circle map of Kelso et.al. [23]. The phase variable may be compared with the phase shift between the coupled self-oscillating neuronal columns of the mean field theory [22]. Here phase locking between different groups of neurons is described by means of the circle (sin-) map. The complete dynamics of the present model is described by three coupled equations for the perception state order parameter (phase difference $v(t)$ between two superimposed neural mean fields), for the attention control parameter $G(t)$ (\sim feedback gain $g(t)$) modulating $v(t)$ after delay T), and for the attention bias or preference $v_b(t)$ representing a memory function via coupling to the low pass filtered perception state $v(t)$. The perception-attention-memory (PAM) equations are given by

$$\tau \dot{v}_{t+T} + v_{t+T} = G \left[1 + \mu \cos(\pi(v_t + v_B)) \right]. \quad (1)$$

$$\dot{G}_t = (v_b - v_t)/\gamma + (G_{\text{off}} - G_t)/\tau_G + L_t. \quad (2)$$

$$\dot{v}_{b_t} = (v_{b_e} - v_{b_t})M/\tau_L + (\bar{v}_t - v_{b_t})/\tau_M. \quad (3)$$

An ambiguous stimulus with strength I and difference of meaning μ (interference contrast $0 \leq \mu \leq 1$) of the two possible percepts P1, P2 excites two corresponding hypothetical mean fields $a(\Phi_1)$, $b(\Phi_2)$ with phase difference $\Delta\Phi = \pi v_t$ and $\mu = 2a_0b_0/(a_0^2+b_0^2)$ with amplitudes a_0 , b_0 . The nonlinear rhs. of equ. (1) describes the conventional interference between two superimposed coherent fields $J = |a+b|^2$ [30]. A recurrent process is established by feedback of the output after amplification (feedback gain g) with delay T into $\Delta\Phi$ via a hypothetical phase modulation mechanism $\Delta\Phi = K J$. As a quantitative estimate for T the stimulus–visual cortex response delay (≈ 40 ms) was suggested [3][27] which also represents typical recurrent delay times

within the association cortex [20]. In what follows I assume the phase bias $v_B = 0 \text{ mod } 2$. In agreement with Itti & Koch [25] the attention parameter $G(t) \sim \kappa I g(t)$ is the product of feedback gain $g(t)$ and input (stimulus) strength $I (=1 \text{ in what follows})$. The attention dynamics is determined by the attention bias v_b (determining the relative preference of P1 and P2), satiation speed $1/\gamma$, recovery time constant τ_G , and $G_{\text{off}} = \text{attention (gain) parameter for stimulus off}$, defined by $\mu = \mu_{\text{off}} < 0.18$ (see below). Following [18], the random noise due to physically required dissipative processes is added to the attention equation $G(t)$ as a stochastic Langevin force $L(t)$ with band limited white noise power J_ω . The attention bias or preference dynamics dv_b/dt is modelled as the sum of a learning function $M(v_t, v_b, v_{be})(v_{be} - v_b)/\tau_L$, and of a memory component $(\langle v_t \rangle - v_b)/\tau_M$ which couples v_b to the perception state ($\langle \rangle = \text{low pass filter}$). Learning is active only if one of the two percepts dominates whereas the other is initially weakly associated (initial preference $v_{b0} \neq v_{be}$ and $|\langle v_t \rangle - v_{be}| > |\langle v_t \rangle - v_{bt}|$), and a fluctuation induced jump into the weak perception state occurs, switching M from 0 to 1 for the duration of the weak state. The diagram of the model in Fig. 2 represents the highest level of an implementation with the dynamical systems tool Matlab-Simulink.

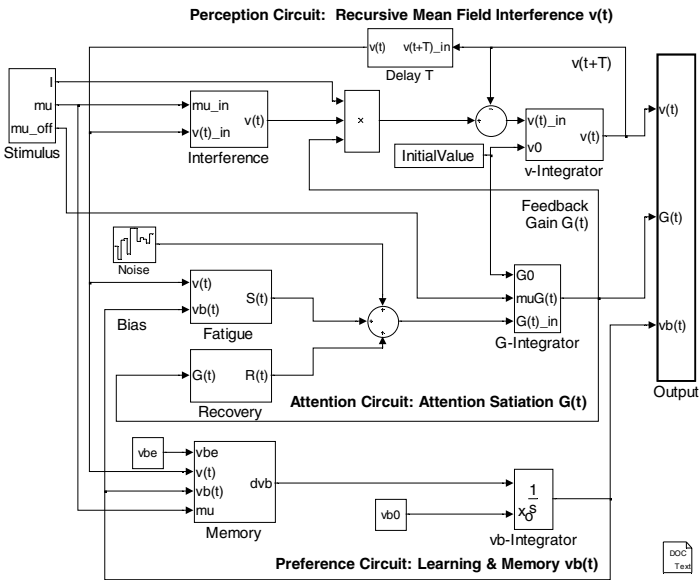


Fig. 2. Simulink implementation of phase oscillator equations (1) – (3). Subroutines (blocks) of the reentrant loops (from top to bottom): *perception circuit* $v(t)$, *attention circuit* $G(t)$ with *Fatigue* and *Recovery* component, and *Preference circuit* $vb(t)$ with two separate components for *Learning* and *Memory*. *Stimulus* of strengt I and *difference of meaning* μ ($=\mu$) are fed as control parameters into perception circuit with nonlinear (cosinoidal) *interference term* and *integrator loop* with time constant τ . *Attention circuit* $G(t)$ with *satiation (fatigue)* $(v_b - v(t))/\gamma$ and *recovery term* $(G_{\text{off}} - G(t))/\tau_G$ controls as gain factor the perception dynamics. The *Preference (Memory) circuit* is coupled to the *perception state* and modulates *Attention* as a dynamic bias $v_b(t)$.

2.2 Stationary Solutions

Quasiperiodic switching between two attractor states $v^*_1(P1)$ and $v^*_2(P2)$ emerges after a node bifurcation of the stationary $v^*(G, \mu)$ graph at $\mu_n \approx 0.18$. It evolves from a monotoneous function for $\mu < \mu_n$ into a hysteresis (S-) shaped ambiguous one with increasing μ [3][27][30]. Figure 3 depicts the first order stationary solutions ($dv/dt = 0, v_{t+T} = v_t = v^*$).

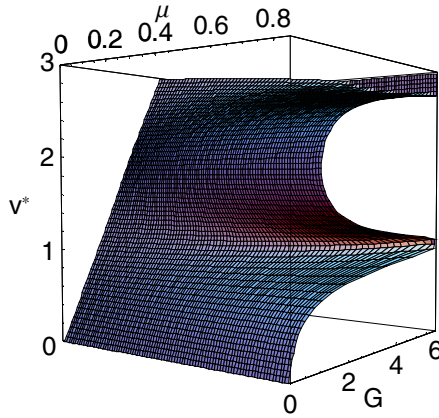


Fig. 3. Stationary solution v^* of perception state equation (1) as dependent on attention control parameter (feedback gain) G and difference of meaning (interference contrast) μ of percepts P1, P2. Node bifurcation at $\mu \approx 0.18$ defines transition into ambiguous solution ($\mu < \mu_n$, stimulus off $\rightarrow \mu > \mu_n$, stimulus on) with positive slope regions $v^*(G)$ representing stationary perception states P1(lower level $v^* \approx 1$), P2 (higher level $v^* \approx 2.5 \dots 3$)

At the critical value, $\mu_n \approx 0.18$, the slope of the stationary system state dv^*/dG becomes infinite with $(G_n, v_n) \approx (1.5, 1.5)$. For $\mu < \mu_n$ both percepts are fused into a single meaning corresponding to an unambiguous stimulus. For $\mu > \mu_n$ the stationary solution $v^*(G)$ becomes multivalued, corresponding to switching on of the ambiguous stimulus. For maximum contrast $\mu = 1$ the horizontal slope $(dG / dv)^{-1} = 0$ yields $v_i^\infty = 2i - 1, i = 1,2,3,\dots$ as stationary perception levels in the limit $G \rightarrow \infty$. The node bifurcation and hysteresis agrees with the qualitative deterministic catastrophe theoretical model of cognitive bistability as proposed by Poston & Stewart [17].

Higher order stationary solutions ($v(t+2^i T) = v(t) = v^*, i=0, 1,2,3,\dots$) yield period doubling pitchfork bifurcations on both positive slope regions of the hysteresis curve, with the G -values of the bifurcation points converging at the chaotic boundary according to the Feigenbaum constant $\delta_\infty = 4.6692 = \lim_{n \rightarrow \infty} (G_n - G_{n-1}) / (G_{n+1} - G_n)$ [3][27]. This proves that within certain parameter ranges (μ, τ) the P1-, P2-limit cycle oscillations include chaotic contributions which was confirmed by evaluation of the Lyapunov coefficient [27]. The linear stability analysis of (1) yields Eigenfrequencies $\beta = 2\pi f$ via $\beta\tau = -\tan(\beta T)$ which may be approximated by $f \approx f_0 i / (1 - \tau/T), i = 1, 2, \dots$

for $\tau \ll T$, with $f_0 = 1 / 2T = 12.5$ Hz for $T = 40$ ms [33]. For $i < 10$ (sufficient damping!) this spectrum lies well within the range of typical EEG frequency bands.

3 Computer Experiments

3.1 Simulated Perception–Attention Dynamics

In this section I present numerical evaluations of the coupled PAM differential–delay equations (1)(2)(3) as obtained with the Matlab–Simulink code of Fig. 2 using the Runge-Kutta solver "ode23tb" for stiff problems, i.e. fast changing dynamics. Figure 4 shows time series $v(t)$ and $G(t)$ (time units = simulation interval T_S) and phase space trajectories $v(G)$ for $T = 2T_S = 40$ ms, $\mu = 0.6$, $\tau = 0.5$, $\gamma = 60$, $\tau_G = 500$, constant attention bias $v_{bc} = 1.5$, noise power $J_\omega = 0.001$ with sample time $t_c = 0.1$, $\tau_M = 1000$, with stimulus–off interval ($\mu_{off} = 0.1$, $G_{off} = 1.5$) at the beginning of the time series.

The $v(t)$ dynamics in Fig.4a) exhibits the spontaneous transitions between stationary perception states P1 (near $v^* \approx 1$) and P2 (near $v^* \approx 2.5$) with the expected superimposed fast limit cycle and chaotic oscillations ($f_i = i \cdot 9.4$ Hz, $i = 1, 2, 3, \dots$) due to the delay $T = 2 T_S$. The transition time between P1 and P2 is of the order of $8 - 10 T_S \approx 150 - 200$ ms (see Fig. 4d)), in reasonable agreement with the time interval between stimulus onset and conscious perception [20].

The attention parameter $G(t)$ (adaptive feedback gain) in Fig.4b) exhibits the slow fatigue dynamics, leading to the quasiperiodic P1→P2 transitions at the G -extrema. The relative P1/P2-duration is modulated by the adaptive attention bias $v_b(t)$ (thick line, starting at $v_{b0} = 1.5$), inducing a memory and learning effect due to coupling to the averaged perception state $\langle v(t) \rangle$.

The phase space plot v vs. G in Fig. 4c) exhibits separate regions of the stimulus–off ($\mu = 0.1$) and stimulus–on ($\mu = 0.6$) states with the latter trajectory class exhibiting the jumps between the P1/P2 levels of v^* near the turning points of the stationary hysteresis curve $v^*(G)$. Limit cycle oscillations and deterministic chaos within P1, P2 originate from the finite delay T with the amplitudes corresponding to the pitchfork bifurcation pattern [3][27] and depending on damping time constant τ . The reversal time period is determined by the slow $G(t)$ dynamics, with fatigue and recovery time constants γ , τ_G . The adjusted absolute value of the attention parameter $|G_t - \langle G \rangle|$ in Figure 4d) exhibits qualitative agreement with the dynamics of the eye blink rate as reported by Ito et.al. [4]. It slows down under concentration on a task, i.e. attention. $\langle G \rangle = 0.5 (3 - \mu)/(1 - \mu^2) = 1.875$ for $\mu = 0.6$ is the center between the turning points of the S-shaped stationary hysteresis curve [3][27]. In agreement with the linear stability analysis [33] a 10 Hz limit cycle oscillation is observed on the P1-state whereas after the jump to the P2 state chaotic oscillations are excited as predicted in [3][27]. The prediction of chaotic oscillations of the perception state variable is consistent with recent research in the analysis and modelling of EEG time series (e.g. [34][35][36]) which emphasizes the relevance of the chaotic and fractal character of brain dynamics.

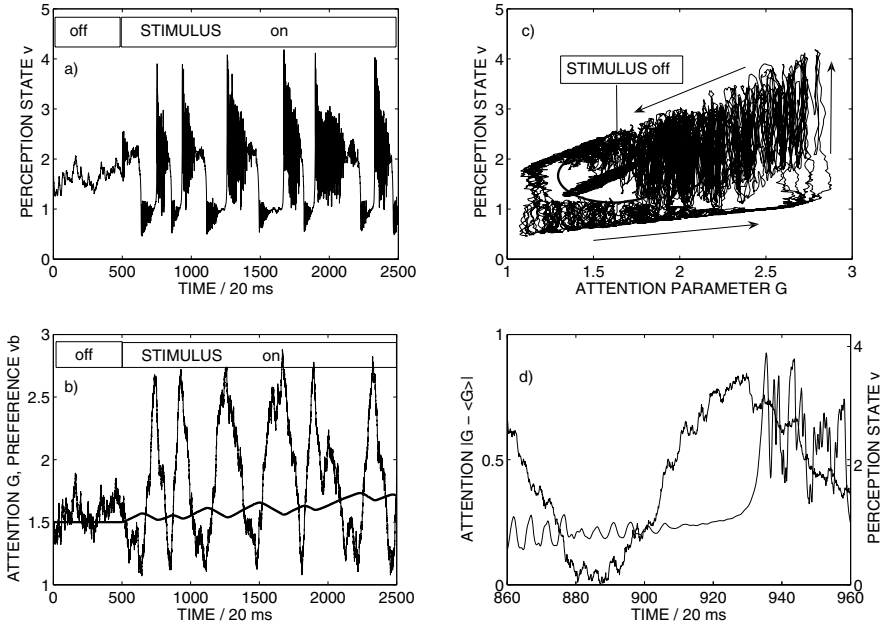


Fig. 4. Numerical evaluation of PAM equations (1)(2)(3). See text for simulation parameters. a) perception state $v(t)$; Stimulus off ($\mu = 0.1$) during $t = 0 \dots 500 T_S$ b) attention parameter $G(t)$ and dynamic attention bias or preference state v_b (thick line). c) Phase space trajectories v vs. G exhibiting separate region for $\mu = 0.1$ (= stimulus off, marked by ellipse) and $\mu = 0.6$. d) 2 s time window with single switch $v(P1 \rightarrow P2)$ (right ordinate) and superimposed adjusted absolute attention parameter $|G - \langle G \rangle|$ (left ordinate).

3.2 Reversal Time Statistics

The numerical simulations produced time series of perceptual switching events which are analyzed with respect to the relative frequency of perceptual duration times $\Delta(P1)$, $\Delta(P2)$ and to long range correlations. The duration time statistics has been shown in numerous experimental investigations (e.g. [7][19][29][31]) and different theoretical modelling approaches ([3][18][23]) to correspond to a Γ -distribution with shape parameter α and scaling parameter λ as a reasonable approximation. Mean and variance are given by $\Delta_m = \alpha/\lambda$ and $\sigma^2 = \alpha / \lambda^2$ respectively. As an example the left graph in Fig. 5 depicts the relative frequencies of the perceptual duration times of 100 simulation runs with $N = 5000$ time steps T_S each (parameters as in Fig. 4, however with larger noise: $J_0 = 0.004$ and $\tau_M = 3000$).

The 100 time series differ by noise generator (random number) seed value and perception state initial value $v_0 = v(t=0)$, $G(t=0) = G(v_0^*)$. Plotted is the distribution of the (typically 2000 – 3000) perceptual durations $\Delta(P2)$ of percept 2. In the above example mean and standard deviation are respectively $\Delta_m(P1) = 246T_S = 4.9$ s, $\sigma = 177T_S = 3.5$ s and $\Delta_m(P2) = 121T_S = 2.4$ s, $\sigma = 44T_S = 1.3$ s (left graph of Fig.5). The

ratios $\sigma/\Delta_m = 1/\sqrt{\alpha} \approx 0.71$ and 0.53 are within the range of most experimental findings reported in the literature (e.g. [7][19][29]).

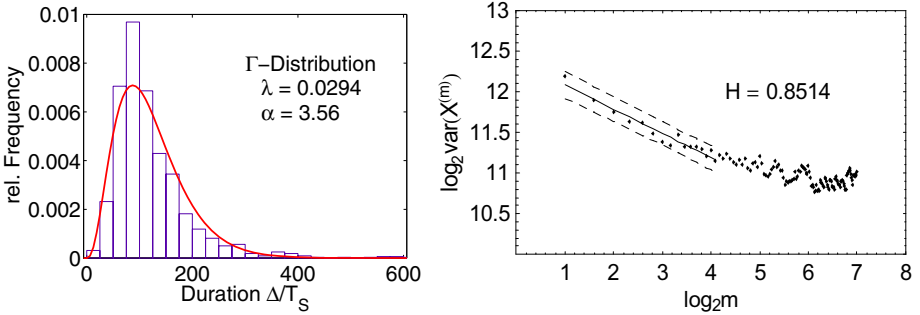


Fig. 5. Left: Relative frequencies of perceptual duration time $\Delta(P2)$, in units $T_S = T/2$, with 100 runs of $5000 T_S$ each. Simulation parameters $\mu = 0.6$, $v_{be} = 1.5$, $T = 2T_S$, $\tau = 0.5$, $\gamma = 60$, $\tau_G = 500$, $J_o = 0.004$ with sample time $t_c = 0.1$, attention bias (preference) time constants $\tau_M = 3000$, $\tau_L = 100000$. Maximum Likelihood fit with Γ -distribution (solid line). Right: plot of $\log(\text{variance})$ vs. $\log(\text{no. of samples } m)$ of the same simulation runs. Linear fit (95% conf. intervals) for estimating H via the slope.

The coupling of the preference parameter (attention bias) v_b to the perception state leads to long term correlations via memory effects which are quantified by the Hurst parameter H . The right graph of Fig. 5 depicts the variance–time plot ($\log(\text{variance}(\Delta(m)))$ vs. $\log(\text{sample size } m)$) with $\text{var}(\Delta(m)) = s^2 m^{2(H-1)}$ as used by Gao et.al. [1]. The H parameter is determined from the slope by fitting a straight regression line to the sample (m) range $m < 20$ (representing the data with sufficient statistics). The fit includes 95% confidence intervals of parameter estimates.

A significant long range correlation is observed due to the (short term) memory effect with time constant $\tau_M = 3000 T_S = 60$ s. As an alternative for estimating H the method of Kettani & Gubner [14] is employed which applies to 2nd order self similar or fractional ARIMA Gaussian processes. It estimates H via the lag-1 autocorrelation-function $\rho_n(1)$ and also provides an upper boundary of the 95% confidence interval:

$$\hat{H}_n \pm \delta \hat{H}_n = 0.5[1 + \log_2(\hat{\rho}_n(1))] \pm 5/\sqrt{n} \quad (4)$$

Table 1 lists results of the Hurst parameter evaluation of the $\Delta(P2)$ time series and Γ -distribution fits to the relative frequencies of $\Delta(P2)$ for memory and learning time constants τ_M, τ_L between $1000 T_S$ and $100000 T_S$ (computer simulations with 100 succeeding runs of $5000 T_S$ simulation length each). For $\tau_{M,L} > 50000$ the shape parameters α , mean percept durations Δ_m , and standard deviations σ converge to those of the previously reported simulations without memory and learning [3][27], i.e. with constant v_b . Large $\tau_{L,M}$ reduces the memory and learning effects and represents quasi static preference ratio. Correspondingly the last row of the table exhibits vanishing long range correlations ($H = 0.5$).

Table 1. Parameters for P2 of simulated perceptual duration time series as obtained with $v_{b0}=v_{be}$ and different memory / learning time constants τ_M, τ_L for $\mu=0.6, \gamma=60, \tau_G=500, T=2, \tau=0.5, J_\omega=0.004$: Hurst parameter, shape parameter, P2-duration mean Δ_m (in seconds), and relative standard deviation of Γ -distribution fit to relative frequencies. $\delta\alpha, \delta H$, are 95% confidence intervals

| τ_M | τ_L | $H_{KG} (\pm \delta H)$ | $H_{var} (\pm \delta H)$ | $\alpha(\pm\delta\alpha)$ | Δ_m / s | σ / Δ_m |
|----------|----------|-------------------------|--------------------------|---------------------------|----------------|---------------------|
| 1000 | 2000 | 0.61(0.06) | 0.72(0.01) | 3.5(0.2) | 2.7 | 0.53 |
| 1000 | 100000 | 0.75(0.08) | 0.86(0.01) | 2.2(0.2) | 2.7 | 0.67 |
| 2000 | 100000 | 0.71(0.07) | | 3.1(0.2) | 2.5 | 0.57 |
| 3000 | 100000 | 0.68(0.07) | 0.85(0.01) | 3.6(0.3) | 2.4 | 0.53 |
| 10000 | 100000 | 0.58(0.07) | | 4.3(0.3) | 2.4 | 0.48 |
| 50000 | 100000 | 0.51(0.06) | | 4.4(0.3) | 2.6 | 0.48 |

Increasing deviations from the Γ -distribution are observed with decreasing time constants τ_M, τ_L . This corresponds to increasing long range correlations due to growing influence of deterministic as compared to stochastic (attention noise) contributions. Significant long range correlations ($H > 0.5$) are observed due to the attention bias dynamics (memory and learning effect) if the time constant of the attention bias $\tau_M < 50000 T_s = 100$ s. In addition the learning component of dv_b/dt influences the dynamics in the initial phase if $|v_{be} - v_b(t=0)| > 0$ (M switches from 0 to 1), but only if $\tau_L < 2000$. If $v_{be}=v_{b0}$ the bias v_b corresponds to its preset equilibrium value v_{be} and learning stops whereas the dynamic memory variation continues.

The self similarity (Hurst) parameter H_{KG} (0.6 - 0.75 for $\tau_M \leq 3000$) as obtained with the Kettani-Gubner method [14] is systematically smaller as compared to H_{var} (0.7 - 0.85) obtained with the variance - time method used by Gao et.al. [1] for the evaluation of their bistability experiments. This may be due to the fact that equation (4) for H_{KG} was derived for exactly second order self similar processes which may not be true for our case. Nevertheless, because an evaluation of autocorrelation functions of simulated perceptual duration time series revealed no significant time dependence, stationarity appears to be fulfilled and the general agreement of the simulations with the experimental results in [1] ($H_{var} = 0.6 - 0.84$, for different subjects) supports the fractal character of the simulated reversal time series. This finding again fits into the proposed picture of underlying nonlinear brain dynamics as derived from analysis and theoretical modeling of EEG time series (e.g. [34][35][36]).

An interesting aspect of the simulation results in Table 1 concerns the comparison with experimentally observed differences between binocular rivalry and ambiguous figure reversal [37][41]. Whereas binocular rivalry appears to involve a more automatic, stimulus driven form of competition [41] and exhibits no chaotic contribution in the reversal time statistics [37][43], alternation rates of ambiguous figure reversal on the other hand show strong response to selective attention, i.e. can be voluntarily controlled by observers [41]. Moreover, when determining the correlation dimension D_2 from experimental reversal time series a significant deterministic contribution was observed which was not the case for rivalry [37]. In terms of the present model binocular rivalry corresponds to large attention bias time constants τ_M, τ_L (last row of Table 1) with vanishing long range correlations ($H \approx 0.5$). The influence of memory and learning vanishes under these conditions and the perceptual switching between the different stimuli is reduced to the fatigue and noise induced alternation.

4 Conclusion and Outlook

The reversal time statistics of alternating perception states is derived by computer simulations using a recursive nonlinear dynamics model with a minimum architecture based on perception-attention-memory (PAM) coupling. The model is mapped to a simplified reentrant circuit within the association cortex including attentional feedback modulation of the ventral stream [25]. The dynamics of a phase oscillator perception circuit is modulated by reentrant adaptive gain with delay $T \approx 40$ ms for modelling attention fatigue, and includes additive attention noise. The attention (= feedback gain) in turn is biased by an adaptive preference parameter coupled to the (filtered) perception state for simulating memory effects. Perceptual reversal time statistics are fitted by Γ -distributions with mean and variance obtained in the experimentally observed range of some seconds [6][7][29][31]. Reversal time series exhibit long range correlations characterized by a Hurst (2^{nd} order self similarity) parameter $H > 0.5$ in agreement with experimental results of Gao et.al. [1]. Within the present model these long range correlations are explicitly related to the dynamics of memory and learning. The experimentally observed differences between binocular rivalry and ambiguous figure reversal [37][41] can be attributed to different attention bias (memory) time constants. The simulated attention-perception dynamics based on the ventral loop of Fig.1 agrees with the experimental results of Nakatani & van Leeuwen [4] and supports the assumption that attentional effort which is expressed by eye blinking and saccade frequencies controls switching rates [29]. The present model furthermore supports the early proposal of Poston & Stewart [17] and of Wilson [39] of a deterministic catastrophe topology as the formal basis of the perception reversal dynamics. Further numerical simulations will consider continuously changing stimulus parameters (stimulus strength I , difference of meaning μ) for comparison with corresponding experimental conditions. An extension of the present model aims at multistable perceptual switching between three and more stationary states. This can be achieved by a vector field approach, describing each percept by a separate set of PAM-equations.

Acknowledgement

I am indebted to Monika Mittendorf for help with the computer experiments, to H. Nakatani of Riken Brain Science Institute for information on recent experimental results, and to J.B. Gao and K.D. White of Univ. of Florida for providing a preprint of their work.

References

1. Gao, J.B., Merk, I., Tung, W.W., Billok, V., White, K.D., Harris, J.G., Roychowdhury, V.P.: Inertia and memory in visual perception. *Cogn. Process* 7, 105–112 (2006)
2. Mandelbrot, B.B.: *The fractal Geometry of Nature*. German translation: Birkhäuser, pp. 265–270 (1991)

3. Fürstenau, N.: Modelling and Simulation of spontaneous perception switching with ambiguous visual stimuli in augmented vision systems. In: André, E., Dybkjær, L., Minker, W., Neumann, H., Weber, M. (eds.) PIT 2006. LNCS (LNAI), vol. 4021, pp. 20–31. Springer, Heidelberg (2006)
4. Ito, J., Nikolaev, A.R., Luman, M., Aukes, M.F., Nakatani, C., van Leeuwen, C.: Perceptual switching, eye movements, and the bus paradox. *Perception* 32, 681–698 (2003)
5. Blake, R., Logothetis, N.K.: Visual competition. *Nature Reviews / Neuroscience* 3, 1–11 (2002)
6. Orbach, J., Ehrlich, D., Heath, H.A.: Reversibility of the Necker Cube: An examination of the concept of satiation of orientation. *Perceptual and Motor Skills* 17, 439–458 (1963)
7. Borsellino, A., de Marco, A., Allazetta, A., Rinesi, S., Bartolini, B.: Reversal time distribution in the perception of visual ambiguous stimuli. *Kybernetik* 10, 139–144 (1972)
8. Hupe, J.-M., Rubin, N.: The dynamics of bistable alternation in ambiguous motion displays: a fresh look at plaids. *Vision Research* 43, 531–548 (2003)
9. Koch, C.: *The Quest for Consciousness – A Neurobiological Approach*, German Translation, Elsevier, München (2004)
10. Engel, A.K., Fries, P., Singer, W.: Dynamic Predictions: Oscillations and Synchrony in Top-Down Processing. *Nature Reviews Neuroscience* 2, 704–718 (2001)
11. Engel, A.K., Fries, P., König, P., Brecht, M., Singer, W.: Temporal binding, binocular rivalry, and consciousness. *Consciousness and Cognition* 8, 128–151 (1999)
12. Srinivasan, R., Russel, D.S., Edelman, G.M., Tononi, G.: Increased synchronization of magnetic responses during conscious perception. *J. Neuroscience* 19, 5435–5448 (1999)
13. Edelman, G.: *Wider than the Sky*. Penguin Books, pp. 87–96 (2004)
14. Kettani, H., Gubner, J.A.: A Novel Approach to the Estimation of the Long-Range Dependence Parameter. *IEEE Trans. Circuits and Systems-II: Express Briefs* 53, 463–467 (2006)
15. De Marco, A., Penengo, P., Trabucco, A., Borsellino, A., Carlini, F., Riani, M., Tuccio, M.T.: Stochastic Models and Fluctuations in Reversal Time of Ambiguous Figures. *Perception* 6, 645–656 (1977)
16. Merk, I.L.K., Schnakenberg, J.: A stochastic model of multistable perception. *Biol. Cybern.* 86, 111–116 (2002)
17. Poston, T., Stewart, I.: *Nonlinear Modeling of Multistable Perception*. *Behavioral Science* 23, 318–334 (1978)
18. Ditzinger, T., Haken, H.: A Synergetic Model of Multistability in Perception. In: Kruse, P., Stadler, M. (eds.) *Ambiguity in Mind and Nature*, pp. 255–273. Springer, Berlin (1995)
19. Levelt, W.J.M.: Note on the distribution of dominance times in binocular rivalry. *Br. J. Psychol.* 58, 143–145 (1967)
20. Lamme, V.A.F.: Why visual attention and awareness are different. *Trends in Cognitive Sciences* 7, 12–18 (2003)
21. Tononi, G., Edelman, G.M.: Consciousness and Complexity. *Science* 282, 1846–1851 (1998)
22. Schuster, H.G., Wagner, P.A.: A Model for Neural Oscillations in the Visual Cortex: 1. Mean field theory and the derivation of the phase equations. *Biol. Cybern.* 64, 77–82 (1990)
23. Kelso, J.A.S., Case, P., Holroyd, T., Horvath, E., Racaszek, J., Tuller, B., Ding, M.: Multistability and metastability in perceptual and brain dynamics. In: Kruse, P., Stadler, M. (eds.) *Ambiguity in Mind and Nature*, pp. 255–273. Springer, Berlin (1995)

24. Nakatani, H., van Leeuwen, C.: Transient synchrony of distant brain areas and perceptual switching in ambiguous figures. *Biol. Cybern.* 94, 445–457 (2006)
25. Itti, L., Koch, C.: Computational Modelling of Visual Attention. *Nature Reviews Neuroscience* 2, 194–203 (2001)
26. Robinson, D. (ed.): *Neurobiology*. Springer, Berlin (1998)
27. Fürstenau, N.: A chaotic attractor model of cognitive multistability. In: *Proceedings IEEE 2004 Int. Conf. on Systems, Man and Cybernetics*, IEEE cat. no. 04CH37583C, pp. 853–859 (2004)
28. Hillyard, S.A., Vogel, E.K., Luck, S.J.: Sensory gain control (amplification) as a mechanism of selective attention: electrophysiological and neuroimaging evidence. In: Humphreys, G.W., Duncan, J., Treisman, A. (eds.) *Attention, Space, and Action*, pp. 31–53. Oxford University Press (1999)
29. Nakatani, H., van Leeuwen, C.: Individual Differences in Perceptual Switching rates: the role of occipital alpha and frontal theta band activity. *Biol. Cybern.* 93, 343–354 (2005)
30. Watts, C., Fürstenau, N.: Multistable fiber-optic Michelson Interferometer exhibiting 95 stable states. *IEEE J. Quantum Electron* 25, 1–5 (1989)
31. Zhou, Y.H., Gao, J.B., White, K.D., Merk, I., Yao, K.: Perceptual dominance time distributions in multistable visual perception. *Biol. Cybern.* 90, 256–263 (2004)
32. Pitts, M.A., Nerger, J.L., Davis, T.J.R.: Electrophysiological correlates of perceptual reversals for three different types of multistable images. *J. of Vision* 7, 1–14 (2007)
33. Fürstenau, N.: Nonlinear dynamics model of cognitive multistability and binocular rivalry. In: *Proceedings IEEE 2003 Int. Conf. on Systems, Man and Cybernetics*, IEEE cat. no. 03CH37483C, pp. 1081–1088 (2003)
34. Lutzenberger, W., Preissl, H., Pulvermüller, F.: Fractal dimension of electroencephalographic time series and underlying brain processes. *Biol. Cybern.* 73, 477–482 (1995)
35. Dafilis, M.P., Liley, D.T.J., Cadusch, P.J.: Robust chaos in a model of the electroencephalogram: Implications for brain dynamics. *Chaos* 11, 474–478 (2001)
36. Burke, D.P., de Paor, A.M.: A stochastic limit cycle oscillator model of the EEG. *Biol. Cybern.* 91, 221–230 (2004)
37. Richards, W., Wilson, H.R., Sommer, M.A.: Chaos in percepts. *Biol. Cybern.* 70, 345–349 (1994)
38. Deco, G., Marti, D.: Deterministic Analysis of Stochastic Bifurcations in Multi-Stable Neurodynamical Systems. *Biol. Cybern.* 96, 487–496 (2007)
39. Wilson, H.R.: *Spikes, Decisions, and Actions*. Oxford University Press (1999)
40. von der Malsburg, C.: The Coherence Definition of Consciousness. In: Ho, M., Miyashita, Y., Rolls, E.T. (eds.) *Cognition, Computation, and Consciousness*, pp. 193–204. Oxford University Press (1997)
41. Meng, M., Tong, F.: Can attention selectively bias bistable perception? Differences between binocular rivalry and ambiguous figures. *J. of Vision* 4, 539–551 (2004)
42. Hamker, F.H.: A dynamic model of how feature cues guide spatial attention. *Vision research* 44, 501–521 (2004)
43. Lehky, S.R.: Binocular rivalry is not chaotic. *Proc. R. Soc. Lond. B* 259, 71–76 (1995)

On Constructing a Communicative Space in HRI

Claudia Muhl, Yukie Nagai, and Gerhard Sagerer

Applied Computer Science, Faculty of Technology, Bielefeld University,
33594 Bielefeld, Germany
{cmuhl,yukie,sagerer}@techfak.uni-bielefeld.de

Abstract. Interaction means to share a communicative space with others. Social interactions are reciprocally-oriented activities among currently present partners. An artificial system can be such a partner for humans. In this study, we investigate the effect of disturbance in human-robot interaction. Disturbance in communication is an attention shift of a partner caused by an external factor. In human-human interaction, people would cope with the problem to continue to communicate because they presuppose that the partner might get irritated and thereby shift his/her interactive orientation. Our hypothesis is that people reproduce a social attitude of reattracting the partner's attention by varying their communication channels even toward a robot. We conducted an experiment of hybrid interaction between a human and a robot simulation and analyzed it from a sociological and an engineering perspective. Our qualitative analysis revealed that people established a communicative space with our robot and accepted it as a proactive agent.

Keywords: Human-Machine Interaction, Social Robotics, Disturbance in Communication.

1 Introduction

Humans are social beings. The interaction partners use alternating demonstrations and utterances, which corporately construct a communicative space. Such immaterial space is *social*. People together establish a binary structured situation which is constructed through their communicative activities in each human-human interaction (HHI) [1]. However we can wonder whether the interaction with artificial partners is also social. Can double sided activity be performed? What can help to clarify human-robot interaction (HRI) and make it easier?

Social activities can be expanded to inanimate and be build up among humans and artifacts. We can evaluate proactive social activities of a robot system by means of its social embeddedness. Dautenhahn et al. [2] have been measuring the degrees of embodiment, situatedness, and social embeddedness in different biological and artificial systems. They give requirements for the design of interaction-aware machines, which have a high social impact.

Many researchers have been attempting to find factors which are influencing the human impression of robots [3]. For example, Goetz et al. [4] suggested

that matching a task and the robot's behavior to it would improve human-robot cooperation. They designed two types of robots, a playful and a serious one, and compared the people's acceptance of the robots working either on an entertaining or a serious task. People accepted better and interacted longer with the robot which acted in an appropriate manner to the task. Minato et al. [5] defined the familiarity of a robot with respect to its appearance and behavior. They extended the uncanny valley proposed by Mori [6] and described how the above two factors synergistically affect the people's impression of a robot. Their experiments using their android, in which the human response of breaking eye contact was measured, showed that people dealt with the android as a human-like agent. These studies had a major impact on the design of communication robots, however, they focused only on the factors directly relevant to HRI.

Various activities in interactions and elements of communication have been studied. We here also explore factors which could encourage HRI. We propose to concentrate on a foremost counter-intuitive aspect: disturbance in an interaction. In HRI this phenomenon is usually disliked and is regarded as a problem. However the disturbance in interaction can be a positive factor. Focusing on HHI, we notice accompanying effects in disrupted communications. If two people are talking while a television or radio is turned on, different effects can be observed: People continue to communicate with the partner and even intensify their activities toward the other whose attention has been shifted to the additional visual and/or acoustic input. They can deal with the upcoming problems because they presuppose an irritation. People know the strategies to regain the interactive orientation.

We designed a robot simulation embedded with a mechanism of primal visual attention. Although the robot shows only primitive reactions, it can motivate the human partners to hold on the interaction. This enables us to explore the variety of human reactions and to point out the observed social activity, which is highly oriented toward the artificial partner. We suggest that people who are accepting the robot as a situated intentional partner might dislike a distracted robot's attention, try to regain it, and make it re-engage in the interaction actively.

Here we investigate from a sociological perspective how disturbance in HRI affects the behavior of human partners. We wish to add an analytical perspective from the field of sociology to the continuously emerging interdisciplinary discourse in the field of robotics (e.g., [7,8]). We believe that a qualitative approach helps to reveal effects induced by disturbance, to discover how people act and react toward a robot, and thereby to contribute to the development of social robots.

In Section 2, we depict sociological communication theory of interaction between humans and the phenomenon of disturbance in communication. Our robot simulation as a communication partner is explained in Section 3. In Sections 4 and 5, we introduce an experiment of HRI using the robot simulation and show the results of the qualitative analysis of people's reactions. The results are discussed from both a sociological and an engineering point of view in Section 6. Finally, the conclusion and further research issues are given in Section 7.

2 Sociological Aspects of Communication

2.1 Involvement in Social Interaction

Every contact which takes place between humans, who are addressing others, is social. Sociological systems theory denominates communications among partners in attendance as *interacting systems* [9]. One criteria for communication is the reciprocal percipience, whereas the presence or media mediated utterances of both partners is a prerequisite to be acquired [10]. The reciprocity of awareness of the coparticipants means that both are sharing contextual perceptions which enable them to construct a common sense and to build a situated common ground. These operations open an intersubjective space of social actions and expressions. Participating in an interaction system means to be engaged in the reciprocal course of action with interactive practices and to shape it with reciprocal addressed behavior [11].

Sequences of contributions of speech and actions like mimic, gesture, and body movements are aligned with the partner in each interaction [12,13]. All elements of the dialog are organized as reciprocal turns which are successively arranged in a turn-taking set. Each participant of an interaction is oriented toward the partner by considering his/her individual situated involvement. This phenomenon includes a sensitivity to the coparticipant and his/her situatedness. Sacks et al. [14] named the context-aware possibility of referencing to partners' actions and utterances in HHI *recipient design*. Garfinkel [15] takes constructivist arguments into account when he describes interactions as situated in a specific context, which is constructed by each interaction partner employing his/her own category systems, commonsense knowledge, and practical reasoning to the actual experience. Though the interaction partners achieve mutual understanding. As a consequence of this individual construction of the specific social situation, humans are able to act within their circumstances and to interpret others. However communications consist of mutual constructions of the situation. Von Glasersfeld [16,17] takes into account that our reality is built upon experiences and the utilization of feasible strategies. Every individual is constructing such a space within his/her mind by using the perceived world and relating it to former and actual experiences. This argument of constructivist theory and the findings about social interaction characteristics inspire the idea of a constructed communicative space.

2.2 Dealing with Disturbances

Communications are fragile and their alternating follow-up is often disrupted by surrounding factors. What happens in case of addressing someone who has lost concentration and is occupied with processing information derived from a third person's perspective? This shift of the attention will be recognized by the partner and cause some reaction. Spontaneously appearing reasons might effectuate severe irritations that can lead to discontinuity in the dialog processing and the interaction might be terminated. By lifelong practice humans learn to

deal with such disturbances. They can defy the problem, and thus an originally negative cause leads to positive effects.

In HHI the problem of focusing the attention in a communication has been investigated intensively. Social interactions can be studied in everyday life, and such analysis revealed that humans moreover often implement disturbances in communications themselves. Goodwin and colleagues [10, 12] analyzed multiple face-to-face interactions of humans in different contexts. They discovered that the ideal turn-taking in *talk-in-interactions* is often disturbed by the co-participants themselves [10]. Such strategic elements are used in order to evaluate whether the partner's co-orientation is still focused on the ongoing interaction [12].

Those techniques are applied to organize the exchange of speech and gaze, and, if inconvenience is discovered, *repairing mechanisms* are initiated [18]. For example, the speaker often stops him-/herself and restarts the sentence with identical words. Such explicitly evoked breaks in the verbal flow ensure about the interaction partner's concentration on the mutual topic. As a positive effect, those disturbances affirm the interaction and the dialog can be continued.

3 A Communication Robot with Primal Visual Attention

In order to study the effect of disturbance in HRI, we developed a robot simulation of which attention can be naturally distracted by a visual disturbance.

3.1 Robot Simulation

Fig. 1(c) shows a robot simulation used in our experiment, which was originally developed by Ogino et al. [19]. The robot has only the face with an infant-like appearance, thus being considered adequate to examine the nature of HRI.

In our experiment, the robot interacted with a human partner by changing its gaze direction as well as facial expression in response to visual input. The gaze direction was controlled so that communication partners could perceive that the robot was looking at a likely interesting location in the environment. The explanation of the attention mechanism is given in the next section. Facial expression was also used to facilitate the interaction. The robot, for example, showed pleased expression by rising the eyebrows and opening the mouth when it could stably look at a static target.

3.2 Mechanism of Primal Visual Attention

As the mechanism for the robot's vision, we adopted the model of saliency-based visual attention proposed by Itti et al. [20, 21]. The model based on the neuronal mechanism of primates enables the robot to imitate the primary attention of primates, who can rapidly detect and gaze at salient locations in their views. A salient location is here defined as a spot which locally stands out from the surroundings with respect to its primitive visual features: color, intensity, orientation, flicker, and motion [21]. For example, a human face can be detected

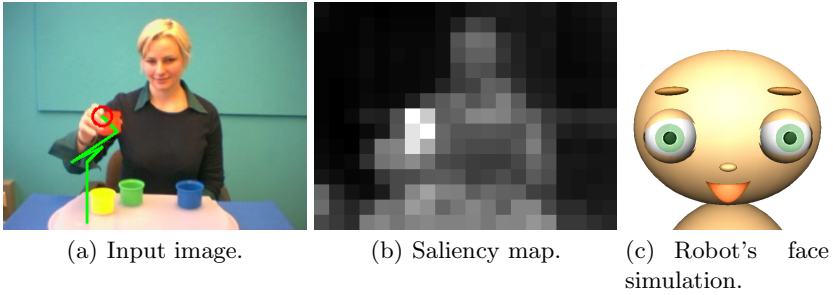


Fig. 1. A scene without disturbance. The robot is gazing at the red cup held by the human partner because of its outstanding color and motion.

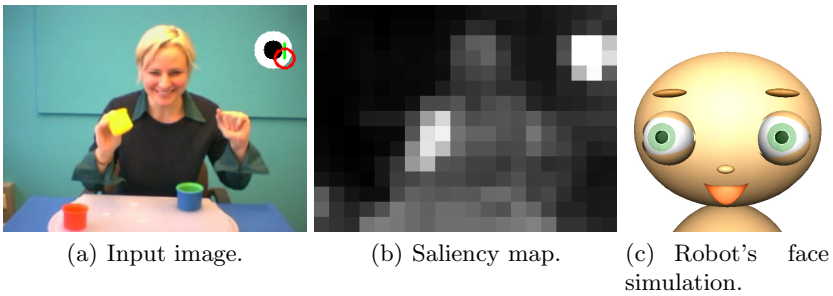


Fig. 2. A scene with disturbance, a black and white circle. The robot is looking at the disturbance because of its highly contrasted intensity and motion, although the human partner tries to attract its attention by showing the yellow cup and shaking her left hand.

as salient because of its intrinsic features, i.e., the skin color and the compound form, as well as of the motion even though no face model is applied to. The saliency model therefore enables the robot to detect likely important locations in the interaction without any top-down knowledge about the situation or the communication partners. The effectiveness of the model has been demonstrated in the studies of social robot learning and social robot interaction (e.g., [22,23]).

Fig. 1 (a) shows an example of the visual input captured in the experiment. The human partner was picking up and showing a red cup to the robot in a blue background. Fig. 1 (b) gives the corresponding saliency map, in which the degree of saliency is represented by the brightness of the pixels. The map was generated by calculating the difference between each pixel and the surrounding ones, which highlighted the prominent pixels in the image. Refer to [20,21] for more detailed mechanism. In our HRI experiment, the robot gazed at the most salient location in each image frame. In the scene shown in Fig. 1 (a), the red cup held in the right hand of the human partner had been attended to for a while because of its outstanding color and motion. The current position of the attended location and its trajectory are denoted by a red circle with green lines.

The robot shown in Fig. 1 (c) was captured when it was gazing at the red cup. The robot's eyes were controlled so that human partners could perceive that it was responding to their action and was looking at an interesting location for it. Note that, in our experiment, human partners could only see the simulation of the robot's face, but not the input image or the saliency map.

3.3 Disturbance in Robot's Vision

To distract the visual attention of the robot during the interaction, we created a salient object superimposed in the input image. Fig. 2 shows a scene captured while a disturbing object was put at the upper-right corner of the image. The object was designed as a white circle with a smaller black circle, which vibrated randomly. Because of the highly contrasted intensity and the motion, it attracted mostly, but not certainly, the robot's attention. In Fig. 2, the robot gazed at the disturbing object although the human partner tried to attract its attention by showing the yellow cup and shaking her left hand. Note that the disturbing object was presented only in the robot's vision, not in the real environment, and therefore human partners could not discover anything at the location where the robot was looking. The initial position of the disturbance had been fixed at the upper-left or upper-right corner of the image.

4 Method for HRI Experiment

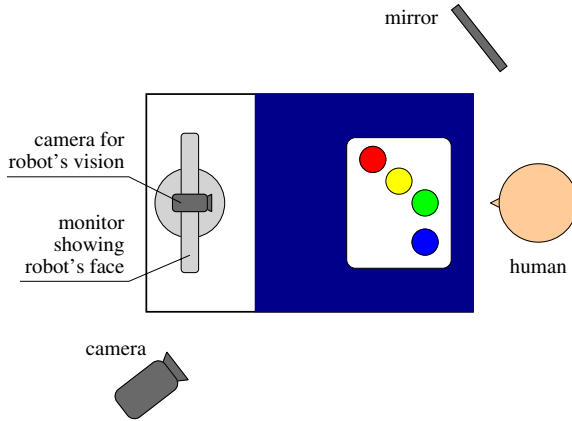
We conducted a HRI experiment using our robot simulation. By controlling the disturbance in the robot's vision, we investigated its effect on the following behavior of human partners.

4.1 Participants

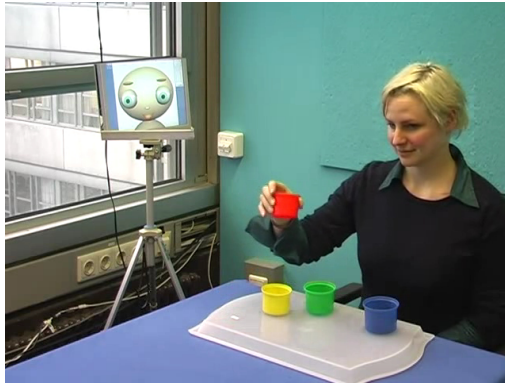
Twenty-two university students (sixteen males and six females) participated in the experiment as communication partners. Sixteen of them major in computer science and thus are familiar with robotic systems, while the others studying sociology or linguistics are not. All of them saw our robot for the first time in this experiment.

4.2 Setting

Fig. 3 (a) illustrates the experimental setup, and (b) shows a sample scene of the experiment. A human partner was seated at a table facing the robot simulation displayed on a computer monitor. The window for the simulation was enlarged to fill the screen so that the partners could get the impression of the embodied robot with a monitor head. A FireWire camera for the robot's vision was placed on the monitor. No other sensors or actuators, e.g., microphone or speaker, were used, meaning the robot could respond only visually but not by other modalities. Another camera beside the monitor videotaped the interaction between the robot reflected on a mirror and a partner, which was used for the later analysis.



(a) Top-view of the experimental environment.



(b) A scene of the videotaped HRI, in which the robot is reflected on the mirror.

Fig. 3. Experimental setup for HRI

4.3 Procedure

The human partners were asked to teach some tasks, e.g., stacking cups, serving tee, and sweeping on the table, to the robot by using prepared objects. They were allowed to choose the objects and to decide what to and how to teach with them. Nothing about the usage of their gesture or speech was instructed. That is, they could use all their communication channels if they wanted although they were told of the robot's capability, i.e., it could perceive and respond only visually, beforehand. The mechanism of the robot's visual attention was not explained to them.

The interaction with the robot went on for five to more than thirty minutes depending on the partner. Over the interaction, the disturbing object was presented in the robot's vision three to thirty times at a maximum. The timing to insert and to remove the disturbance was decided by an experimenter responding

to the partner's reaction. In other words, the partner's efforts to reattract the robot's attention did not directly but indirectly effected it although they could not realize that the experimenter was controlling the disturbance.

4.4 Sociological Analysis

Qualitative sociological methodology helps to identify concrete human behavior and social interaction in a contextual setting [24]. It seeks to describe the underlying social patterns which occur as concrete phenomena in the real world. The resource for this method is data taken from everyday phenomena like dialogs. Here communication patterns can be studied and framed. In this experiment we make use of ethnomethodological conversation analysis to investigate the video data of the HRI.

Conversation analysis is a qualitative method to evaluate the speech and action processes of individuals in a continuous interaction situation [10]. This close grained analytical technique starts with describing prominent elements from the empirical data. With the categorization of action patterns, the interaction structure can be revealed.

The goal of the sociological reasoning in our HRI experiment is to evaluate the interactive potential of irritation. The disturbance of the robot becomes part of the interaction system, meaning it causes irritation in the human partners that leads to a change in their behavior.

5 Results of People's Responses to Robot's Disturbance

The human partners were showing different strategies concerning eye-contact in the interaction. Some of them mostly concentrated on their own actions and thus inspected the robot's gaze immediately after having fulfilled a task. Others checked the robot's attention during the activity. When they recognized extraordinary changes in the robot's gaze behavior, all of them got irritated and swerved their task. By ascertaining a differentiated set of actions in case of disturbance, we searched for specific features in the human behavior. Here we focused on aspects that occurred during the interactions affected by the disturbance.

5.1 Categorization of People's Frequent Responses

Analyzing the individual performances, we found a set of main strategies over all human participants in the experiment. People were directed toward the robot and attended to evaluate the cause of its behavior. Summing these observations, we propose a map shown in Fig. 4, which is locating categories of the people's responses caused by the robot's disturbance. We have five categories scaled on two axes: the physical and psychological distance to the robot and the implied change in the subjects' activity, strong enough to recover the relationship in the ongoing HRI.

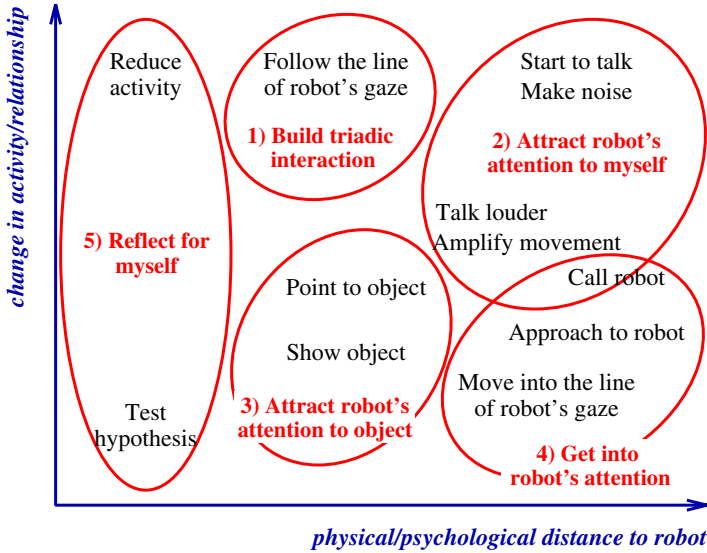


Fig. 4. People's responses to the robot's disturbance. Five categories are created on a two-dimensional map with reference to the physical/psychological distance to the robot and the change in the activity/relationship.

(1) Building triadic interaction: While interacting with the robot, some participants followed the line of the robot's gaze and tried to achieve joint attention when the robot had been disturbed (see Fig. 5 (a)). At the same time, they often commented verbally on the expected direction of the robot's gaze, although there would not be anything to discover. This reaction shows situated involvement. That is, a human partner follows the robot's action and attributes a participant's role to it. This phenomenon marks the evolvement of a triadic interaction, which includes the surrounding context.

(2) Attracting the robot's attention to oneself: The next category represents a huge variety in the reactive intensity. The human partners began exaggerating their already performed actions. They enlarged their gestures and movement (see Fig. 5 (b)). Others called the robot, just started to talk to the robot or made noise, even though they already had tested the robot would not react to acoustic signals (see Fig. 5 (c)). They seemed to try to attract the robot's attention to themselves.

(3) Attracting the robot's attention to an object: The third category assembles strategies that could possibly attire the robot's attention back to the object, i.e., getting closer to the robot while demonstrating the object (see Fig. 5 (d)). The object had been shaken or closely presented to the robot. Some people also pointed to the object to re-attract the robot's gaze.

(4) Getting into the robot's attention: Reaching closer to the robot builds the fourth category of action. Here we sum movements like a physical approach

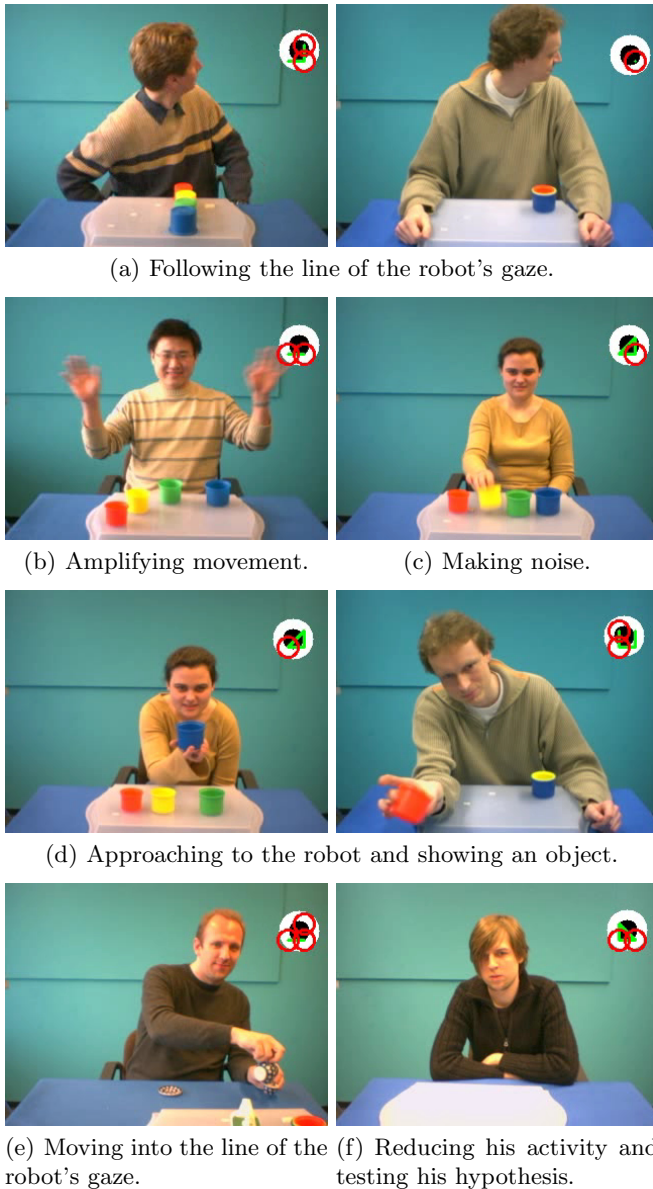


Fig. 5. Sample scenes of the people's responses caused by the disturbance

of the partners to the robot. Some of them even spatially moved into the line of the robot's gaze (see Fig. 5 (e)). As a consequence, they became more present to the robot and decreased the psychological distance to it.

(5) Reflecting to oneself: The fifth category assembles the biggest and smallest change in the human activity compared to their former way of action toward

the robot. Some of them tested their hypothesis on the robot's functions by increasing and others by reducing the intensity of their activity (see Fig. 5 (f)). This included the sequential variation of their former applied action patterns toward the robot.

In this experiment, humans performed social actions toward the robot. Depending on the complexity of the interaction and its direction, we consider three groups of observed behavior: a triadic interaction (category 1), a dyadic one (categories 2 to 4), and other (category 5), in which people did not direct interactive utterances but rather went along with inner reflections.

5.2 Diversity of People's Responses

As expected, people interacting with the robot, which was attracted by an emerging disturbance, showed an immediate change in their behavior when they realized the interaction had been affected. Some reactions like little smiles and very brief frowns came up slightly and were presumably unconscious. Therefore only the most common reactions of the human partners have been listed and classified.

We observed the tendency to repair the situated disorder. Our findings prove that human action is likely to be varied in case the expected results are not relieved. The concrete reactions demonstrate a renewed conceptualization of the situation and the modification of the human hypothesis on the robot's functions, which reminds of recipient design in HHI, which also allows to flexibly change the expectations. All of these strategies tend to refresh and repair the irritated flow of communication. After the appearance of a disturbance, the completion of the primordial task often has been abandoned. In these cases the communicative process was reestablished and the interaction was mostly even intensified.

6 Discussions

6.1 Constructing Communicative Spaces in Interactions

Each interaction in this HRI experiment opened a new communicative space. Both the robot and the human partner contributed to it. In order to evaluate the impact of situatedness and social dynamics, individual differences should be taken into account. The usage of action, speech, and interpretation as well as their relevance to the interactions became evident. The human partners used social repairing mechanisms known in sociological conversation analysis. One effect of the disturbance is the encouragement of gestures and utterances. The robot's distraction motivated the partners to reveal strategies to regain its attention and to recover a turn-taking process. This phenomenon can also be observed in participants in HHI. People use the turn-taking to exchange the information efficiently and also apply repair mechanisms to vanquish distractions.

We could also find differences in the proactive engagement of the humans. Their interaction strategies seem to vary corresponding to their familiarity with

robot systems. As people applied recipient design, they introduce background knowledge and projections derived from the expectations to the actual interaction. If the participants take their background knowledge into account, we need to find which specific knowledge they bring in. In our experiment, for example, people studying computer science showed a systematic behavior to test what could have caused the mistake in the interaction. Our evaluation demands further investigations concerning the variety of the interactive behavior influenced by the background knowledge of humans.

Although we did not develop an embodied robot, our robot simulation has been treated as an interaction partner, which is sharing the same spatial situation. This observation corresponds to the discussion given by Kidd and Breazeal [25]. They compared interactions with either a physically present robot or its presentation on a screen, and could not find a significant difference in the participants' responses. However they presented a real robot, not simulation, in both cases. These findings allow us to summarize that although embodiment is important, it is not the only factor which influences the success of an HRI.

6.2 Potential of Robot's Primal Visual Attention in HRI

In developing our communication robot, we introduced the primal attention mechanism based on saliency for the robot's vision. A more common approach to the design of social robots is to apply specific capabilities to detect human features, e.g., a face, a body, and skin color, to their vision systems. Such mechanisms usually function well under well-defined conditions, however, they often face problems in unexpected situations and even in presupposed ones. A reason is that applying top-down knowledge develops the frame problem. In contrast, our robot was not embedded with any task-specific or situation-specific capabilities but instead used a fully bottom-up model to interact with humans. The qualitative analysis of the videotaped input image shown in Fig. 4 (a) revealed that the robot had been looking at likely important locations in the interactions. The robot, for example, gazed at an object when a human partner was handling it, and sometimes shifted its attention to the partner's face, which looked as if the robot tried to check the ongoing interaction. We will further analyze how valuable locations can be attended to by the model.

The attention behavior of the robot also gave the impression of a proactive and infant-like agent to the partners. When the robot was distracted by a visual disturbance, some human partners tried to follow the line of its gaze in order to achieve joint attention (see Fig. 5 (a)). Joint attention [26] is a basis of triadic interaction, which expands the communicative space between the robot and a human partner to the third party. Compared to the former studies (e.g., [27, 28, 29]), in which a robot was able to only follow the human gaze, our robot could take the initiative of joint attention. It indicates that our robot was allowed to proactively explore the communicative space, which is considered as an important capability for a social robot. Another valuable finding is that the human partners modified their task-demonstrating actions when they realized the robot's distracted attention. They, for example, exaggerated actions by



(a) iCat, a desktop robot developed by Philips Research. (b) BARTHOC, an anthropomorphic robot.

Fig. 6. Examples of embodied communication robots

making the movement larger and closely showing objects, which are also observed in parental actions toward infants [30]. The phenomena are moreover suggested to help robots as well as infants to learn the actions [31, 23]. Nagai and Rohlfsing [23] have showed that the same attention mechanism as used for our robot can take advantage of parental infant-directed action in robot's action learning. We intend to further investigate how the robot's attention behavior influences the action demonstration of human partners.

7 Conclusion and Future Issues

In designing communicative robots, we can profit from knowledge of HHI. Therefore the focus of this experiment has been a common issue in human communication, i.e., the disturbance. We confronted human partners with a communication robot which was not always attentive but diverted, and collected their reactions to it. The results can be outlined positive: even if the robot was equipped only with a simple attention mechanism, it enabled the partners to treat it as a social partner. They used additional communication channels and increased their utterances to restore the dialog. The effects caused by disturbance reinforced some human partners to help the robot to presume the meaning and the intention of their actions.

For investigating the impact of embodiment, we propose a direct comparison with other robots. Fig. 6 presents two examples: the pet-like iCat and the anthropomorphic BARTHOC. Both robot systems could be equipped with the same attention mechanism and comparative studies could be driven. We suppose a direct comparison would reveal multiple effects.

Acknowledgments. This work has partly been funded by a fellowship of the Sozialwerk Bielefelder Freimaurer e.V.

References

1. Pörksen, B.: Die Gewissheit der Ungewissheit. Gespräche zum Konstruktivismus. In: jedem Augenblick kann ich entscheiden, wer ich bin. Heinz von Foerster über den Beobachter, das dialogische Leben und eine konstruktivistische Philosophie des Unterscheidens, Carl-Auer-Systeme Verlag (2001)
2. Dautenhahn, K., Ogden, B., Quick, T.: From embodied to socially embedded agents - implications for interaction-aware robots, December 2001 (2002)
3. Fong, T., Nourbakhsh, I., Dautenhahn, K.: A survey of socially interactive robots. *Robotics and Autonomous Systems* 42, 143–166 (2003)
4. Goetz, J., Kiesler, S., Powers, A.: Matching robot appearance and behavior to tasks to improve human-robot cooperation. In: Proceedings of the 12th IEEE International Workshop on Robot and Human Interactive Communication, pp. 55–60. IEEE Computer Society Press, Los Alamitos (2003)
5. Minato, T., Shimada, M., Itakura, S., Lee, K., Ishiguro, H.: Evaluating the human likeness of an android by comparing gaze behaviors elicited by the android and a person. *Advanced Robotics* 20(10), 1147–1163 (2006)
6. Mori, M.: Bukimi no tani (the uncanny valley). *Energy* 7(4), 33–35 (1970)
7. Asada, M., MacDorman, K.F., Ishiguro, H., Kuniyoshi, Y.: Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems* 37, 185–193 (2001)
8. Lungarella, M., Metta, G., Pfeifer, R., Sandini, G.: Developmental robotics: a survey. *Connection Science* 15(4), 151–190 (2003)
9. Luhmann, N.: *Social Systems*. Stanford University Press, Stanford, CA (1995)
10. Goodwin, C., Heritage, J.: Conversation analysis. *Annual Review of Anthropology* 19, 283–307 (1990)
11. Schütz, A.: *Collected Papers, The problem of the social reality*. vol. 1, Nijhoff, The Hague (1962)
12. Goodwin, C., Goodwin, M.H.: Concurrent operations on talk: Notes on the interactive organization of assessments. In: *IPRA Papers in Pragmatics*, vol. 1, pp. 1–54 (1987)
13. Goodwin, C.: *Discourse of the Body*, pp. 19–42. Palgrave/Macmillan, New York (2003)
14. Sacks, H., Schegloff, E.A., Jefferson, G.: A simplest systematics for the organization of turn-taking for conversation. *Language* 50, 696–735 (1974)
15. Garfinkel, H.: *Studies in Ethnomethodology*. Prentice-Hall, Englewood Cliffs (1967)
16. Glasersfeld, E.v.: Wege des Wissens. Konstruktivistische Erkundungen durch unser Denken. In: *Fiktion und Realität aus der Perspektive des radikalen Konstruktivismus*, pp. 45–61. Carl-Auer-Systeme Verlag (1997)
17. von Glasersfeld, E.: *Radikaler Konstruktivismus. Ideen, Ergebnisse, Probleme*. Suhrkamp, Frankfurt a. M (1995)
18. Goodwin, C.: *Notes on story structure and the organization of participation*. Cambridge: Cambridge University Press (1984)
19. Ogino, M., Watanabe, A., Asada, M.: Mapping from facial expression to internal state based on intuitive parenting. In: Proceedings of the Sixth International Workshop on Epigenetic Robotics, pp. 182–183 (2006)
20. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(11), 1254–1259 (1998)

21. Itti, L., Dhavale, N., Pighin, F.: Realistic avatar eye and head animation using a neurobiological model of visual attention. In: Proceedings of the SPIE 48th Annual International Symposium on Optical Science and Technology, pp. 64–78 (2003)
22. Kemp, C.C., Edsinger, A.: What can i control?: The development of visual categories for a robot's body and the world that it influences. In: Proceedings of the 5th International Conference on Development and Learning (2006)
23. Nagai, Y., Rohlfling, K.J.: Can motionese tell infants and robots "what to imitate"? In: Proceedings of the 4th International Symposium on Imitation in Animals and Artifacts, pp. 299–306 (2007)
24. Bergmann, J.R.: Handbuch Qualitative Sozialforschung. In: Studies of Work/Ethnomethodologie, München, Psychologie Verlags Union, pp. 269–272 (1991)
25. Kidd, C.D., Breazeal, C.: Effect of a robot on user perceptions. In: Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 28 - October 2, 2004, IEEE/RSJ (2004)
26. Moore, C., Dunham, P.J. (eds.): Joint Attention: Its Origins and Role in Development. Lawrence Erlbaum Associates, Mahwah (1995)
27. Scassellati, B.: Theory of mind for a humanoid robot. *Autonomous Robots* 12, 13–24 (2002)
28. Nagai, Y., Hosoda, K., Morita, A., Asada, M.: A constructive model for the development of joint attention. *Connection Science* 15(4), 211–229 (2003)
29. Nagai, Y., Asada, M., Hosoda, K.: Learning for joint attention helped by functional development. *Advanced Robotics* 20(10), 1165–1181 (2006)
30. Brand, R.J., Baldwin, D.A., Ashburn, L.A.: Evidence for 'motionese': modifications in mothers' infant-directed action. *Developmental Science* 5(1), 72–83 (2002)
31. Rohlfling, K.J., Fritsch, J., Wrede, B., Jungmann, T.: How can multimodal cues from child-directed interaction reduce learning complexity in robot? *Advanced Robotics* 20(10), 1183–1199 (2006)

Natural Language Descriptions of Human Behavior from Video Sequences

Carles Fernández Tena¹, Pau Baiget¹, Xavier Roca¹, and Jordi González²

¹ Computer Vision Centre, Edifici O. Campus UAB, 08193, Bellaterra, Spain

² Institut de Robòtica i Informàtica Ind. UPC, 08028, Barcelona, Spain

{perno,pbaiget,xroca,poal}@cvc.uab.es

Abstract. This contribution addresses the generation of textual descriptions in several natural languages for evaluation of human behavior in video sequences. The problem is tackled by converting geometrical information extracted from videos of the scenario into predicates in fuzzy logic formalism, which facilitates the internal representations of the conceptual data and allows the temporal analysis of situations in a deterministic fashion, by means of Situation Graph Trees (SGTs). The results of the analysis are stored in structures proposed by the Discourse Representation Theory (DRT), which facilitate a subsequent generation of natural language text. This set of tools has been proved to be perfectly suitable for the specified purpose.

1 Introduction

The introduction of Natural Language (NL) interfaces into vision systems is becoming popular, especially for surveillance systems. In these surveillance systems, human behavior is represented by scenarios, i.e. predefined sequences of events. The scenario is evaluated and automatically translated into text by analyzing the contents of the images over time, and deciding on the most suitable predefined event that applies in each case. Such a process is referred to as Human Sequence Evaluation (HSE) in [3]. HSE takes advantage of cognitive capabilities for the semantic understanding of human behaviors observed in image sequences.

This automatic analysis and description of temporal events was already tackled by Marburger et al. [7], who proposed a NL dialogue in German to retrieve information about traffic scenes. More recent methods for describing human activities from video images have been reported by Kojima et al. [6], and automatic visual surveillance systems for traffic applications have been studied by Nagel [8] and Buxton and Gong [2], among others. These approaches present one or more specific issues such as textual generation in a single language, surveillance for vehicular traffic applications only, restrictions for uncertain data, or very rigid environments, for example.

We aim to build a system which addresses the aforementioned drawbacks by following the proposals of HSE, in order to generate NL descriptions of human behavior appearing in controlled scenarios, for several selectable languages. Such

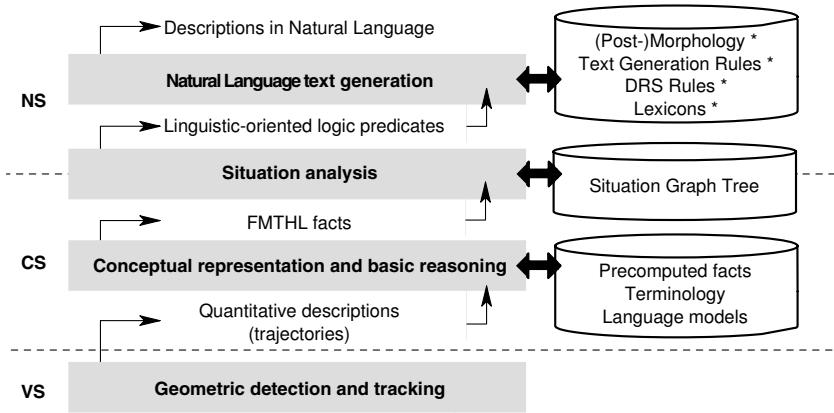


Fig. 1. General schema of the stages and interfaces related to the current text generation system. The left acronyms represent different sub-systems, the boxes describe the main processes that produce changes in data representations, and the right components specify some of the external tools required by the processes. An asterisk remarks that a resource is language-dependent.

a system builds upon three disciplines, namely computer vision, knowledge representation, and computational linguistics. Thus, the overall architecture consists of three subsystems, see Fig. 1: a Vision Subsystem (VS), which provides the geometric information extracted from a video sequence by means of detection and tracking processes, a Conceptual Subsystem (CS), which infers the behavior of agents from the conceptual primitives based on the geometric information extracted by the VS, and a Natural Language Subsystem (NS), which in principle comprises the NL text generation, but also becomes a good stage for providing a complete interface of communication with a final user [8]. Due to space limitations, the extraction of visual information is not treated here. Details can be found, for example, in [10]. We proceed on the basis that structural information consisting of geometrical values are available over time.

The obtention of knowledge derived from visual acquisition implies a necessary process of abstraction. In order to understand the quantitative results from vision, it becomes fundamental to reduce the information to a small number of basic statements, capable of detecting and relating facts by means of qualitative derivations from what has been ‘seen’. The conversion of observed geometrical values over time into predicates in a fuzzy logic formalism allows to reach an intermediate state, the conceptual representation layer, which facilitates schematic representations of the scenarios [1] and, in addition, enables characterizations of uncertain and time-dependent data extracted from image sequences. Next, a classification can be performed by integrating these resulting facts into preconceived patterns of situations. Such an inference system produces not only an interpretation for the behavior of an agent, but also reasons for its possible reactions and predictions for its future actions [4].

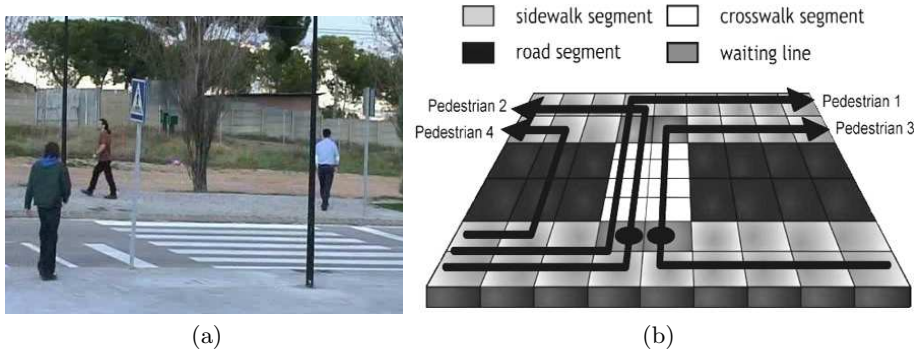


Fig. 2. Original pedestrian crosswalk scene (a) and groundplane schematic map of the main regions considered in this scene (b). Pedestrian trajectories have been included in the scheme. Black circles represent a stop on the waiting line.

Discourse Representation Theory seems to be of particular interest for the conversion from conceptual to linguistic knowledge, since it discusses algorithms for the translation of coherent NL textual descriptions into computer-internal representations by means of logical predicates [5]. The reverse step is also possible, so that the results of the conceptual analysis are stored into semantic containers, the so-called Discourse Representation Structures (DRS), which facilitate the construction of syntactical structures containing some given semantic information. A final surface realization stage over these preliminary sentences embeds the morphological and orthographical features needed for obtaining final NL textual descriptions.

Next chapter describes the chosen scenario, and explains how the evaluation of human behaviors is achieved from spatiotemporal data and prior knowledge about the scene. Section 3 details the mechanisms which convert high-level predicates obtained from situational analysis into NL textual descriptions. Some experimental results for Catalan, English, and Spanish are shown in Section 4. Finally, Section 5 concludes the paper and suggests future lines of work.

2 Evaluation of Human Behaviors in Video Sequences

The chosen scenario for evaluation of basic human behaviors has been a crosswalk, see Fig. 2. On it, a certain number of pedestrians, each one with a different behavior, start from one of the sidewalks and cross the road to get to the other side. At first, the presence of traffic vehicles has been omitted.

2.1 The Conceptualization Step

The structural knowledge acquired by the VS needs to be abstracted and converted into logic knowledge in order to facilitate further manipulations and reasonings. To do so, trajectories and other types of estimated spatiotemporal

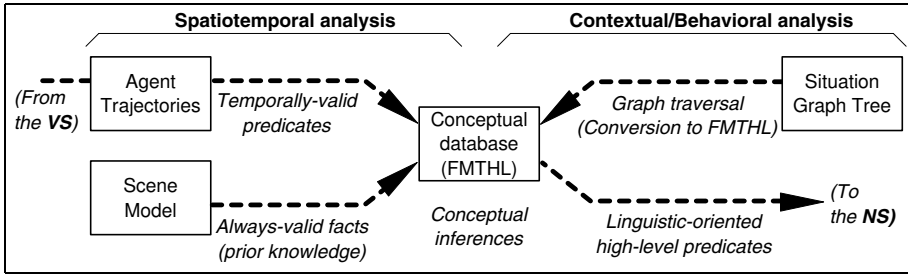


Fig. 3. Quantitative data (e.g. trajectories from the agents) is evaluated upon the predefined facts from the scene model, and thus converted into qualitative FMTHL knowledge. On the other hand, the behavioral model encoded into a SGT is traversed and converted into FMTHL conditions, too. Finally, the entire set of asserted spatiotemporal qualitative results is logically classified by the traversed SGT, and *linguistic-oriented predicates* are generated as a result.

information are associated to basic concepts identifying recognizable simple actions, which can be described by using elementary verb-phrases (e.g., ‘approaching_to_location’, ‘turning’, ‘has_speed’). These *conceptual predicates* are not yet proper linguistic expressions, but system-internal representations resulting from classification and abstraction processes.

Fuzzy Metric Temporal ‘Horn’ Logic (FMTHL) has been conceived as a suitable mechanism for dealing with uncertain, time-dependent information [11]. This formalism allows to represent knowledge explicitly and hierarchically, not coded into conditional probabilities, and enables to manage data requiring both *temporal* and *fuzzy* properties [4]. In our case, observed trajectories from the agents are analyzed within a predefined scene model, see Fig. 3. As a result, geometrical, quantitative values are acquired, such as postures, velocity, or positions for the agents. After an abstraction process is carried out, the reasoning system is conferred a capability for representing uncertain qualitative descriptions inferred from the quantitative data. The logic productions evolve over time as the received data does, so this conceptual knowledge is also time-delimited, and thus the development of events can be comprehended and even anticipated.

The qualitative knowledge extracted from quantitative results is encoded using these fuzzy membership functions, so that the generated predicates are related to conceptual ‘facts’ for each time-step. For example, a collection of positions over time allows to derive fuzzy predicates such as ‘has_speed(zero)’, ‘has_speed(small)’, or ‘has_speed(very_high)’, depending on the displacements of the agent detected between consecutive points of time.

2.2 Agent Trajectories

Trajectory files are ordered collections of observed values over time for a certain agent, which are obtained as a result of the tracking processes for the agents [10]. From the evolution of the states of the agent, a certain *behavior*, i.e. a sequence

of situated actions, will be assumed. Four agent trajectories have been obtained, which consist of a set of FMTHL logical predicates of type `has_status`. These predicates comprise the required knowledge for the human behavior analysis in the following scheme or *status vector* for the agent at time t :

$$t ! \quad \text{has_status}(\text{Agent}, X0, Y0, \text{Theta}, \text{Vel}).$$

As can be seen, the `has_status` predicates for the interpretation of human actions contain five fields so far, all of them being identifiers to entities and objects detected during the tracking process, or otherwise concrete geometrical values for spatiotemporal variables. The `Agent` field gives information about the name given to the agent. The rest of the fields give quantitative values to the geometrical variables needed: 2-D spatial position in the ground plane (`X0`, `Y0`), angle of direction (`Theta`), and instant velocity (`Vel`). The `Vel` field provides the necessary information for determining the action being performed by the agent (i.e. *standing*, *walking*, *running*).

2.3 Scene Modeling

The scenario in which pedestrians perform their actions has been included as an additional source of knowledge for the reasoning stage. The geometrical modeling of the location has been done first in a ground plane bidimensional approach, so a set of spatial descriptors are declared to distinguish the relevant topographic or interesting elements in the scene, see Fig. 2 (b). This source provides the spatial distribution taken into account for the given situation.

A second source of knowledge contains other logical statements that will confer semantic significance on the initial geometrical descriptors of the scene. The different regions can be enclosed into different categories (*sideway*, *road*, *crosswalk*) and can also be given different attributes (*walking zone*, *waiting line*, *exit*). This step is necessary for identifying significative regions, so the movements and interactions of the agents can be contextualized by means of valid identifiers. These geometrical considerations have been encoded using FMTHL predicates.

2.4 Situation Graph Tree

Situation Graph Trees (SGTs) are hierarchical structures used to model the knowledge required from human behavior analysis in a specific discourse domain [3]. A SGT has been designed for the crosswalk scene, see Fig. 4. The conceptual knowledge about a given actor for a given time step is contained in a so-called *situation scheme*, which constitutes the basic components of a SGT. The knowledge included in these components is organized in two fields: *state predicates* and *action predicates*.

- First, a set of logic conditions describes the requirements that need to be accomplished to instantiate that situation. The assertion of these *state predicates* is performed by evaluating the semantic predicates inferred from the agent status vectors obtained at the visual stage.

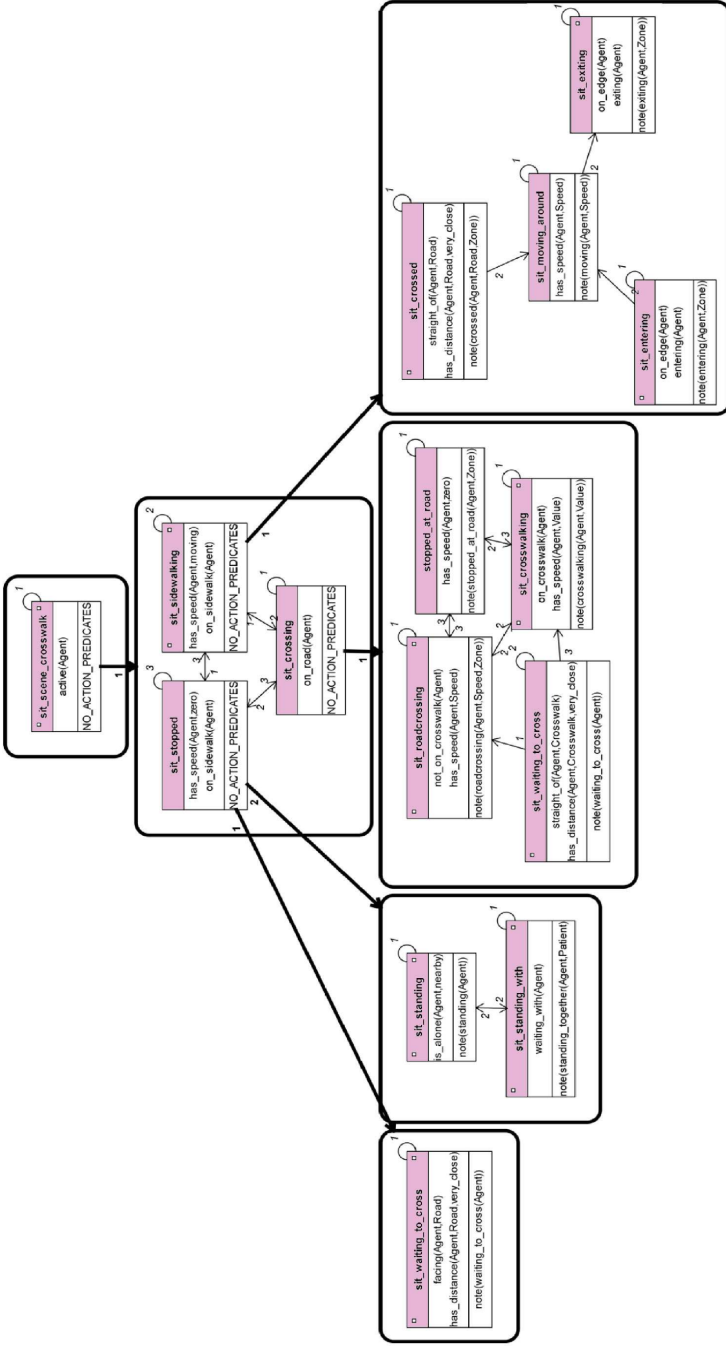


Fig. 4. Situation graph tree describing the behaviors of pedestrians on a crosswalk. Situation graphs are depicted as rounded rectangles, situation schemes are shown as normal rectangles. Bold arrows represent particularization edges, thin arrows stand for prediction edges, and $\frac{3}{4}$ -circle arrows indicate self-predictions. Small rectangles to the left or to the right of the name of situation schemes mark that scheme as a start- or end-situation, respectively [4]. A SGT needs to focus on *behaviors* of the agents, while avoiding dependance to a particular scenario. The more this approach is achieved, the more flexible and scene-independent the SGT will be.

- After the conditions have been asserted, certain domain-specific *action predicates* are generated and forwarded for defined purposes. Only generation of NL text will be considered here, so *linguistic-oriented* logic predicates will be generated (*note* commands in Fig. 4).

A single SGT incorporates the complete knowledge about the behavior of agents in a discourse [1]. Every possible action to be detected has to be described in the SGT. Consequently, it is necessary to have accuracy to precisely identify the desired actions, but it is also important that it does not become excessively complex in order to avoid a high computational cost. On the other hand, the SGTs are transformed into logic programs of a FMTL for automatic exploitation of these behavior schemes, as shown in Fig. 3.

Depending on the behavioral state, a new high-level predicate will be sent to the NS Subsystem, by means of a *note* method. The new predicates offer language-oriented structures, since their attribute scheme comprises fields related to ontological categories such as *Agent*, *Patient*, *Object* or *Event*. These predicates are the inputs for the NS Subsystem, which will be discussed next.

3 Linguistic Implementation

It is in the NS where the logical predicates are used to provide the representational formalism, making use of the practical applications of the Discourse Representation Theory. Inside the NS layer, there are several stages to cover:

1. Lexicalization
2. Discourse Representation
3. Surface Realization

Besides, the set of lemmata been used has to be extracted from a restricted corpus of the specific language. This corpus can be elaborated based upon the results of several psychophysical experiments on motion description, collected over a significative amount of native speakers of the target language. In our case, ten different people have independently contributed to the corpus with their own descriptions of the sample videos. Three different languages have been implemented for this scenario: Catalan, English, and Spanish.

3.1 Generation of textual descriptions

The overall process of generation of NL descriptions is based on the architecture proposed by Reiter & Dale [9], which includes three modules; a document planner, a microplanner, and a surface realizer (see Fig. 5). The VS provides the information to be communicated to the user; this task is considered to be part of the Document Planner. The CS decides how this information needs to be structured and gives coherency to the results. This module provides general reasoning about the domain and determines the content to be included in the sentences to be generated, which are tasks related to the Document Planner, too. Further tasks, such as microplanning and surface realization, are included into the NS. An example for the entire process of generation is shown in Fig. 6.

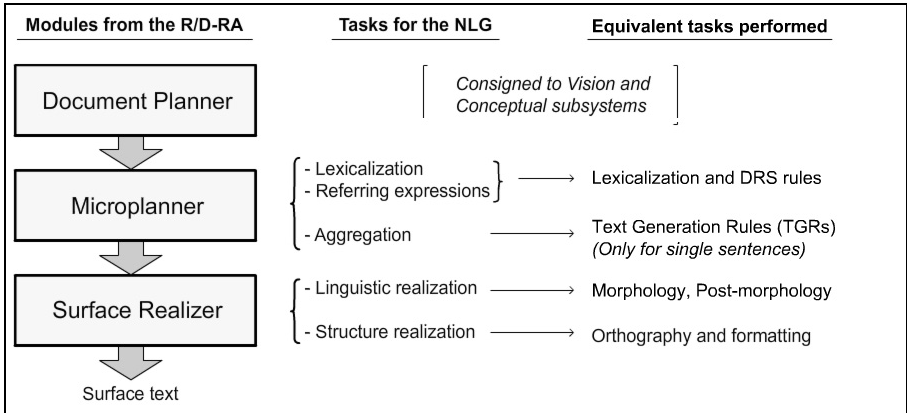


Fig. 5. Schema of Reiter/Dale Reference Architecture (R/D-RA) [9], including the tasks related to each module that are necessary for a Natural Language Generator

Lexicalization. It is necessary to convert the abstract predicates from the CS into linguistic entities for communication, such as agents, patients, objects, or events, for instance. The classification of linguistically-perceived reality into thematic roles (e.g. agent, patient, location) is commonly used in contemporary linguistic-related applications as a possibility for the representation of semantics, and justifies the use of computational linguistics for describing content extracted by vision processes. The lexicalization step can be seen as a *mapping process*, in which the semantic concepts identifying different entities and events from the domain are attached to linguistic terms referring those formal realities. This way, this step works as a real dictionary, providing the required lemmata that will be a basis for describing the results using natural language.

Representation of the Discourse. Nevertheless, bridging the semantic gap between conceptual and linguistic knowledge cannot be achieved only with a lexicalization step. Discourse Representation Structures (DRSs) are the actual mechanism that facilitates to overcome the intrinsic vagueness of NL terms, by embedding semantics inferred at the conceptual level into the proper syntactical forms. Lemmata are just units that will be used by these structures to establish the interrelations which will convey the proper meaning to the sentences.

DRSs are semantic containers which relate referenced conceptual information to linguistic constructions [5]. A DRS always consists of a so-called *universe* of referents and a set of conditions, which can express characteristics of these referents, relations between them, or even more complex conditions including other DRSs in their definition. These structures contain linguistic data from units that may be larger than single sentences, since one of the ubiquitous characteristics of the DRSs is their semantic cohesiveness for an entire discourse.

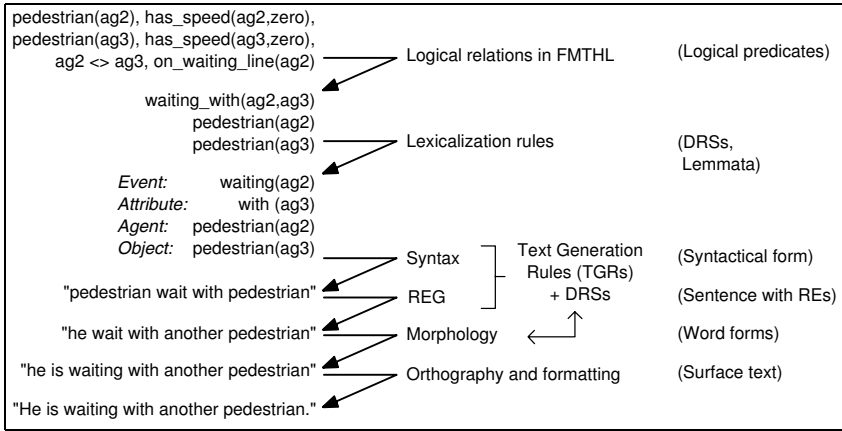


Fig. 6. Example for the generation of the sentence ‘‘He is waiting with another pedestrian’’ from logical predicates. The center column contains the tasks being performed, and the right column indicates the output obtained after each task.

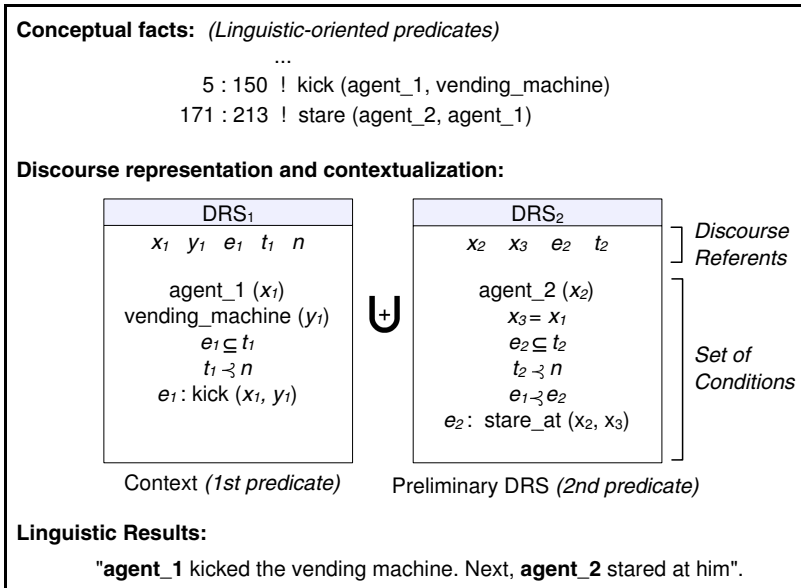


Fig. 7. A pattern DRS allows to convert a stream of FMTHL into a string of textual symbols. Here, two predicates are validated. The first one instantiates a DRS, which serves as context for the following asserted facts. Once the new predicate is validated, it instantiates another DRS which merges with that context, thus providing a new context for subsequent facts. The temporal order of the events is stated by relating them to time variables ($e_1 \subseteq t_1$), placing these variables in the past ($t_1 \prec n$), and marking precedence ($e_1 \prec e_2$).

When a contextual basis is explicitly provided, the maintenance of the meaning for a discourse, including its cross-references, relations and cohesion can be granted. Then, linguistic mechanisms such as anaphoric pronominalization for referring expressions can be successfully implemented, e.g. ‘*The pedestrian is running*’ \rightarrow ‘*He is running*’. In our case, since situational analysis is performed individually for every detected agent, we base on previously mentioned information about the focused agent to decide whether to use pronouns or full descriptions. An example which shows how the semantic representation and contextualization is undertaken by a DRS is illustrated in Fig. 7.

DRSs facilitate the subsequent tasks for sentence generation. The syntactical features of a sentence are provided by the so-called Text Generation Rules (TGRs), which establish the position for the elements of the discourse within a sentence for a particular language. Due to the specific goals considered for this system, several assumptions have been taken: we use simple sentences for effective communication

Surface Realization. The Surface Realization stage is accomplished in two steps. A first morphological process applies over each single word and partially disambiguates the individual abstraction of that word, by means of morphological attributions such as gender or number. These attributions can be propagated upon the semantic relations previously established by DRSs among the lemmata of a single piece of discourse. After that, a set of post-morphological rules has been conceived to enable interactions among predefined configurations of words, thus affecting the final surface form of the text. This additional step is indispensable for many languages, in which certain phenomena force the surface form to change, e.g. contractions (‘*a*’+‘*el*’ \rightarrow ‘*al*’ in Spanish), or order variation (‘*es*’+‘*va*’+‘*en*’ \rightarrow ‘*se’n va*’ in Catalan).

4 Experimental Results

We address the problem in-depth for a particular domain, instead of finding a generically-applicable solution. For this reason, an ad-hoc solution has been chosen for the identification of a predefined set of behaviors in the described scenario. In such a framework, situations are specialized as long as spatiotemporal information can be classified by the given models. If a non-modeled situation occurs, the SGT cannot specialize a concrete interpretation, and instead of this it generates a more general description, e.g. pedestrians being detected in certain regions, or agents grouping or splitting. A group of native speakers provided linguistic interpretations for the set of behaviors, for each individual language considered.

Thus, the coverage of the generated descriptions is tightly related to the extent of situations modeled by the SGT. A vertical growing of this classifier, i.e. an increment of the particularization edges, increases the granularity of the descriptions, thus disambiguating or specializing the discourse. On the other hand, by enhancing the human motion models for the scene and the prediction edges, we



Pedestrian 3 (Catalan)

- 203** : *Lo vianant surt per la part inferior dreta.*
252 : *Va per la vorera inferior.*
401 : *S'espera per creuar.*
436 : *S'està esperant amb un altre vianant.*
506 : *Creua pel pas zebra.*
616 : *Va per la vorera superior.*
749 : *Se'n va per la part superior dreta.*

Pedestrian 3 (English)

- 203** : *The pedestrian shows up from the lower right side.*
252 : *He walks on the lower sidewalk.*
401 : *He waits to cross.*
436 : *He is waiting with another pedestrian.*
506 : *He enters the crosswalk.*
616 : *He walks on the upper sidewalk.*
749 : *He leaves by the upper right side.*



Pedestrian 4 (Spanish)

- 523** : *El peatón aparece por la parte inferior izquierda.*
572 : *Camina por la acera inferior.*
596 : *Cruza sin cuidado por la calzada.*
681 : *Camina por la acera superior.*
711 : *Se va por la parte superior izquierda.*

Pedestrian 4 (English)

- 523** : *The pedestrian shows up from the lower left side.*
572 : *He walks on the lower sidewalk.*
596 : *He crosses the road carelessly.*
681 : *He walks on the upper sidewalk.*
711 : *He leaves by the upper left side.*

Fig. 8. Some of the descriptions in NL which have been generated for the crosswalk scene. The results match perfectly with the purposed set of natural language sentences suggested by a group of native speakers of the given languages.

increase the number of possible situations and their temporal structure respectively. Finally, the quality of the textual corpora provided by native speakers, which link linguistic patterns to the conceived situations, determines the goodness of the discourse representation. A set of deterministic linguistic rules were designed so that results matched perfectly with the selection of the descriptions provided by native users.

Some results for the situation analysis of the crosswalk scene are shown in Fig. 8. Textual descriptions in Catalan, English, and Spanish have been selected for Agents 3 and 4, respectively. These descriptions include agents appearing or leaving the scene, interactions between pedestrians and locations within the scenario (crosswalk, sidewalks), and interpretations for some detected behaviors, such as waiting with other agents to cross, or crossing in a dangerous way (i.e. directly by the road and not caring for vehicular traffic). Only static cameras were used in this first step, so no expressions concerning the action of the cameras are generated. Next improvements should focus on the semantic content provided by the behavioral and inference subsystems, i.e. which situations must be considered for a certain domain and scenario, and in which way the reasonings for these situations have to be done. Further approaches will lead to more complex requirements regarding linguistic capabilities, which have been restricted so far.

5 Conclusions

A system that evaluates video sequences involving human agents by generating NL descriptions in multiple languages has been successfully developed in a first stage. A brief overview of the tasks performed may help to understand how the generated text contributes to the goal of human behavior evaluation. After the conceptualization of the spatiotemporal information is achieved, and basic inferences are done, SGTs are in charge of integrating the deduced semantic knowledge. Also, contextual and behavioral models are applied here, since SGTs can be seen as actual classifiers of content for situations in a definite domain. The generation of NL is built upon the high-level semantic predicates generated by a SGT. In some way, the generated descriptions are *interpretations* of this semantic knowledge accomplished by native speakers of a certain language. The group of native speakers choose the linguistic expressions they find more appropriate, in order to incorporate the situations from a SGT into a suitable discourse. Hence, the situations appearing in a video sequence from a given domain can be interpreted and described in multiple natural languages.

The current NS allows for a flexible and fast incorporation of languages into a facility for multilingual generation of textual descriptions in NL. The natural language formalism makes possible to generate fluid rich sentences to the user, allowing for detailed and refined expressions that are not possible by using other mechanisms. The interconnection of all the stages involved in the system has been proved as convenient for the whole process of evaluation, although several gaps still have to be solved. Further steps should include the extension of current behavioral models, the detection of groups and more complex interactions

among agents and/or vehicles, and the use of uncertainty for not only predicting behaviors, but also to enhance possible hypothesis of interpretation for the detected events within the scene.

Lastly, results from NL texts can be interpreted as semantic tags to provide content segmentation of the video sequences over time. We are currently studying the connection of a user interaction stage accepting input NL-based queries to a large database of video sequences, generic or specific. This will be the starting point for search engines capable of retrieving video sequences showing specific motion or factual contents. In addition to this, the segmentation of video sequences into time-intervals showing cohesive information can be applied for extracting a collection of few semantic shots from these sequences. This way, a compression of the relevant information – user-definable and freely configurable by declaring attentional factors – can be done by summarizing the entire videos with a list of behavior concepts. Thus, we aim to improve motion description patterns for video standards such as MPEG-7, thus allowing for high-level annotations related to the motion within the scene.

Acknowledgements

This work has been supported by EC grant IST-027110 for the HERMES project and by the Spanish MEC under projects TIC-2003-08865 and DPI-2004-5414. Jordi Gonzàlez also acknowledges the support of a Juan de la Cierva Postdoctoral fellowship from the Spanish MEC.

References

1. Arens, M., Nagel, H.H.: Representation of Behavioral Knowledge for Planning and Plan-Recognition in a Cognitive Vision System. In: Jarke, M., Koehler, J., Lake-meyer, G. (eds.) KI 2002. LNCS (LNAI), vol. 2479, pp. 268–282. Springer, Heidelberg (2002)
2. Buxton, H., Gong, S.: Visual surveillance in a dynamic and uncertain world. *AI-magazine* 78(1), 431–459 (1995)
3. Gonzàlez, J.: Human Sequence Evaluation: The Key-Frame Approach. PhD thesis, Universitat Autònoma de Barcelona, Barcelona, Spain (2004)
4. Haag, M., Theilmann, W., Schäfer, K., Nagel, H.H.: Integration of Image Sequence Evaluation and Fuzzy Metric Temporal Logic Programming, pp. 301–312. Springer, London, UK (1997)
5. Kamp, H., Reyle, U.: *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht, Boston, London (1993)
6. Kojima, A., Tamura, T., Fukunaga, K.: Natural language description of human activities from video images based on concept hierarchy of actions. *International Journal of Computer Vision* 50(2), 171–184 (2002)
7. Marburger, H., Neumann, B., Novak, H.J.: Natural Language Dialogue about Moving Objects in an Automatically Analyzed Traffic Scene. In: Proc. IJCAI-81, Vancouver (1981)
8. Nagel, H.H.: Steps toward a Cognitive Vision System. *AI-Magazine* 25(2), 31–50 (2004)

9. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press, Cambridge/UK (2000)
10. Rowe, D., Rius, I., Gonzalez, J., Villanueva, J.J.: Improving Tracking by Handling Occlusions. In: Singh, S., Singh, M., Apte, C., Perner, P. (eds.) ICAPR 2005. LNCS, pp. 384–393. Springer, Heidelberg (2005)
11. Schäfer, K., Brzoska, C.: F-Limette Fuzzy Logic Programming Integrating Metric Temporal Extensions. *Journal of Symbolic Computation* 22(5-6), 725–727 (1996)

Detecting Humans in 2D Thermal Images by Generating 3D Models

Stefan Markov and Andreas Birk

School of Engineering and Science
Jacobs University Bremen*
Campus Ring 1, D-28759 Bremen, Germany
a.birk@iu-bremen.de

Abstract. There are two significant challenges to standard approaches to detect humans through computer vision. First, scenarios when the poses and postures of the humans are completely unpredictable. Second, situations when there are many occlusions, i.e., only parts of the body are visible. Here a novel approach to perception is presented where a complete 3D scene model is learned on the fly to represent a 2D snapshot. In doing so, an evolutionary algorithm generates pieces of 3D code that are rendered and the resulting images are compared to the current camera picture via an image similarity function. Based on the feedback of this fitness function, a crude but very fast online evolution generates an approximate 3D model of the environment where non-human objects are represented by boxes. The key point is that 3D models of humans are available as code snippets to the EA, which can use them to represent human shapes or portions of them if they are in the image. Results from experiments with real world data from a search and rescue application using a thermal camera are presented.

1 Introduction

Machine detection of humans is an integral part of many AI applications ranging from traffic surveillance [PWB⁺05, PJHK99] over home security [CGPV05] to robots serving as guides in museums [TBB⁺99]. While having such a wide application range and hence a large amount of contributions, it is at the same time a very complex task due to the high variability of human beings. People do not only look quite differently among different individuals, but also the same person can have very different appearances based on pose, clothing, environment conditions, and so on [MPP01, Spa].

Current human detection algorithms can be coarsely classified by three main categories. First, model-based methods where a general model is matched to different parts of an image trying to find a fit [Yui91]. Second, image-invariance systems, which base the matching on a set of image pattern relationships that uniquely determine the object [Sin94]. And finally example-based algorithms

* International University Bremen until spring 2007.

that learn the detection by being trained on a set of positive and negative examples [MPP01, OPOP97, PP00, YC97]. In addition, these techniques most commonly employ preprocessing steps for figure ground separation. Common examples are differential imaging in order to detect the silhouettes of human beings from motion, or color analysis in order to determine different parts of the human body [AT04, KM00, Spa, WADP97, MPP01, Hog83].

Due to the high complexity of the problem the above algorithms impose restrictions on the environment in which they will be detecting humans or on the to-be-detected humans themselves. Example of such constraints are: greater dynamics of people relative to the background [Spa, WADP97], only one person may be in the view of the camera [PP00], restrictions on the lightning dynamics [WADP97], limitations to the people's poses and the circumvention of occlusions [PP00]. One special constraint, which spans a whole sub-field of its own, is to concentrate on the human face for detection and recognition purposes [ZCPR03, YKA02, YSMW98, VAJOWC94].

These assumptions limit the applicability of the algorithms, e.g., in application domains like search and rescue robotics, where people that have to be detected can exhibit a number of these restrictions. The approach presented here tries to overcome some of the constraints and thus to extend the application area of human detecting systems. It is based on so-called reproductive perception that is able to recognize large scenes of known objects quickly even with partial occlusion between the objects [Bir96]. The main idea is to use a set of programs, which generate data that reproduces the vast amounts of sensor data. In doing so, the goal is to find a small programs as dense representations of large amounts of input data [BJP94, PS95].

Here, the programs as representations are based on a scene description language using OpenGL. Code from the 3D scene language can be rendered to generate pseudo sensor data, i.e., 2D images like the ones coming from the thermal camera used in the application presented here. The goal is to generate 3D code that generates pseudo sensor data corresponding to the current input data from a camera.

2 Background

Our interest in the problem is motivated by work on intelligent behaviors up to full autonomy on robots for search and rescue missions. Existing fieldable systems in this domain are optimized for core locomotion [SC04, Mur04, Dav02] and they provide sensor streams via plain tele-operation [Sny01, Abo98]. But adding intelligent functions can significantly improve the usability of the devices [BC06, MCM01], e.g., to ease the large cognitive load on human operators [SYDY04], or even for fully autonomous behavior to overcome drop outs in the communication systems or to allow a single operator to handle a whole team of robots.

The latest type of robots from Jacobs University are the so-called Rugbots, short for "rugged robot" [BPSC06]. The robots are capable of fully autonomous operation as demonstrated at the RoboCup 2006 world championship

[BMDP06]. The RoboCup rescue league [KT01] features a very challenging environment (Fig. 1) including several standardized test elements that can be used to assess the quality of the robots [JWM03, JMW+03].

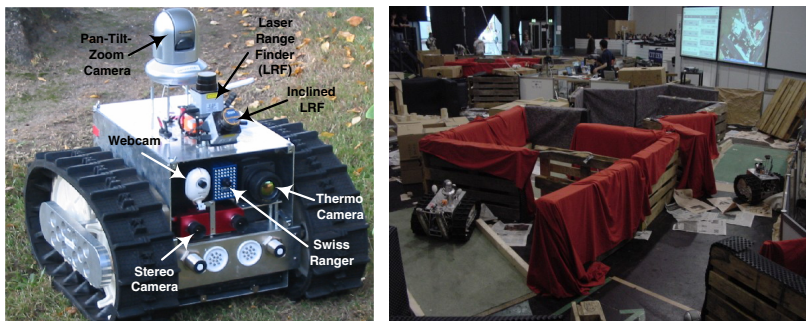


Fig. 1. On the left, the autonomous version of a *Rugbot* with some important onboard sensors pointed out, especially the thermal camera. On the right, a photo from the 2006 RoboCup world championship where the team from Jacobs University demonstrated a combined usage of a tele-operated with a fully autonomous robot.

The automatic detection of human victims in search and rescue applications is not only of interest for fully autonomous systems. It can also be a very helpful feature to ease the task for a human tele-operator. Common systems rely on visual inspection of video streams by a human operator [Dav02], a tedious and error-prone task [SYDY04]. Attempts to automatically detect humans under the challenging circumstances of rescue missions include chemical sensors [TGP02], template matching using stereo vision [BINS05], and a variety of thermal sensors [HBG+06, AHF+05, Bur98].

Temperature sensing is an interesting option for search and rescue missions. Especially, thermal cameras already provide simple processing options. This is very useful for segmentation in particular. The on-camera processing can for example be used to highlight candidate data ranges through suited color palettes. Thermal imaging can therefore also be beneficial for systems where a human operator inspects the data. But the temperature ranges that have to be classified as "interesting" are relatively large; uncovered skin is much warmer than clothed body parts, environmental effects play a role, and so on. Pure temperature as indicator for the presence of humans leads to hence many false positives (figure 2). It is therefore important to take shape into account to get a reliable, automatic detection.

The device used in the work presented in the following sections is a Flir A20 thermal camera. It has an uncooled, high resolution Focal Plane Array (FPA). Its 160x120 imager elements provide temperature information in a range of -40°C to 120°C . Humans emit about 36°C only at exposed body parts. Parts covered by clothes, dust, and so on appear colder. Hence a range of 28°C to 38°C is segmented to be potentially human. Anything else is considered as background

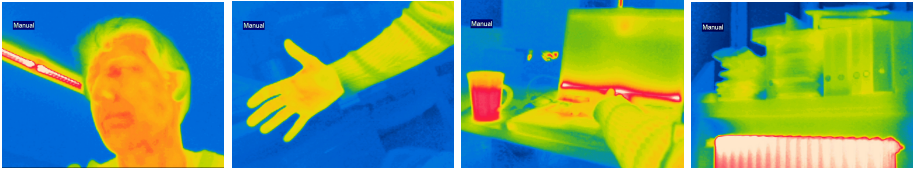


Fig. 2. Thermal imaging can ease the detection of humans, but it does not provide a perfect solution without taking object shape into account

temperature. A thresholded bitmap with two colors is produced: the background temperature illustrated by black and temperatures in the human range indicated by blue.

3 The 3D Scene Representation

A core part of the approach presented here is the set of programs, which generate images that describe the perceived environment. The main idea is to model the environment as a collection of 3D humans and boxes distributed in space that are projected to a 2D image using OpenGL. The boxes serve as simple objects to generate occlusions (if they are "cold") and false positives (if they are "hot"). Arbitrarily complex non-human shapes can be composed from combining boxes. The parameters of the OpenGL camera model are roughly based on the parameters of the real camera on the Jacobs RugBot. An exact calibration is not necessary as no metric information is extracted from the scenes.

A human is created as a composition of its basic body parts – head, torso, arms, legs. The developed human model has 14 rotational joints through which it can mimic almost any pose of a real human. In addition it has six more degrees of freedom (DOF) for the position and orientation of the torso in space. The dimensions of the body parts are fixed, since one can represent a taller human with a smaller one which is closer to the camera and vice versa. Figure 3 displays the output of a sample program for the case of drawing a whole human and for

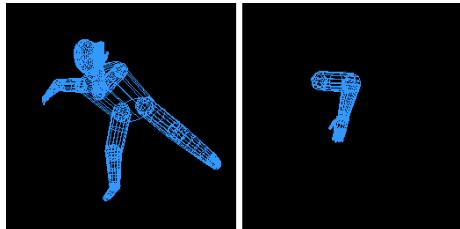


Fig. 3. The 2D rendering of a 3D model of a human and an arm drawn in wireframe mode

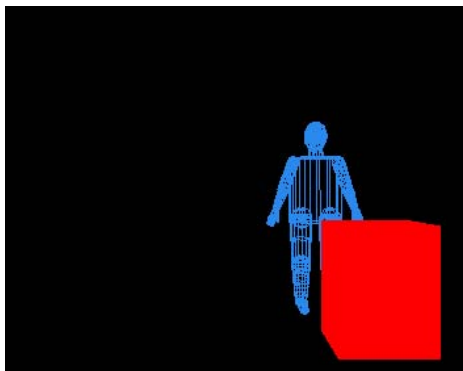


Fig. 4. The output of a sample 3D drawing program. The human is drawn in wireframe mode for illustration purposes. When matching renderings to the thermo-images a uniform color texture is used for humans. Note that most of the time, they boxes have a dark color like the background, i.e., they are at room temperature. These dark boxes are mainly used to represent occlusions. "Hot" boxes can be used to create false positives.

only an arm. Boxes as the only other components of the 3D scenes are simply defined by their dimensions and positions in space.

Based on the OpenGL routines for drawing humans and boxes a complete drawing program is created as a set of calls to these functions. Each call can be defined as an *instruction*. An instruction places the corresponding model on a 3D scene and after all calls are executed the projection of the drawn scene is taken and returned as the output of the program, as shown in figure 4.

4 Image Distance Function

The next step is to be able to compare how well the OpenGL-generated images reproduce an input image from the infrared camera. To achieve this some initial image processing on the input image is done first. The main goal of the pre-processing is to eliminate noise and segment the input image by converting the 24-bit bitmap to a thresholded bitmap with two colors: the room temperature illustrated by black and temperatures in the human range indicated by blue. Humans emit about 36°C only at exposed body parts. Parts covers by clothes, dust, and so can appear colder. Hence a range of 28°C to 38°C is segmented to be potentially human. Anything else is considered as room temperature. Figure 5 shows the result from applying this segmentation to an input image.

To generate the models, there is the need for measuring how well a pre-processed camera image a' matches to the output a of the OpenGL program.

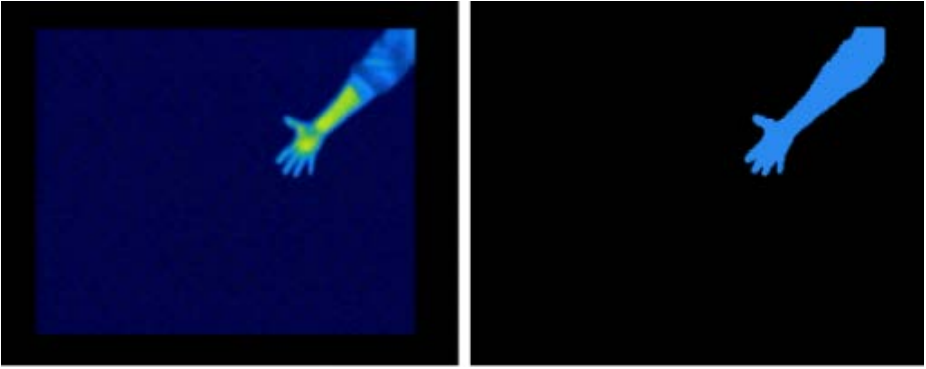


Fig. 5. On the left is the original bitmap from the thermo camera. On the right is the same image after segmenting everything in the range of 28°C to 38°C to be potentially human.

For this purpose an image distance function which has been defined in [Bir96] is used:

$$D(a, a') = \sum_{c \in C} d(a, a', c) + d(a', a, c) \quad (1)$$

$$d(a, a', c) = \frac{\sum_{a[p_1]=c} \min(md(p_1, p_2) \mid a'[p_2] = c)}{\#_c(a)}$$

where

- c is a color ($c \in C$, C denotes the set of all colors)
- $a[p]$ is the color at position $p = (x, y)$ in a
- $md(p_1, p_2)$ is the Manhattan distance between p_1 and p_2
- $\#_c(a) = \#\{p_1 \mid a[p_1] = c\}$, i.e. the number of pixels in a with color c .

The lower the value of the image distance function, the better the match between the two images. Furthermore, this function has two very important properties, which make it very suitable for the task in hand:

- It gives proper gradients with respect to translation, rotation and scaling of objects.
- It can be computed in a time that is linear in the number of pixels in a with color c .

5 The Evolutionary Algorithm

Up to now we have programs that generate images, also called hypotheses, which attempt to predict the robot's environment and we can evaluate how good these predictions are. The last component of the approach is an evolutionary algorithm

that works on a set of such programs and attempts to improve the validity of the generated hypotheses by altering the programs themselves. An evolutionary algorithm (EA) in general is characterized by a population, a fitness function and selection and transformation operators used to evolve the population.

The population consists of so-called individuals, which here are programs that define a complete 3D scene as described in section 3. Furthermore, each program stores the image that is produced after the execution of all its instructions, i.e., the hypothesis defined by the program. The population at any particular time step of the EA is called a generation. The fitness of an individual is defined via the image distance function between the output of the individual and the pre-processed infrared image. In this way, lower fitness value indicates that an individual is highly fit and vice versa.

At each iteration of the EA a subset from the population is chosen by the selection operators to evolve through the transformation operators. Individuals that are highly fit are more likely to yield superior ones after transformation [S.F93], hence they should be selected with higher probability. For this purpose, roulette selection is used. Roulette selection is a randomized variant of fitness-proportionate selection. Each individual is selected with a probability determined by its fitness. However, since in our case low fitness indicates a better individual we are using the inverse of the fitness value in order to implement the roulette selection.

The transformation operators are responsible for "evolving" the selected individuals, thus producing generation S_i from generation S_{i-1} . For the task in hand the main operator that we apply is hill-climbing. Hill climbing takes a single individual, chooses a drawing instruction to optimize, then selects a DOF from this drawing instruction and then explores a space of nearby values to the current one in order to return a better new individual. For now, we will consider the case where we have only a single human drawing instruction per individual.

Due to the many DOF of the human model, hill climbing will perform very inefficiently if the to-be-optimized DOF is selected randomly. To improve this we assign a-priori probabilities to each of the 14 joints, position and orientation, based on our expectation of how well modifying this joint/DOF will improve the quality of the individual. For example, position in 3D space is expected to greatly influence the fitness of an individual, the same applies to rotation and the upper leg joints, hence they are assigned high probabilities. On the other hand, the elbow and knee joints could be assigned lower probabilities since they do not have that much influence on the fitness of an individual. Based on these probabilities, we run roulette selection on the joints/DOF of the human in order to select a variable that will be optimized. After that, we explore a space of m neighboring values and select the one that has improved the fitness most.

Further improvement of the hill climbing is to re-apply it on the newly created individual, if it is better than its parent. This addition significantly improved the performance of the HRS since now within a single iteration of the EA we can bring a human model in an individual which is far away in space, for example, close to the target position of the human on the infrared image. For the case

where we have more than a one human in the scene, we first randomly select the instruction to be optimized and then apply the above-described algorithm.

After hill climbing has returned another individual we have to decide how it will be inserted in the population, so that the next generation is formed. We have several choices – inverse roulette selection of the individual that will be replaced, replacement of the parent if better or replacement of the worst individual in the population. The one that showed best results is replacement of the parent in the case when the child has a better fitness. This can be easily explained with the specifics of the problem. If we replace the worst program, or we use inverse roulette selection, the population will easily converge to a set of good individuals, from which however could be impossible to produce a very good match, i.e. we easily reach a local optimum. On the other hand, a bad program can be easily made the best by multiple-step hill climbing on position, hence we should not replace bad programs, but rather just the parents when an improvement has been found.

The performance of an EA depends to some extent on the initial population. When creating it purely randomly it often happens that there are not any really good individuals, hence evolution takes more steps. Here some domain bias can be exploited, namely, the 0^{th} generation is seeded with 3D scenes that are likely to be encountered, for example a simple standing human, or a human lying on the floor etc.

6 Experiments and Results

First, the principles and the performance of our approach are illustrated with a test case of an arm with a hand. The original infrared image is shown in figure 5. Figure 6 shows the input image after segmentation together with three good matches produced by the EA.

Figure 7 shows an example plot of the fitness of the best individual as a function of time. Note that the fitness decreases fast in the first iterations while it takes more time to reach lower values, which is typical for evolutionary algorithms. Large "jumps" in fitness levels can also be observed, which is mainly due to the multiple-step hill climbing when applied on the position, since it is the parameter that influences the fitness most. Figure 7 also shows that the evolution of the whole population as measured by the average fitness of the population behaves much in the same way.

The above experiment more or less demonstrates the performance of the approach when the model only differs in its pose from the real object in the scene. Now, the posture is also taken into account. In the following example, an image from a situation is used where an elbow joint is visible in addition to the lower arm and the hand. Figure 8 displays the pre-processed input image and again three good matches. Figure 9 shows the best fitness and the average population fitness as a function of the number of iterations for an example run of the EA. In these graphs we again observe the same characteristics as described for the above test case.

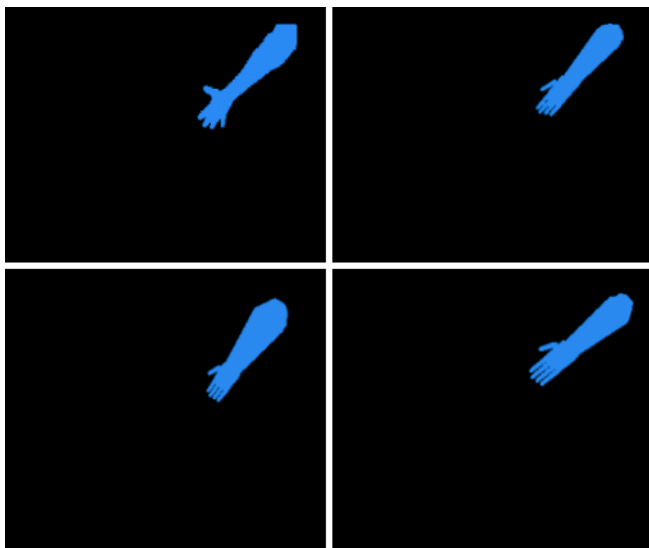


Fig. 6. Three examples of best matches found in an experiment with a lower arm. Top left is the input image after pre-processing, the other three images are different results of the EA. The classification of the according image part as "human" happens very fast as the EA can transform the parameters of the general model very efficiently via hill climbing.

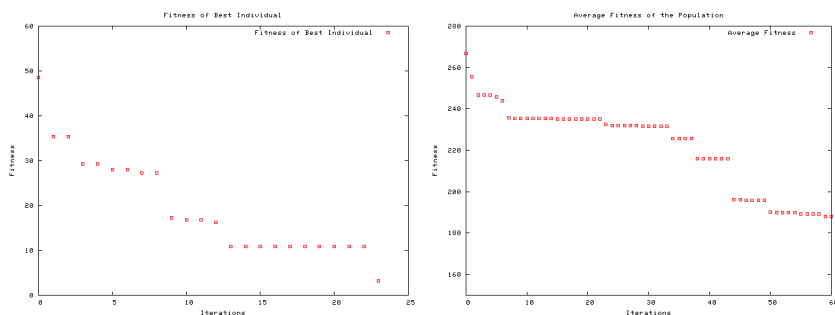


Fig. 7. The development of the fitness of the best individual (left) and the average fitness of the population (right) in an example run of the EA in the test case with a lower arm

The approach does not only work with body parts but as well with whole humans with their many possible posture, i.e., their many DOF. This is indicated by experiments with scenes that contain infrared images of whole humans. In the following two examples are presented. In the first one the human is in a position which is quite standard. In the second one a very complex pose is used, which is nevertheless successfully reproduced and hence the human is recognized.

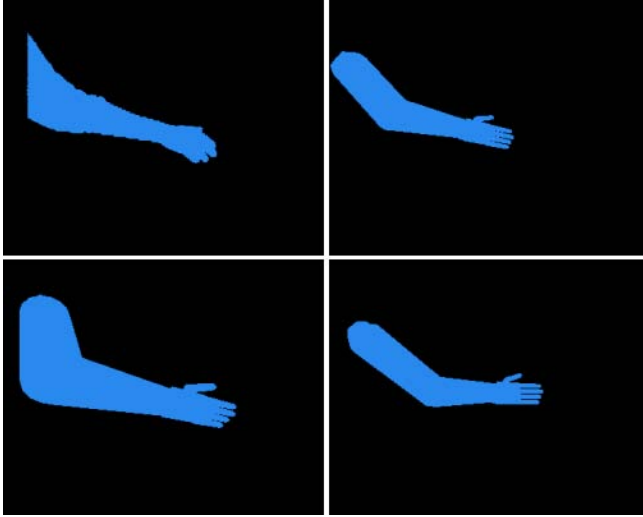


Fig. 8. An experiment where a full arm with an elbow joint and a hand have to be matched. Top left is the input image after pre-processing. The best matches, here again three examples, clearly identify the image by properly representing it via human code snippets.

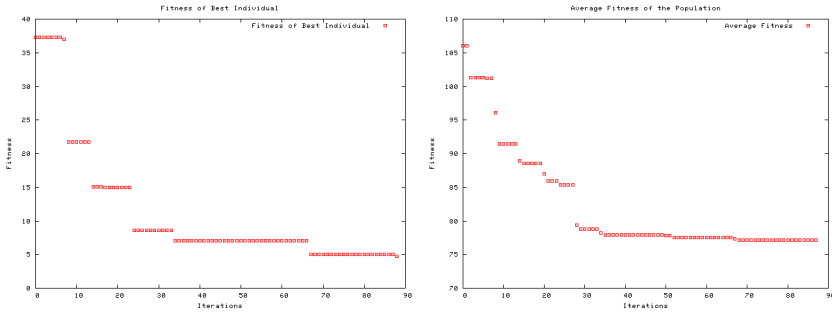


Fig. 9. Fitness of the best individual (left) and average fitness of the whole population (right) when an elbow joint is added

Figure 10 shows the segmented infrared image and three good matches for the first test case. Pretty good matching can be observed, even though in the original image the human back is bended and the model does not support such bending. Figure 11 shows the best fitness, respectively the average population fitness as a function of the number of iterations for this example. Here we also observe the "jumps" in the fitness due to the hill climbing operator. Also, the number of generations it takes to improve the fitness has slightly increased, which can be expected as not only the pose but also the many DOF of the human model have to be adapted.

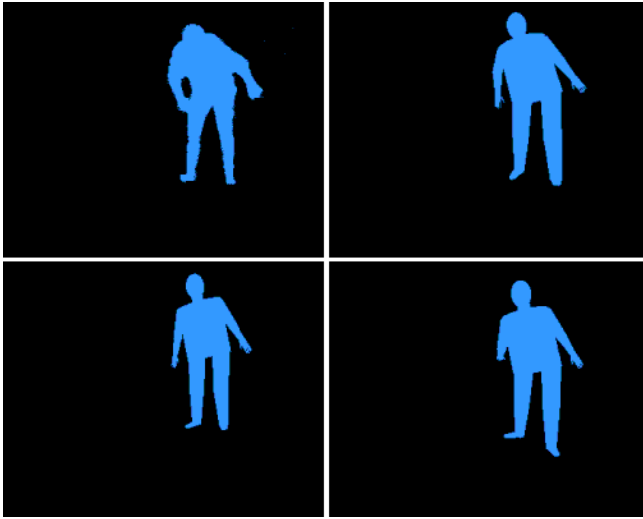


Fig. 10. Also complete humans can be reliably detected as shown here with an input image on the top left and three examples of best matches

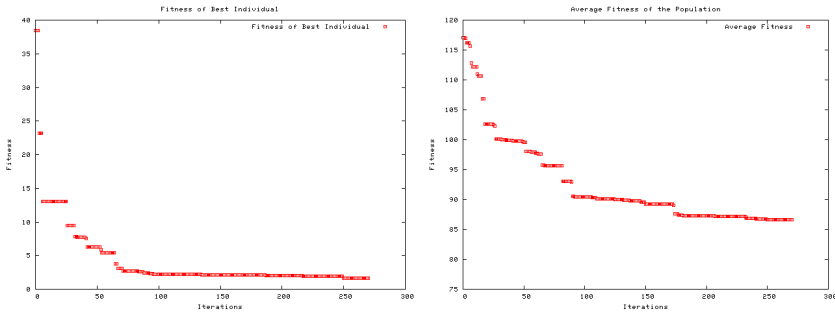


Fig. 11. Fitness of the best individual (left) and average fitness of the whole population (right) when recognizing a human

The last example contains a human in a very complex pose. Figure 12 shows the segmented image with a typical good match. It can be observed that some parts of the human are not matched completely correctly. But for example the torso, the head and the leg are still pretty good. What matters most is that the image is nevertheless represented by a code snippet for a human 3D model. It is hence reliably recognized. The best-fitness and average-fitness graphs look similar to the ones for the other example of a human, especially in respect to runtimes.

The runtimes in general allow an online recognition of humans with the on-board processing capabilities of a RugBot, which features a Pentium-M 1.2GHz processor, i.e., the images can be analyzed while the robot moves along as each

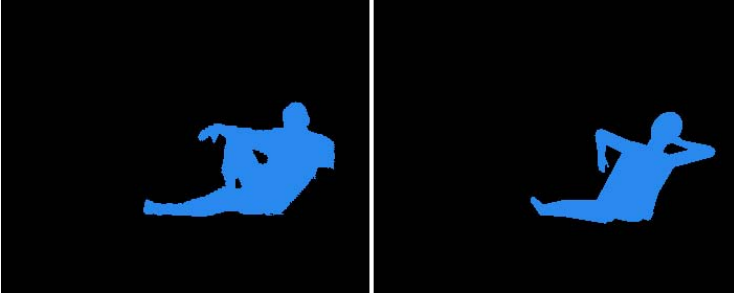


Fig. 12. The segmented infrared image of a real human in a rather difficult posture and an example rendering of an evolved representation. Though there are a few minor discrepancies between the 2D image and the rendering of the 3D model, the scene is clearly recognized to contain a human.

generation takes about 300 msec. Note that a victim test is typically only triggered if there is some "suspicious" data, i.e., a critical temperature range is present in the thermo image. For the general case, there is no exact estimate of the actual time it takes to detect a human or even several ones in a scene. First of all, the performance depends on which body parts can be seen. An arm can be perfectly matched in a few seconds. For the human in the complex posture, which forms a so-to-say worst case here, it took about 1.5 minutes on average to nicely match the 3D model to the 2D image. Second, the recognition is a stochastic process where increased computation time simply increases the confidence that there is indeed a human in the scene. It usually just takes a few generations, i.e., several hundred milliseconds, to transform a code snippet representing a human such that its rendering roughly matches an image of a real human, which is indicated by low fitness values of the best individual. It strongly depends on the application whether this first rough match is considered important enough to trigger other functions or whether the EA should continue first for several seconds to produce a "perfect" fit.

7 Conclusion

In this paper a new approach to detecting humans in images is presented. Instead of processing images in a classical way, 3D representations of the underlying scene are generated, then rendered, and the resulting 2D image is compared to the 2D input snapshot. In doing so, a special image distance function is used to measure the similarity between the input snapshot and the renderings. 3D models of humans and of simple boxes based on OpenGL code are the building blocks for the 3D representations. As shown in experiments with snapshots from the Jacobs rescue arena and real humans, the approach is successful. Given images that contain whole humans or just parts, the EA generates proper representations that clearly indicate by their good fitness the presence of the human in the scene.

Acknowledgments

The authors gratefully acknowledge the financial support of *Deutsche Forschungsgemeinschaft* (DFG).

Please note the name-change of our institution. The Swiss Jacobs Foundation invests 200 Million Euro in **International University Bremen (IUB)** over a five-year period starting from 2007. To date this is the largest donation ever given in Europe by a private foundation to a science institution. In appreciation of the benefactors and to further promote the university's unique profile in higher education and research, the boards of IUB have decided to change the university's name to **Jacobs University Bremen**. Hence the two different names and abbreviations for the same institution may be found in this paper, especially in the references to previously published material.

References

- [Abo98] Abouaf, J.: Trial by fire: teleoperated robot targets chernobyl. *Computer Graphics and Applications* 18(4), 10–14 (1998)
- [AHF⁺05] Aoyama, H., Himoto, A., Fuchiwaki, O., Misaki, D., Sumrall, T.: Micro hopping robot with ir sensor for disaster survivor detection. In: *IEEE International Workshop on Safety, Security and Rescue Robotics, SSRR*, pp. 189–194. IEEE Computer Society Press, Los Alamitos (2005)
- [AT04] Agarwal, A., Triggs, B.: Learning to track 3d human motion from silhouettes. In: *Twenty-first international conference on Machine learning*, ACM Press, New York (2004)
- [BC06] Birk, A., Carpin, S.: Rescue robotics - a crucial milestone on the road to autonomous systems. *Advanced Robotics Journal* 20(5), 595–695 (2006)
- [BINS05] Bahadori, S., Iocchi, L., Nardi, D., Settembre, G.P.: Stereo vision based human body detection from a localized mobile robot. In: *IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 499–504. IEEE Computer Society Press, Los Alamitos (2005)
- [Bir96] Birk, A.: Learning geometric concepts with an evolutionary algorithm. In: *Proc. of The Fifth Annual Conference on Evolutionary Programming*, The MIT Press, Cambridge (1996)
- [BJP94] Birk, A., Paul, W.J.: Schemas and genetic programming. In: *Intern. Conf. on the Integration of Elementary Functions into Complex Behavior* (1994)
- [BMDP06] Birk, A., Markov, S., Delchev, I., Pathak, K.: Autonomous rescue operations on the iub rugbot. In: *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*, IEEE Press, Los Alamitos (2006)
- [BPSC06] Birk, A., Pathak, K., Schwertfeger, S., Chonnaparamutt, W.: The iub rugbot: an intelligent, rugged mobile robot for search and rescue operations. In: *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*, IEEE Press, Los Alamitos (2006)

- [Bur98] Burion, S.: Human detection for robotic urban search and rescue masters thesis. Technical report, Institute de Production Robotique (IPR) (1998)
- [CGPV05] Cucchiara, R., Grana, C., Prati, A., Vezzani, R.: Computer vision system for in-house video surveillance. *Vision, Image and Signal Processing*, IEE Proceedings- 152(2), 242–249 (2005)
- [Dav02] Davids, A.: Urban search and rescue robots: from tragedy to technology. *Intelligent Systems* 17(2), 81–83 (2002)
- [HBG⁺06] Hao, Q., Brady, D.J., Guenther, B.D., Burchett, J., Shankar, M., Feller, S.: Human tracking with wireless distributed pyroelectric sensors. *IEEE Sensors Journal* 6(6), 1683–1696 (2006)
- [Hog83] Hogg, D.: Model-based vision: A program to see a walking person. *Image and Vision Computing* 1(1), 5–20 (1983)
- [JMW⁺03] Jacoff, A., Messina, E., Weiss, B., Tadokoro, S., Nakagawa, Y.: Test arenas and performance metrics for urban search and rescue robots. In: *Proceedings of the Intelligent and Robotic Systems (IROS) Conference* (2003)
- [JWM03] Jacoff, A., Weiss, B., Messina, E.: Evolution of a performance metric for urban search and rescue. In: *Performance Metrics for Intelligent Systems (PERMIS)*, Gaithersburg, MD (2003)
- [KM00] Kakadiaris, L., Metaxas, D.: Model-based estimation of 3d human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 1453–1459 (2000)
- [KT01] Kitano, H., Tadokoro, S.: Robocup rescue. a grand challenge for multiagent and intelligent systems. *AI Magazine* 22(1), 39–52 (2001)
- [MCM01] Murphy, R., Casper, J., Micire, M.: Potential tasks and research issues for mobile robots in robocup rescue. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) *RoboCup 2000*. LNCS (LNAI), vol. 2019, pp. 339–334. Springer, Heidelberg (2001)
- [MPP01] Mohan, A., Papageorgiou, C., Poggio, T.: Example-based object detection in images by components. *Pattern Analysis and Machine Intelligence*, *IEEE Transactions on* 23(4), 349–361 (2001)
- [Mur04] Murphy, R.R.: Trial by fire. *IEEE Robotics and Automation Magazine* 11(3), 50–61 (2004)
- [OPOP97] Oren, M., Papageorgiou, C., Sinhaand, P., Osuna, E., Poggio, T.: Pedestrian detection using wavelet templates. In: *Proc. Computer Vision and Pattern Recognition*, June 1997, pp. 193–199 (1997)
- [PJHK99] Park, S.H., Jung, K., Hea, J.K., Kim, H.J.: Vision-based traffic surveillance system on the internet. In: *Computational Intelligence and Multimedia Applications*. In: *ICCIMA*. Third International Conference on, pp. 201–205 (1999)
- [PP00] Papageorgiou, C., Poggio, T.: A trainable system for object detection. *Int'l J. Computer Vision* 38(1), 15–33 (2000)
- [PS95] Paul, W.J., Solomonoff, R.: Autonomous theory building systems. *Annals of Operations Research* 55(1), 179–193 (1995)
- [PWB⁺05] Paletta, L., Wiesenhofer, S., Brandle, N., Sidla, O., Lypetsky, Y.: Visual surveillance system for monitoring of passenger flows at public transportation junctions. *Intelligent Transportation Systems*, 862–867 (2005)
- [SC04] Shah, B., Choset, H.: Survey on urban search and rescue robots. *Journal of the Robotics Society of Japan (JRSJ)* 22(5), 40–44 (2004)

- [S.F93] Forrest, S.: Genetic algorithms - principles of natural selection applied to computation. *Science* 261, 872–878 (1993)
- [Sin94] Sinha, P.: Object recognition via image invariants: A case study. *Investigative Ophthalmology and Visual Science* 35, 1735–1740 (1994)
- [Sny01] Snyder, R.G.: Robots assist in search and rescue efforts at wtc. *IEEE Robotics and Automation Magazine* 8(4), 26–28 (2001)
- [Spa] Sparacino, F.: In: *Inter-face body boundaries*, issue editor emanuele quinz, anomalie, n.2, paris, france, anomos (2001), <http://citeseer.ist.psu.edu/615750.html>
- [SYDY04] Scholtz, J., Young, J., Drury, J., Yanco, H.: Evaluation of human-robot interaction awareness in search and rescue. In: *Proceedings of the International Conference on Robotics and Automation, ICRA'2004*, pp. 2327–2332. IEEE Press, Los Alamitos (2004)
- [TBB⁺99] Thrun, S., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., Schulz, D.: Minerva: A second-generation museum tour-guide robot. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)* (1999)
- [TGP02] Teo, A.W., Garg, H.K., Puthusserypady, S.: Detection of humans buried in rubble: an electronic nose to detect human body odor. In: *Engineering in Medicine and Biology, 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society, EMBS/BMES*, vol. 3, pp. 1811–1812 (2002)
- [VAJOWC94] Valentine, D., Abdi, H., O'Toole, A.J., Cottrell, G.W.: Connectionist models of face processing: a survey. *Pattern Recognition* 27 (1994)
- [WADP97] Wren, C.R., Azarbayejani, A., Darrell, T., Pentland, A.: Pfunder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(7), 780–785 (1997)
- [YC97] Yow, K., Cipolla, R.: Feature-based human face detection. *Image and Vision Computing* 15(9), 713–735 (1997)
- [YKA02] Yang, M.-H., Kriegman, D.J., Ahuja, N.: Detecting faces in images: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24(1), 34–58 (2002)
- [YSMW98] Yang, J., Stiefelhagen, R., Meier, U., Waibel, A.: Real-time face and facial feature tracking and applications. In: *Proceedings of Auditory-Visual Speech Processing Conference*, pp. 79–84. Terrigal, South Wales, Australia (1998)
- [Yui91] Yuille, A.: Deformable templates for face recognition. *J. Cognitive Neuroscience* 3(1), 59–70 (1991)
- [ZCPR03] Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. *ACM Comput. Surv.* 35(4), 399–458 (2003)

Extent, Extremum, and Curvature: Qualitative Numeric Features for Efficient Shape Retrieval

B. Gottfried, A. Schuldt, and O. Herzog

Centre for Computing Technologies (TZI)
University of Bremen, Am Fallturm 1, D-28359 Bremen

Abstract. In content-based image retrieval we are faced with continuously growing image databases that require efficient and effective search strategies. In this context, shapes play a particularly important role, especially as soon as not only the overall appearance of images is of interest, but if actually their content is to be analysed, or even to be recognised. In this paper we argue in favour of numeric features which characterise shapes by single numeric values. Therewith, they allow compact representations and efficient comparison algorithms. That is, pairs of shapes can be compared with constant time complexity. We introduce three numeric features which are based on a qualitative relational system. The evaluation with an established benchmark data set shows that the new features keep up with other features pertaining to the same complexity class. Furthermore, the new features are well-suited in order to supplement existent methods.

1 Introduction

Content-based retrieval from large image databases is a challenging problem in computer vision. Its importance grows continuously with the increasing penetration of image databases in many areas of everyday life. As an example, think of Flickr¹, which is an internet platform for uploading and sharing of photographic content. Large amounts of image data can also be found in the economic as well as the scientific area. The pure amount of content, and even more its fast growth, illustrates the demand for efficient and effective search strategies. Therefore, it is particularly desirable to choose features with only little computational complexity for the comparison of images. In image retrieval, this has already been applied for a long time to colour and texture. As an example, think of colour histograms having a fixed number of entries. Consequently, they can be compared with constant time complexity.

For the comparison of objects by their shape there exist also meaningful methods [9]. However, their efficiency in terms of computational complexity is still a problem. As an example, the approach of [8] which achieves promising retrieval results has a biquadratic time complexity, $O(n^4)$. This is different for numeric

¹ <http://www.flickr.com/>

shape features [3,4]. They characterise shapes by a single numeric value. This entails two advantages: First, two shapes described by such a feature can be compared with constant time complexity, $O(1)$. Second, the shapes of an image database can be ordered in accordance to a numeric shape feature. This allows retrieval algorithms to be applied which employ binary search strategies with a time complexity of $O(\log n)$, with n being the number of images in the database.

Applied exclusively, however, the retrieval performance for each of these features is rather limited since each one describes only one simple property of an object, e.g. the aspect ratio [3] of the minimal enclosing rectangle. But combining such simple features improves classification results significantly, still with constant time complexity for the comparison of two shapes. Characterising an object twice by similar features, however, does most likely not improve its description. By contrast, it is more promising to combine features which are built upon different foundations. We introduce three new numeric shape features based on a qualitative approach, thereby complementing existing features. Afterwards, we combine them with existing quantitative numeric shape features.

The remainder of this paper is structured as follows: In Sect. 2 we introduce previous work that underlies our new approach which is then presented in Sect. 3. We evaluate our method in Sect. 4 by comparing it to other approaches. Eventually, a conclusion follows in Sect. 5.

2 Previous Work

The work presented in this paper focuses on the characterisation of shapes. For this purpose, it is assumed that silhouettes have been segmented from raster images before. Contours of silhouettes can then be represented by polygons, as it has been motivated from the cognitive point of view by [2]. Additionally, the following reasons support a pure cognitive motivation: First of all, confining oneself to contour points the uniform distribution of points of the silhouette's interior can be excluded. Since only the contour points are relevant concerning any object's outer shape, this restriction does not entail any loss of relevant information. Secondly, the application of polygonal approximation algorithms [11] allows a massive data reduction with only little influence on the perception of shape. We apply especially the method of [10], thereby choosing a scale-invariant approximation error of one percent of a polygon's perimeter.

2.1 Reference System

The polygon obtained in the previous step of abstraction forms a quantitative description of the underlying shape. The concrete representation depends on scale, translation, and rotation of the object under consideration. Furthermore, it is imprecise due to noise in the underlying image data. The aim is therefore to achieve an invariance against scale, translation, and rotation as well as a certain robustness against noise.

In order to meet the above objectives we apply the orientation grid of [15] which brings in a qualitative abstraction. It is induced by each of the polygon's

line segments as depicted in Fig. 1 (after an orientation has been imposed on the polygon) and it consists of three auxiliary lines. The first one runs through the reference segment allowing the qualitative distinction whether a point is located on its left or right hand side. The two other lines are oriented orthogonally to the first one, whereby each of them passes either the reference segment’s start point or its end point. Their arrangement enables the decision whether a point lies in front of the reference segment, next to, or behind it. In general, the orientation grid divides the two-dimensional plane into six sectors, as depicted on the left hand side of Fig. 1. Instead of its quantitative coordinates it is then possible to characterise a point by its position relative to the respective line segment. This is the third sector in the example depicted in the centre of Fig. 1. This description is invariant against scale, translation, and rotation since the orientation grid is an intrinsic reference system of the polygon, i.e. it is induced on each of its line segments. A certain robustness against noise is achieved by partitioning the two-dimensional plane into sectors. Generally, changing a point’s quantitative position does not change the sector it is located in; even larger movements of points only result in neighbouring sectors.

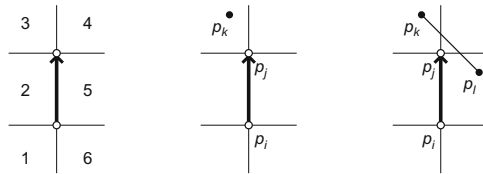


Fig. 1. Left: The orientation grid divides the two-dimensional plane into six sectors. Centre: The qualitative position of p_k is in sector 3, which is located front left w. r. t. the reference line $\overline{p_i p_j}$. Right: The line segment $\overline{p_k p_i}$ passes the sectors 3, 4, and 5.

2.2 Bipartite Arrangements

Apart from characterising single points it is also possible to apply the orientation grid in order to relate two polygonal line segments to each other. This is achieved by the qualitative concept of bipartite arrangements [5, 6], in short \mathcal{BA} . The extension from characterising single points to line segments is straightforward. As each line segment is defined by a start and an end point, these points have to be taken into consideration. Both of them can be located in any of the six sectors of the orientation grid (Fig. 1 right). Hence, this theoretically leads to a number of $6^2 = 36$ conceivable arrangements between two line segments. By omitting symmetries and intersections [5] it is possible to reduce this number to those 23 \mathcal{BA}_{23} relations that are depicted on the left hand side of Fig. 2. Their mnemonic labels are given in the centre of the same figure. As this approach relates line segments in the two-dimensional plane it can be categorised as an extension of Allen’s 13 qualitative relations between one-dimensional intervals [1].

A bipartite arrangement relation describes the position of a line segment w. r. t. a reference segment. A polygon’s whole course can then be characterised

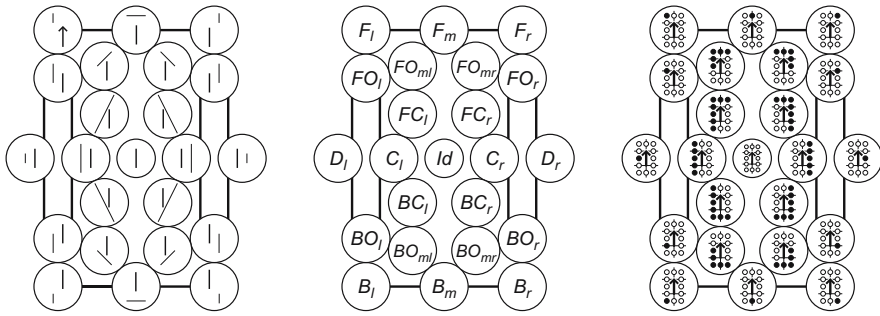


Fig. 2. Example configurations (left) and mnemonic labels (centre) for the 23 \mathcal{BA}_{23} relations between two line segments in the two-dimensional plane. Right: The iconic representation of the bipartite arrangement’s scopes.

by applying a sequence of \mathcal{BA}_{23} relations, describing each of the n polygonal line segments, one after another [6]:

Definition 1 (Course). Let x be a line segment of a simple, closed polygon. Its course, in short $C(x)$, contains the \mathcal{BA}_{23} relations of all segments y_i w. r. t. x :

$$C(x) := (x_{y_0}, \dots, Id, \dots, x_{y_{n-1}}), x_{y_i} \in \mathcal{BA}_{23}; i = 0, \dots, n - 1$$

Hence, we obtain a qualitative description of the considered polygon w. r. t. one of its segments. In order to arrive at a complete description it is necessary to apply not only one line segment as a reference, but all of them, one after another. This results in the following definition:

Definition 2 (Polygonal Course). Let P be a simple, closed polygon. Its polygonal course, in short $C(P)$, is the conjunction of all courses of P :

$$C(P) := \bigwedge_{i=0}^{n-1} C(x_i)$$

The result is a matrix that comprises all n^2 \mathcal{BA}_{23} relations that exist between the polygon’s n line segments.

2.3 Scopes of Bipartite Arrangements and Courses

Based on the work of [5,6] a more general approach has been introduced by [12,13]. Their idea is to represent \mathcal{BA}_{23} relations and even courses as sets of atomic relations. The advantage of such a representation is that it allows to apply standard set operations, e.g. union and intersection. A \mathcal{BA} is considered atomic if it populates only one of the orientation grid’s sectors, which holds for B_l, D_l, F_l, F_r, D_r , as well as B_r (Fig. 2 left). Furthermore, those relations connecting adjacent sectors, namely $BO_l, FO_l, F_m, FO_r, BO_r$, and B_m , are also atomic. Altogether, these twelve relations form $\mathcal{BA}_{12} \subset \mathcal{BA}_{23}$.

Each \mathcal{BA}_{23} relation can then be represented by its *scope*, i. e. the set of atomic relations it consists of. The right hand side of Fig. 2 visualises the \mathcal{BA}_{23} relations' scopes. Each of the twelve circles stands for the atomic relation that is located at its position in the orientation grid. An opaque circle thereby means that the atomic relation is part of a scope, while a transparent one indicates its absence. This results in the following definition:

Definition 3 (Scope of a \mathcal{BA}). *Let x and y be line segments of a simple, closed polygon. The set of atomic \mathcal{BA}_{12} relations that represents the relation $x_y \in \mathcal{BA}_{23}$ is called the relation's scope, in short $\sigma(x_y)$:*

$$\sigma(x_y) := \{x_{y_1}, \dots, x_{y_n}\}, x_{y_i} \in \mathcal{BA}_{12}$$

Each scope is a description with constant space complexity as the total number of atomic relations that may be contained in a scope is limited to $|\mathcal{BA}_{12}| = 12$. While a single \mathcal{BA}_{23} relation is limited to characterising the relationship between two line segments, this limitation does not hold for scopes. By contrast, it is also possible to characterise the position of a whole course by a single scope relation. This can be achieved by exploiting the scope's set property. In particular, we create the union of the scopes of all \mathcal{BA}_{23} relations participating in a course:

Definition 4 (Scope of a Course). *Let x be a line segment of a simple, closed polygon and $C(x)$ its course. The set of atomic relations describing the position of $C(x)$ is called the scope of the course, in short $\sigma(C(x))$:*

$$\sigma(C(x)) := \bigcup_{i=0}^{n-1} \sigma(r_i), r_i \in \mathcal{BA}_{23}$$

Proceeding this way, however, leads to a certain loss of information since Gestalt features are not further considered within the scope of a given course. Nevertheless, the resulting characterisation offers still the expressiveness for applying concepts such as the scope histogram [13] which computes the frequencies of a polygon's scopes, leading to promising results as has been shown in [12].

3 Qualitative Numeric Shape Features

The scope histogram [13] forms a very compact representation as it characterises the shape of an object with constant space complexity. It is a statistical shape descriptor as it considers only the frequencies with which the courses' scopes occur. The notion of scope, however, is not limited to statistics. By contrast, it is also possible to derive further compact shape features from a polygonal course (Definition 2). In this section we particularly introduce three of them, namely the numeric shape features *extent*, *extremum*, and *curvature*, which are defined on the basis of the scope approach.

3.1 Extent

The notion of a course (Definition 1) can be applied in order to characterise a polygon qualitatively w.r.t. its line segment x . Definition 4 introduces the so-called scope of a course as a more compact representation. By applying a set of atomic \mathcal{BA}_{12} relations the scope $\sigma(C(x))$ specifies which of the orientation grid's sectors are populated by the given course. In general, the complexity of a shape increases with the number of orientation grid sectors that are passed by its course. This observation leads to the definition of the extent, which is a simple measure of a shape's complexity. The number of populated sectors correlates with the atomic relations within the respective scope. Thus, it is sufficient to count the atomic relations a scope comprises (Fig. 3). The maximum range η_{\max} of an extent η is defined by the maximally possible number of atomic relations within a scope, which is $\eta_{\max} = |\mathcal{BA}_{12}| = 12$. This results in the following definition:

Definition 5 (Extent). *Let x be a line segment of a simple, closed polygon and $\sigma(C(x))$ the scope of its course. The number of the atomic \mathcal{BA}_{12} relations of the scope is called the extent of the scope, in short $\eta(\sigma(C(x)))$, and it holds that*

$$\eta(\sigma(C(x))) := |\sigma(C(x))| \in \{1, 2, \dots, \eta_{\max}\}$$

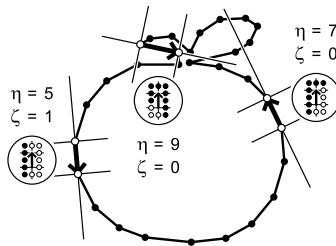


Fig. 3. An apple. The three highlighted reference segments demonstrate the computation of extent η and extremum ζ from the respective line segments' scopes.

Definition 5 determines the extent of a single course. This extent characterises the complexity of a polygon as perceived by the line segment on which the orientation grid is currently induced. However, as mentioned before each polygon is characterised by n courses, where n is the total number of line segments. We obtain a single numeric value, that characterises the whole polygon, by computing the average extent for all courses. In order to arrive at a normalised value in $[0, 1]$, the polygonal extent is divided by η_{\max} :

Definition 6 (Polygonal Extent). *Let P be a simple, closed polygon. Its polygonal extent, in short $\eta(P)$, is the average number of the atomic \mathcal{BA}_{12} relations of the scopes of all courses $C(x_i)$ of P :*

$$\eta(P) := \frac{1}{n \eta_{\max}} \sum_{i=0}^{n-1} \eta(C(x_i))$$

Figure 4 depicts example silhouettes from the database of [9] that illustrate the range of possible extents η . The spring at the top left position exhibits the highest extent. This is due to the line segments that are located within the spring’s ends. Their extent is maximal, i. e. $\eta_{max} = 12$, since the course of these segments runs completely around them. The extent of the depicted objects decreases from the top left to the bottom right. Out of all objects the triangle has the lowest extent: for each of its line segments the polygon is solely located in the second sector of the orientation grid, i. e. $\eta = 1$. These examples demonstrate that the extent of a shape can easily be comprehended. Generally spoken, the extent is the higher the more complex the underlying shape is, in terms of indentations and how complex they are shaped.

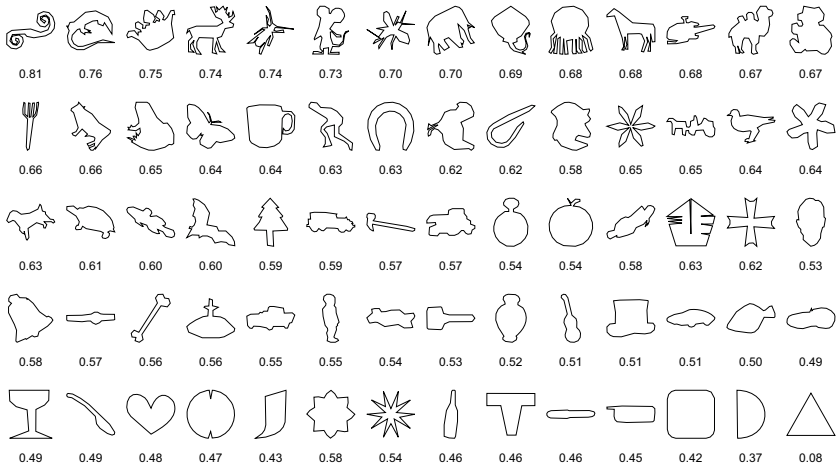


Fig. 4. Example shapes ordered accordingly to their extent η

3.2 Extremum

The second qualitative shape feature that can be derived from the scope is called the extremum. It tells us for a given line segment whether it is an extremum of its respective polygon. Thereby, we denote a line segment as extreme if it is part of the polygon’s convex hull. This in turn is the case whenever no other part of the polygon is located on the right hand side of the considered line segment. Detecting such a configuration on the basis of the scope representation is fairly straightforward: in this case the whole polygon is l , i. e. it is located left w. r. t. the reference segment. One possibility to realise l is the scope $\sigma(C_l)$ of the \mathcal{BA}_{23} relation C_l (Fig. 2).

Proposition 1 (Extremum). *Let x be a line segment of a simple, closed polygon. x is said to be an extremum, in short $\zeta(C(x))$, if*

$$\zeta(C(x)) = \begin{cases} 1 & \text{iff } \eta(\sigma(C(x)) \cup l) = \eta(l) \\ 0 & \text{else} \end{cases}$$

Proof: A union between l and a scope $\sigma(C(x))$ of a course $C(x)$, that leads to an increase in the scope's extent, means that further atomic relations have been added by means of the union operation. As the scope l already contains all atomic relations on the reference segment's left hand side, the additional atomic relations must lie on its right hand side. \square

From the three example line segments highlighted in Fig. 3 only the leftmost one is extreme. It is the only one that comprises atomic relations solely on its left side. The other scopes' atomic relations populate both halves of their respective orientation grids. In order to obtain a characterisation of the whole polygon, we count the extreme segments and relate them to the number of line segments contained in the polygon:

Definition 7 (Polygonal Extremum). *Let P be a simple, closed polygon. Its polygonal extremum, in short $\zeta(P)$, measures, how many segments of P are extremes:*

$$\zeta(P) := \frac{1}{n} \sum_{i=0}^{n-1} \zeta(C(x_i))$$

Figure 5 shows example silhouettes in conjunction with their respective extremum values. Since they are completely convex the first three objects (a square, a triangle, and a semi circle) have the highest extremum values. The number and size of concavities increase from the top left to the bottom right. None of the line segments of the last three devices is convex. Consequently, the extremum values of these shapes are zero. Therewith, the ordering established by the extremum corresponds to the visual perception of the considered shapes' convexity.

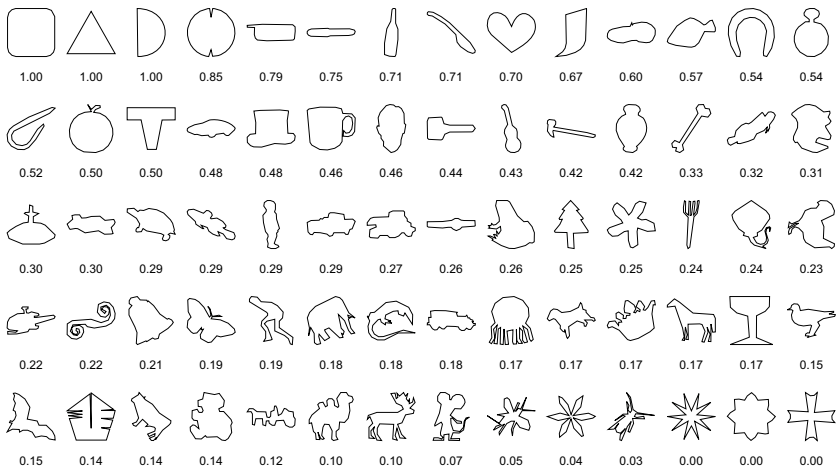


Fig. 5. Example shapes ordered accordingly to their extremum ζ

3.3 Curvature

The third feature we shall introduce here is referred to as the curvature. It describes how often the course $C(x)$ changes its position as perceived from its reference segment x . More specifically, it is examined how often the relations within the course change from one atomic \mathcal{BA}_{12} relation to another one. For this purpose, it is not sufficient to analyse the scope $\sigma(C(x))$ of the course $C(x)$ as a whole. It is rather necessary to analyse the scope of every single relation in $C(x)$ in order to relate it to its successor:

Definition 8 (Curvature). *Let x be a line segment of a simple, closed polygon and $C(x)$ its course. The curvature of $C(x)$, in short $\xi(C(x))$, arises from the sequence of its relations r_i as follows:*

$$\xi(C(x)) := \sum_{i=0}^{n-1} \begin{cases} 0 & \text{if } r_i = Id \wedge (str(r_i) \wedge int(r_{i-1}, r_{i+1})) \\ 1 & \text{if } r_i = Id \wedge (str(r_i) \vee int(r_{i-1}, r_{i+1})) \\ 2 & \text{if } r_i = Id \\ \eta(\sigma(r_i)) - 1 & \text{if } int(r_i, r_{i+1}) \\ \eta(\sigma(r_i)) & \text{else} \end{cases}$$

The first three cases in Definition 8 occur when the currently considered segment is the reference segment itself. Before discussing these special cases, we shall start with an analysis of the more general cases, namely the fourth and the fifth one.

The right hand side of Fig. 2 depicts the distinguishable scopes for single line segments. Some of these scopes contain more than one atomic relation, which means that a change of position w.r.t. the reference segment already occurs within the respective line segment. The total number of changes in position is thereby defined by the extent of the scope of one line segment’s relation: $\eta(\sigma(r_i)) - 1$. Another change in position may occur between two subsequent line segments r_i and r_{i+1} . However, this is only the case, if their scopes do not intersect, i.e. they have no atomic relation in common. In order to determine such an intersection, we apply an auxiliary function. It creates the intersection of the scopes under consideration. The intersection is not empty if the result’s extent is greater than zero:

$$int(r_1, r_2) := \begin{cases} \text{true} & \text{iff } \eta(\sigma(r_1) \cap \sigma(r_2)) > 0 \\ \text{false} & \text{else} \end{cases} \tag{1}$$

We shall now address the special case in which the considered line segment is the course’s reference segment itself. In this case it does not suffice to examine the intersection in the scopes of the predecessor and the successor. It is additionally necessary to find out whether or not the course has a kink around this segment. Therefore, the reference segment’s predecessor and successor have to be compared. If the scope σ of one relation is the inverse σ^{-1} of the other one, the course has no kink. This can be determined using the following auxiliary function:

$$str(r_i) := \begin{cases} \text{true} & \text{iff } \sigma(r_{i-1}) = \sigma^{-1}(r_{i+1}) \\ \text{false} & \text{else} \end{cases} \tag{2}$$

Figure 6 gives an example on the application of the two auxiliary functions with which we are now able to determine the curvature for a single course. However, in order to compute the curvature for a whole polygon, we have to extend our definition again. Therefore, the average curvature for all of the polygon’s n courses is determined. The range of the curvature for a single course is thereby $[1, \infty[$. In order to arrive at a value in $]0, 1]$ like for the other features, we compute each curvature’s multiplicative inverse:

Definition 9 (Polygonal Curvature). *Let P be a simple, closed polygon. Its polygonal curvature, in short $\xi(P)$, is defined as the average of the multiplicative inverses of the curvatures of all courses $C(x_i)$ of P :*

$$\xi(P) := 1 - \frac{1}{n} \sum_{i=0}^{n-1} \frac{1}{\xi(C(x_i))}$$

The curvature values of example silhouettes are given in Fig. 7. The object that is most curved is the spring on the left hand side. The curvature decreases from the top left to the bottom right. The triangle exhibits the lowest curvature. That is, as in the case of the extremum and extent, also the curvature corresponds

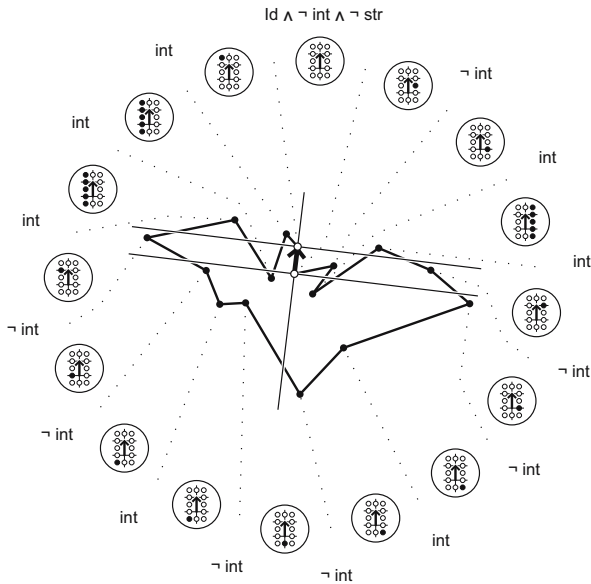


Fig. 6. A bat. All line segments’ scopes w. r. t. the highlighted reference segment. The result of the auxiliary functions is denoted for each pair of consecutive line segments. The curvature ξ of this example course is 21.

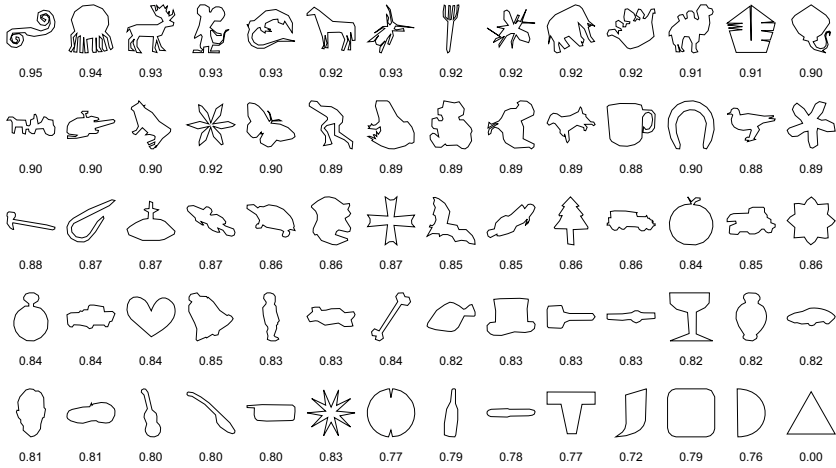


Fig. 7. Example shapes ordered accordingly to their curvature ξ

with the visual perception of the silhouettes: the more curved an object is the higher its qualitative curvature value is.

3.4 Comparison

Figures 4, 5, and 7 illustrate, that convex shapes (high extremum values) coincide with low values for extent and curvature. Thus, there seems to be a correlation of extremum with both extent and curvature. The correlation observed between extremum and extent can be explained by the fact that convex shapes are only located left w. r. t. their polygon which also restricts the range of their extent. Furthermore, convex shapes are bent only into one direction. Since they comprise no reversals also the range for their curvature is limited. These correlations indicate why a combination of these three features will most likely not be as effective as a combination of completely independent features.

However, our new features are by no means completely redundant. On the contrary, there exist also situations in which our new features supplement each other. One counter-example against the correlation is formed by the pencil and the triangle on the left hand side of Fig. 8. While both of them are completely convex (i. e. have an extremum of 1.0) they can still be distinguished by their extent and curvature. The right hand side of Fig. 8 depicts two silhouettes which exhibit the same curvature, but nevertheless differ in their extent and extremum values. This is due to the fact that the right polygon comprises some convex line segments (those lying on the convex hull, i. e. being extreme according to Definition 1) while the left one has none of them (relating to the extremum values). Furthermore, the right hand side polygon is much more folded than the left hand polygon (relating to its higher extent).

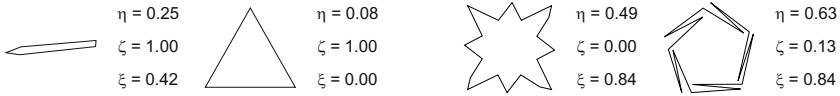


Fig. 8. Left: Two convex shapes with same extremum ζ that can nevertheless be distinguished by their extent η and curvature ξ . Right: Two shapes with same curvature values ξ , but different extent η and extremum ζ .

4 Retrieval Performance

In order to assess the retrieval performance of our approach we shall now conduct an evaluation. As introduced above our method focuses on the shape of objects. Hence, we apply particularly the popular core experiment CE-Shape-1 [9] for the MPEG-7 standard. The purpose of this experiment is to compare different shape descriptors. It takes only retrieval results into account, thereby completely abstracting from the underlying algorithms. Thus, it allows our shape features to be measured in comparison with others that have already been examined with this standardised reference test.

4.1 Experiment

We especially focus on Part B of the reference test which addresses similarity-based shape retrieval. The experiment comes along with a database of 1400 silhouette images. These images are grouped together into 70 classes, whereby each class comprises 20 instances. Figures 4, 5, and 7 depict example instances of all 70 classes. During the test each image serves as a query, one after another. All others are ordered concerning their similarity according to the approach under consideration. The test's result is determined accordingly to the following definition: For each query, the correct matches among the first 40 results are counted. This number is then related to the maximally possible number of correct results. This is 20 for each single query (since each class comprises 20 instances) and 28000 for all 1400 queries. Thus, a result of 100% means that all expected results are found. Nevertheless, such an outcome is most unlikely if only shape knowledge is applied [9]. This is due to the fact that the 70 classes are grouped by semantic aspects, which means that some of them exhibit a broad bandwidth of different shapes. Conversely, using a hypergeometric distribution, it is easy to show that a random ordering of the search results achieves about 2.86% in the MPEG test. This is a lower bound showing how much better an approach is in comparison with mere chance.

4.2 Existing Approaches

Our new shape features are confined to a single numeric value. This allows a comparison with constant computational complexity. Hence, it is a good choice to compare them to approaches exhibiting the same complexity. We consider

three quantitative numeric shape features. These are the compactness [3], which is the ratio $\frac{4\pi A}{P^2}$ of a polygon’s area and perimeter, and the radius ratio [4] $\frac{R_{min}}{R_{max}}$ of the minimum enclosing circle and the maximal contained circle. Furthermore, we apply the aspect ratio [3] $\frac{H_r}{W_r}$ of the minimal enclosing rectangle. These three features have in common that they are based on fundamental geometric properties and that they represent a shape by just one single number.

Apart from the above numeric features, we compare our method also to two other approaches which also pertain to the same class of complexity. On the one hand the seven invariant Hu moments [7], which can directly be applied to polygons [14], on the other hand the scope histogram of [12], which is based on the scope of polygons like our method. In contrast to our approach which determines visual shape properties, the scope histogram simply computes how often the 86 distinguishable scopes occur in a polygon.

4.3 Retrieval Results

The classification results of our new numeric shape features compared to the previous ones can be found in Table 1. The results show that all numeric features separately achieve results between about 16% and 25%. Thereby, our new qualitative features slightly outperform the quantitative ones. All considered numeric features clearly exceed the 3% of a random ordering five to eight times. This is notable since each feature consists of only one single numeric value. The Hu moments and the scope histogram achieve better results of about 34% and 46% respectively. This, however, is not surprising as they comprise a more complex range of distinctions, namely seven and 86 respectively.

Table 1. Retrieval results of the numeric shape features extent (ET), extremum (EM), curvature (CU), compactness (CO), radius ratio (RR), and aspect ratio (AR) examined with CE-Shape-1 Part B. Furthermore, also the Hu moments (HU) as well the scope histogram (SH) are evaluated.

| ET | EM | CU | CO | RR | AR | HU | SH |
|--------------|--------------|--------------|-------|-------|-------|-------|-------|
| 24.97 | 18.19 | 23.30 | 21.86 | 16.82 | 24.12 | 34.13 | 45.52 |

Owing to their low computational complexity, it is possible and reasonable to combine multiple numeric features. This, however, only makes sense if the results are improved by such combinations. The classification results of these combinations are summarised in Table 2. Our new qualitative numeric features achieve a retrieval result of about 34%. This is already remarkable as they slightly outperform the seven Hu moments. Nevertheless, their result lies below 52% achieved by the three quantitative numeric features. This can be explained by the fact that all qualitative features are based on the scope (see Sect. 3.4), while the quantitative ones base on different geometric properties. Together, all six numeric features achieve a retrieval result of about 62%. This is especially

remarkable as we apply only six numeric values for the characterisation of the objects' shapes, i. e. we only need constant time for the comparison of two shapes. Especially, the result of the quantitative numeric features combined with the Hu moments lies about eight percentage points below their combination with our new features. The quantitative numeric features and the scope histogram slightly outperform the combination of the six numeric features. However, the scope histogram consists of 86 numeric values while we apply only six of them.

Table 2. Combining multiple numeric features improves the retrieval results: the qualitative features (QL), the quantitative features (QN), as well as the combination of all numeric features (AN). For comparison, the quantitative features have also been combined with the Hu moments (NH) and the scope histogram (NS).

| QL | QN | AN | NH | NS |
|-------|-------|--------------|-------|-------|
| 34.33 | 51.58 | 61.51 | 53.99 | 63.75 |

From the low computational complexity of $O(1)$ results a fast execution. On a computer with Windows XP and an Intel Centrino Duo processor with 2.16 GHz it takes only about five seconds in order to conduct the whole MPEG test (nearly two million comparisons) for the six numeric shape features. Eventually, it is worth mentioning that a classification result of 62% is only about 15 percentage points less than the approach of [8] who achieve 76.45%. However, their computational complexity is biquadratic while ours is still constant.

5 Conclusion

In this paper we introduce extent, extremum, and curvature, three new numeric shape features. While other numeric shape features directly base on geometric properties, we apply a polygonal approximation as well as a qualitative abstraction before. The orientation grid as an underlying intrinsic reference system brings in an invariance against scale, translation, and rotation. Furthermore, the polygonal approximation and the coarse perspective of the qualitative representation realise a certain robustness against noise in the underlying image data. The new features are easily comprehensible as they are defined on the qualitative relational system of the scope approach.

The evaluation results show that our new numeric features can keep up with comparable methods. The retrieval performance can be improved by combining multiple numeric features. Together with the three quantitative numeric features discussed in this paper a retrieval result of about 62% in the MPEG test is achieved. Hence, the new features in fact supplement the other established features. Other features pertaining to the same complexity class can thus be outperformed. The retrieval result is remarkable as only six numeric values are applied for the characterisation of each shape. The computational

complexity for the comparison of two of these shapes is therefore constant. The achieved result is only about 15 percentage points less than the 76.45% of [8]. However, their computational complexity is biquadratic while ours is constant.

References

1. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11), 832–843 (1983)
2. Attneave, F.: Some Informational Aspects of Visual Perception. *Psychological Review* 61, 183–193 (1954)
3. Duda, R.O., Hart, P.E.: *Pattern Classification and Scene Analysis*. John Wiley & Sons, Chichester (1973)
4. Garson, G.D., Biggs, R.S.: *Analytic Mapping and Geographic Databases*. Sage Publications (1992)
5. Gottfried, B.: Reasoning about Intervals in Two Dimensions. In: *IEEE Int. Conf. on Systems, Man and Cybernetics, The Hague, The Netherlands*, pp. 5324–5332. IEEE Computer Society Press, Los Alamitos (2004)
6. Gottfried, B.: *Shape from Positional-Contrast — Characterising Sketches with Qualitative Line Arrangements*. Deutscher Universitäts-Verlag (2007)
7. Hu, M.-K.: Visual Pattern Recognition by Moment Invariants. *IRE Transactions on Information Theory* 8(2), 179–187 (1962)
8. Latecki, L.J., Lakämper, R.: Shape Similarity Measure Based on Correspondence of Visual Parts. *IEEE PAMI* 22(10), 1185–1190 (2000)
9. Latecki, L.J., Lakämper, R., Eckhardt, U.: Shape Descriptors for Non-rigid Shapes with a Single Closed Contour. In: *IEEE CVPR, Hilton Head Island, SC, USA*, pp. 424–429. IEEE Computer Society Press, Los Alamitos (2000)
10. Mitziias, D.A., Mertzios, B.G.: Shape Recognition with a Neural Classifier Based on a Fast Polygon Approximation Technique. *Pattern Recognition* 27, 627–636 (1994)
11. Rosin, P.L.: Assessing the Behaviour of Polygonal Approximation Algorithms. *Pattern Recognition* 36, 505–518 (2003)
12. Schuldt, A., Gottfried, B., Herzog, O.: Retrieving Shapes Efficiently by a Qualitative Shape Descriptor: The Scope Histogram. In: Sundaram, H., Naphade, M., Smith, J.R., Rui, Y. (eds.) *CIVR 2006. LNCS*, vol. 4071, pp. 261–270. Springer, Heidelberg (2006)
13. Schuldt, A., Gottfried, B., Herzog, O.: Towards the Visualisation of Shape Features: The Scope Histogram. In: Freksa, C., Kohlhase, M., Schill, K. (eds.) *KI 2006. LNCS (LNAI)*, vol. 4314, pp. 289–301. Springer, Heidelberg (2007)
14. Steger, C.: On the Calculation of Arbitrary Moments of Polygons. Technical Report FGBV-96-05, Informatik IX, Technische Universität München (1996)
15. Zimmermann, K., Freksa, C.: Qualitative Spatial Reasoning Using Orientation, Distance, and Path Knowledge. *Applied Intelligence* 6, 49–58 (1996)

Extraction of Partially Occluded Elliptical Objects by Modified Randomized Hough Transform

Kwangsoo Hahn, Youngjoon Han, and Hernsoo Hahn

Dept. of Electrical Engineering, Soongsil University
Sando-Dong, Dongjak-Ku, Seoul 156-743, Korea
{kshahn, young, hahn}@ssu.ac.kr
<http://visionlab.ssu.ac.kr>

Abstract. Ellipse detection is very important in computer vision, object recognition, feature selection and so on. This paper proposes a new ellipse detection method using local information of edge points. It merges line segments using modified randomized Hough transform (RHT) that belongs to same ellipse. It is fast, correct and robust to noise because it detects ellipse using line segments that are constructed by local information of edge points. The proposed method in this paper can not apply only ellipse detection but line, circle and partially occluded elliptical object.

1 Introduction

Detection of elliptical shapes is necessary for collecting peculiar features of objects and for finding various circular objects. Not only in most industrial parts such as bolts, rings and tires etc., they are included but also in natural object such as a human face and erythrocyte. Because these circular objects are transformed to elliptical shapes when they are projected to 2D image sensors, detection of elliptical shapes is a very important task in computer vision.

Many approaches researches have been proposed for detection of ellipses. Beginning from the Hough transform (HT) proposed at 1962 by P.V.C. Hough [1], a number of HT-based algorithms have been developed [2-6,10]. One of them is the randomized Hough transform (RHT) proposed by Lei Xu [5]. It represents an ellipse using a second order polynomial equation which has three parameters so that they can be derived if the coordinates of three edge pixels are given. That is, it reduces the computational complexity by dividing a 5D space, required by the traditional HT, by a 2D space for the epicenter of ellipse and a 3D space for the rest of the ellipse parameters. It also selects edge pixels in random, not by probabilistic model, to reduce the computational time. However, a random selection of three edge pixels causes a detection of false ellipses when objects are overlapped. Connective randomized Hough transform (CRHT)[14,15] proposed by H. Kalviainen finds the line parameters using a connective component that was made of local information within a window whose size is arbitrarily selected and whose center is located at an edge point. Because this method uses local connective component in sub-image, it is faster than RHT for line detection that uses every pixels in the image. However, it is too simple

to be applied to detect an ellipse. Another method proposed by Xie [7] takes the advantages of major axis of an ellipse for finding the ellipse parameters fast and efficiently. It selects a pair of pixels and assumes that they are two vertices on the major axis of an ellipse. Then it can calculate the ellipse parameters such as the center point, the half-length of major axis and the orientation of ellipse. Then, a third pixel is used to determine the half-length of minor axis by ellipse geometry by voting to one dimensional accumulator array. It has an advantage that it does not require the evaluation of the tangents or curvatures of the edge contours. However, it is pointed out as a weakness of this approach, compared to the other HT techniques in ellipse detection, that it becomes rather computationally intensive to select an appropriate pair of pixels on the major axis of the ellipse. Elmowafy [8] proposed fast graphical ellipse detection (FGED) algorithm, similar to RHT. In the first step, it estimates center of ellipse using the arbitrarily selected three pixels and tests its validity using a graphical validity method (GVM). GVM checks if the mid point of selected pixels is located between the estimated center and the intersection point of tangent of each selection pixels. Then, the ellipse center is found by the mid central point voting (MCPV) algorithm and the remained parameters are calculated by RHT. Finally, compute the ratio of the image pixels lying in the ellipse curve and the approximated circumference of the ellipse. If the ratio gives significant evidence of the existence of a real ellipse, the ellipse is recorded as a detected ellipse. This method shows very similar performance in terms of computation time, but it increases the detection accuracy by checking validity of ellipses using GVM. Although other approaches which are not based on Hough Transform but other theories such as genetic algorithms [11,12], still those approaches based on HT are widely used in ellipse detection.

This paper deals with the speed and accuracy problems in ellipse detection. It proposes a new method of grouping edge pixels by line segments and merging them if they are in the same ellipse. Whether two line segments are in the same ellipse or not is tested by comparing their parameters of RHT. Because it can estimate the exact number of ellipses in the image through the number of the merged segment set, it reduces significantly the probability of false ellipse detection. Also, this method can reduce the total execution time because it estimates the ellipse parameters in the line segment level not in the individual edge pixel level. This paper also solved the problem included in the previous paper [13] that the edge segment can be over segmented due to noise resulting in wrong detection and long computation time.

The rest of this paper is organized as follows: Section 2 describes the overview of the system and the preprocessing of an input image. Section 3 illustrates the line segment detection scheme and Section 4 explains how to use the line segments to detect ellipses using RHT. Section 5 shows the experimental results and Section 6 gives the conclusion.

2 System Overview and Preprocessing

2.1 System Overview

The block diagram of the proposed algorithm is summarized in Fig. 1. The algorithm begins with the edge detection and thinning process in the first stage to find the

boundaries of the objects in the image. The detected edge pixels are grouped by line segments using a corner pattern detector and individual line segments are labeled so that each of them may belong to only one ellipse later. In the second stage, every pair of edge segments is tested to see whether they satisfy the ellipse condition to merge. If so, they are merged. This test and merge process is repeated until all line segments are tested.

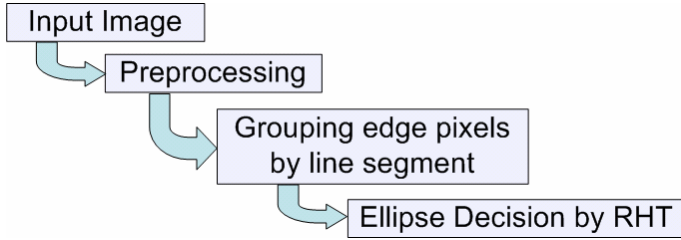


Fig. 1. Block diagram of the proposed algorithm

2.2 Preprocessing

The preprocessing includes the edge detection and thinning processes. For detecting edges in the input image, a modified Canny operator is used which removes noise using a Gaussian filter and estimates the local maximum edge point using the magnitude and orientation of the first differential of each pixel. Figure 2(b) is shows the result of applying the Canny edge detector for an input image given in Fig. 2(a).

For grouping the edge pixels by line segments to use for RHT, a thinning operation is required. In the thinning process, the effect of noise is considered. As shown in Fig. 2(c) and (d), some noise pixels connected to line segment can survive in the edge

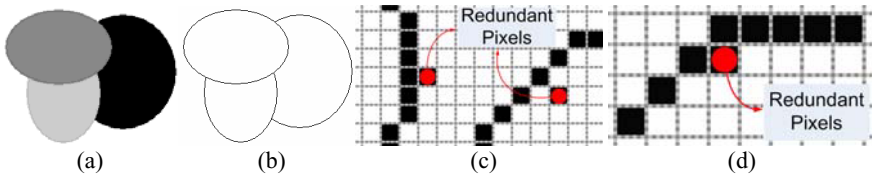


Fig. 2. Canny edge detector (a) Origin image (b) Edge image (c) First patterns of redundant pixels (d) Second patterns of redundant pixels

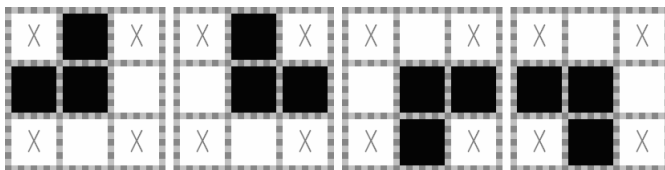


Fig. 3. Second redundant pattern, black is edge pixels, white is background and X is don't care

detection process. These noise pixels are defined as redundant pixels and they form some typical patterns as shown in Fig. 3. By removing these redundant pixels, the over segmentation problem can be significantly reduced.

3 Grouping Edge Pixels by Line Segments

To find ellipses using RHT, three edge pixels should be selected in the object image. As shown in Fig. 2, if there are ellipses more than one, the RHT has a high chance of selecting pixels from different ellipses to construct an ellipse, resulting in false ellipses with spending a large amount of processing time. To solve this problem, edge pixels are grouped by line segments first in this paper using the process given in Fig. 4 and the RHT uses these line segments instead of edge pixels to detect ellipses.

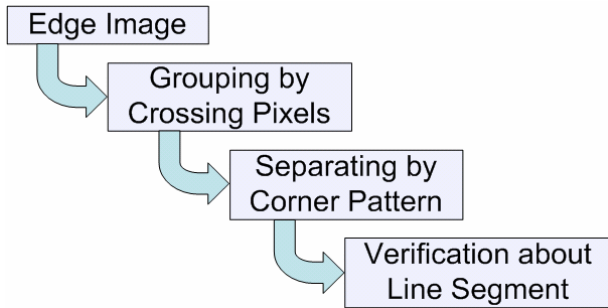


Fig. 4. Process of grouping edge pixels by line segments

To find line segments from the edge image, the process illustrated in Fig. 4 finds the crossing pixels in the first step. For this purpose, edge pixels are classified by four patterns, using the 8-connectivity window. The first one is the normal type having two neighboring edge pixels as shown in Fig. 5(a), the second one is the crossing type having more than three neighboring edge pixels as shown in Fig. 5(b), and the third one is the end type having one neighboring edge pixels as shown in Fig. 5(c).

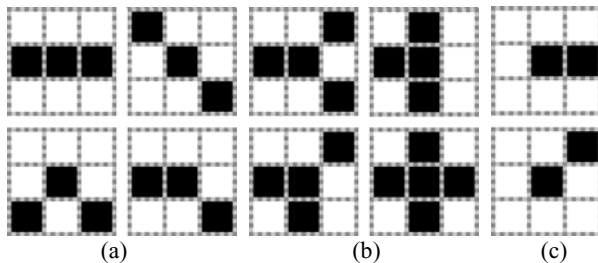


Fig. 5. Types of edge pixels (a) Normal (b) Crossing (c) End

The fourth one is a corner type to be defined by a difference chain code on the way of grouping process, since it cannot be determined simply by a 3x3 window. The difference chain code expresses an angle difference between two vectors, A and B, each of which can take one of 8 directional vectors shown in Fig. 6(a). The angle difference between these two vectors can be represented by one of 7 difference codes ranging from -3 to 3 including 0 as shown in Fig. 6(b). For example, Fig. 6(a) is represented code 1 by Fig. 6(b).

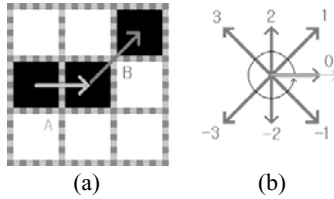


Fig. 6. Definition of a difference code (a) Vector definition (b) Code, bright narrow arrow is vector A in Fig. 6(a) and dark wide one is vector B

Figure 7 shows an example of describing a sequence of edge pixels using the difference chain code. A part of the edge image is described by the difference chain code.

By empirically testing the images, the difference chain code patterns of corners appearing in a 5x5 window are summarized in Fig. 8.

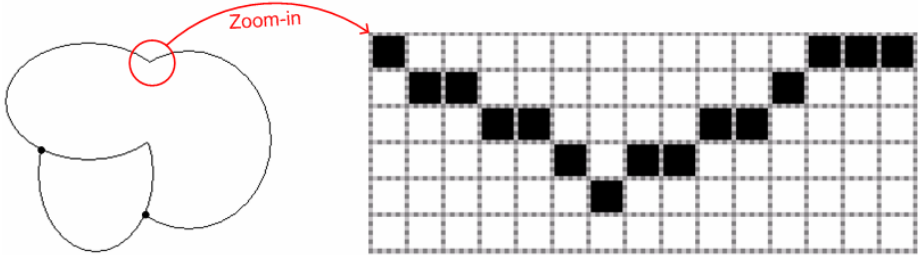


Fig. 7. Example of representing a curve using a difference chain code: 1-11-102-11-110-10

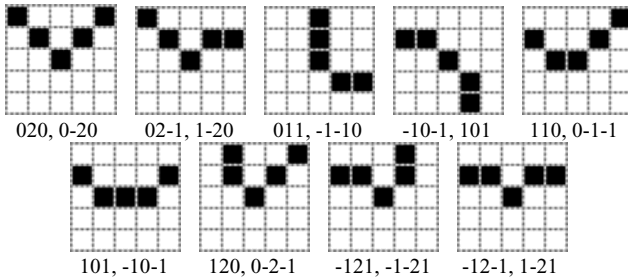


Fig. 8. Corner patterns in a 5x5 window

Based on the types of edge pixels, the grouping and labeling process begins with testing the leftmost top edge pixel in the input image to see which type it is. If it is a normal type, it is considered as a starting pixel of a new line segment and the edge pixel located first in the counterclockwise rotation with reference to X axis is selected to be tested until an edge pixel of crossing or end type is found, while merging the edge pixel as the same line segment if it is a normal type. Once a crossing or end type is found, the process returns to the first edge pixel of the line segment and begins the same procedure with the edge pixel located in the other direction. One line segment is completed when the tests along both neighboring edge pixels are finished. If the first edge pixel is a crossing type, it is the first pixel of three different line segments and its neighboring pixels are tested one by one repeatedly. Figure 9(b) shows the line segment image obtained by applying the proposed algorithm to the edge image in Fig. 9(a) where there is no edge pixel of corner type.

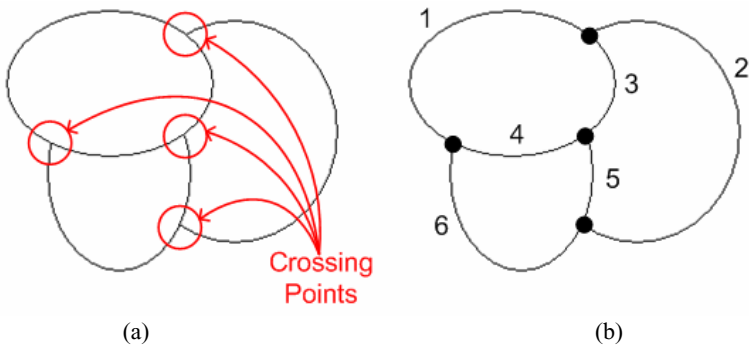


Fig. 9. Edge grouping based on crossing edges (a) Crossing edges (b) Labeled line segments

On the grouping process, if an edge pixel of corner type is found, then it stops the testing and merging operation. The edge pixel of corner type becomes the starting pixel of a new line segment from which the grouping process begins again. For example, let's consider an edge image given in Fig. 10(a) where edge pixels of corner type are included. In this case, if the pixels of corner type are not detected correctly, then the wrong line segments which include sudden curvature changes are resulted. In Fig. 10(b), line segments 1 and 2 include one corner pixel respectively.

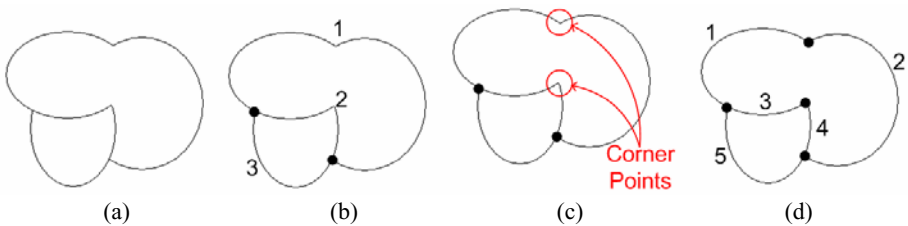


Fig. 10. Grouping process considering corner pixels (a) Edge image (b) Wrong result (c) Corner finding (d) Correct result

If representing the line segment 1 in Fig. 10(b) with the difference chain code, 1-11-102-11-110-10 is resulted where 02-1, one of corner patterns, is included. At the edge pixel where this pattern is found, the line segment is divided and the correct resulting line segments are obtained as given in Fig. 10(d).

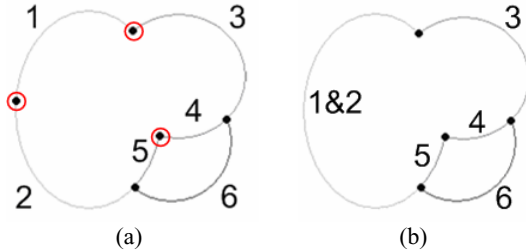


Fig. 11. Test of over segmentation (a) Over segmented line segment and verification point (b) Line segments are connected

Next step is to test whether a line segment is segmented by more than two line segments. Although the noise pixels that may cause over segmentation are eliminated by the thinning process, the survived noise pixels may result in over segmentation as shown in Fig. 11(a). To find the over segmented line segments, those verification points are selected first where the line segment is disconnected and the end points of the line segments are close enough. As shown in Fig. 11, among the five disconnected points given in Fig. 11(a) only 3 points are selected as the verification points which are marked by a circle as shown in Fig. 11(a). Since those two points connecting line segment 2 to line segments 5 and 6 and connecting line segments 4 and 6 to line segment 3 are branching points, they are not included in the verification points. The line segments located near the verification points are connected and the difference chain code is generated to see whether the connected line segment forms a corner pattern or not. If not, then the connected line segment is considered as one line segment. Fig. 11(b) shows the line segments where the over segmented line segments 1 and 2 are connected.

4 Ellipse Decision

4.1 Randomized Hough Transform

Ellipses in a X-Y plane can be completely represented by the following quadratic equation with five parameters (a,b,c,d,e) as other 2nd order curves can be.

$$ax^2 + by^2 + cxy + dx + ey + 1 = 0 \tag{1}$$

Eq. (1) can be restructured into Eq. (2) to explicitly include the coordinates of the ellipse center (p,q). Eq. (3) is given as the ellipse condition of Eq. (2). Still five parameters (A,B,H,p,q) are required to represent an ellipse.

$$A(x - p)^2 + 2H(x - p)(y - q) + B(y - q)^2 = 1 \tag{2}$$

$$AB - H^2 > 0 \tag{3}$$

Since as the number of parameters increases the computational complexity in Hough Transformation also does rapidly, Yuen et al. [2] proposed a method of representing an ellipse with 3 parameters to use for Randomized Hough Transformation. It takes two points (x_1, x_2) on an ellipse and finds their midpoint (m_1) and intersection (t_1) of their tangents, as shown in Fig. 12.

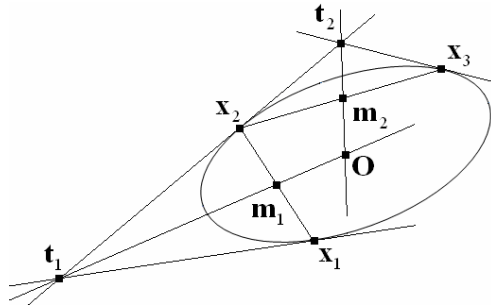


Fig. 12. Estimate center of ellipse

The line connecting m_1 and t_1 passes the center (O) of the ellipse. Therefore if three points (x_1, x_2, x_3) on an ellipse are given, its center can be determined as the intersection of two lines, and as shown in Fig. 11. Thus, the parameters (p, q) in Eq. (2) can be removed and Eq. (4) can be used to represent an ellipse where three parameters (A, H, B) are sufficient.

$$Ax^2 + 2Hxy + By^2 = 1 \tag{4}$$

These three parameters in Eq. (4) can be obtained in the parameter space if any three points on an ellipse are given in X-Y plane, using Eq. (5).

$$\begin{pmatrix} x_1^2 & 2x_1y_1 & y_1^2 \\ x_2^2 & 2x_2y_2 & y_2^2 \\ x_3^2 & 2x_3y_3 & y_3^2 \end{pmatrix} \begin{pmatrix} A \\ H \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{5}$$

4.2 RHT with Line Segments

To test if a pair of line segments belongs to the same ellipse or not, the process given in Fig. 13 is used where RHT takes the main role in ellipse decision.

In the first step, the line segments are sorted by length in a decreasing order and their ellipse parameters of RHT are calculated using Eq. (5). If a line segment satisfies the ellipse condition given in Eq. (3), it stays in the list. Otherwise, it is considered not a part of an ellipse and omitted from the list. The merging operation considers the

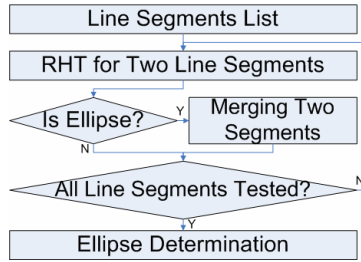


Fig. 13. Block diagram for ellipse decision

first, longest one in the list as the reference line segment. One of the rests in the list is selected and its ellipse matching ratio (EMR) defined in Eq. (6) is calculated.

$$EMR = \frac{N_c}{N_w} \tag{6}$$

In Eq. (6), N_w is the total number of pixels included in two line segments and N_c is the number of those pixels of segments pair that can be included in the estimated ellipse which is derived by RHT from the two line segments. Thus, N_c can be represented by Eq. (7).

$$N_c = \sum_{i=1}^{N_w} Near(i) \tag{7}$$

$$Near(i) = \begin{cases} 1 & \text{if } \overline{PQ} < th \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

In Eq. (8), if Q is the pixel to be tested and P is the pixel of the estimated ellipse at the intersection with the line from the center of the ellipse to Q , \overline{PQ} is the distance between the estimated ellipse and the pixel included in the selected line segment, as shown in Fig. 14(a). If the selected line segment has an EMR with the estimated ellipse within the threshold to be determined empirically, then it is merged to the reference line segment.

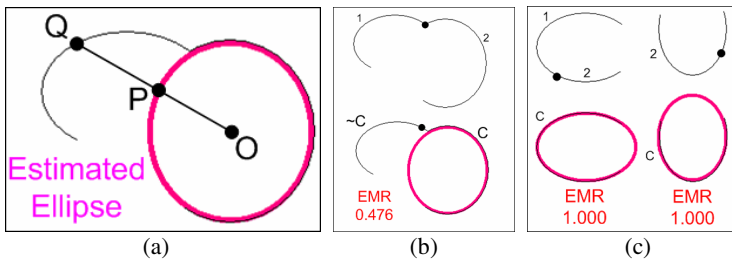


Fig. 14. Merging decision (a) Distance (b) Do not merging (c) Merging

Figure 14 shows the example images acquired as the results of the line segment merging. The upper row image in Fig. 14(b) includes two line segments and line segment 1 (longest one) is selected as the reference one to derive the reference ellipse C given in the bottom image. Then line segment 2 is selected and its EMR is calculated which is smaller than the threshold 0.95. Thus it cannot be merged with Line segment 1. Figure 14(c) shows the cases where the line segments can be merged.

5 Experiments

For proving the efficiency of the proposed algorithm, its performance has been compared with those of the RHT [5], Xie's algorithm [7], and fast graphical ellipse detection (FGED) [8] in terms of computation time and accuracy. For the experiments, total 450 synthetic images have been used which are collected in such a way that every 50 images contain 1 to 9 randomly positioned ellipses. Also, 100 real images collected in the fields are used for the experiments. Figure 15 shows one example of synthetic images where the images in the upper row contain four ellipses while those in the lower row do seven ellipses.

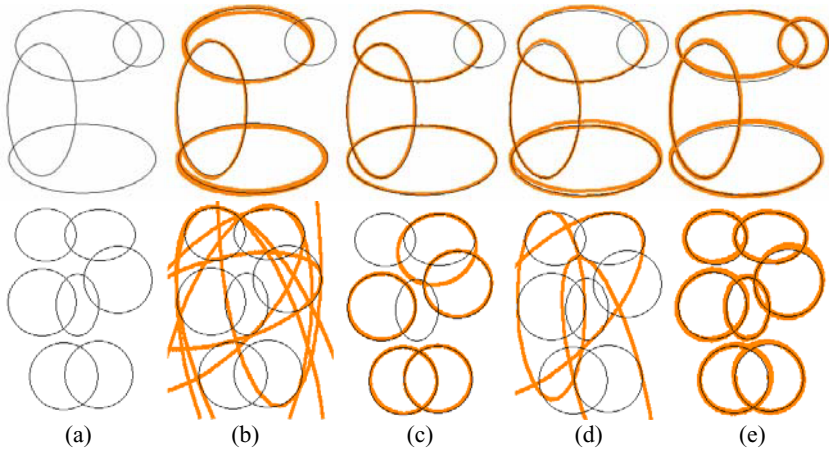


Fig. 15. Example of ellipse extraction in synthetic image, (a) Origin image, (b) RHT, (c) Xie's algorithm, (d) FGED, (e) Proposed method

In Fig. 15, the origin images are shown in (a) and the result images are given in (b)~(e) where the detected ellipses are described by gray thick lines. The images in the upper row show the result images containing four ellipses. The algorithms in (b) and (c) of Fig. 15 failed in detecting the small ellipse in the upper right corner and estimated the ellipses with visible error. Because RHT detects every ellipse if its voting is larger than a given threshold, one ellipse can be detected several times as shown in Fig. 15(b). The result image of FGED given in Fig. 15(d) looks like showing a better result compared to the other algorithms. But, its accuracy is not that good or

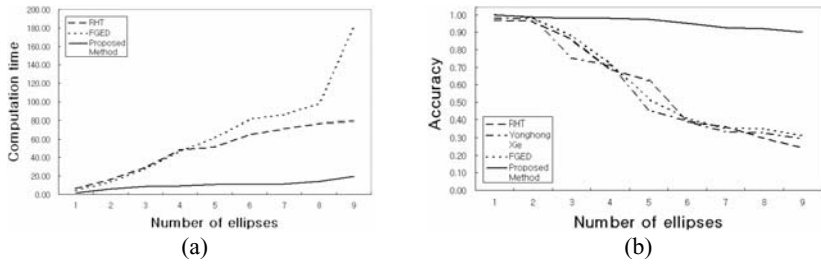


Fig. 16. Comparing the performance of RHT, Xie's algorithm, FGED and proposed method, (a) Computation time versus number of ellipse exact Xie's algorithm because it is so slow, (b) Accuracy versus number of ellipses

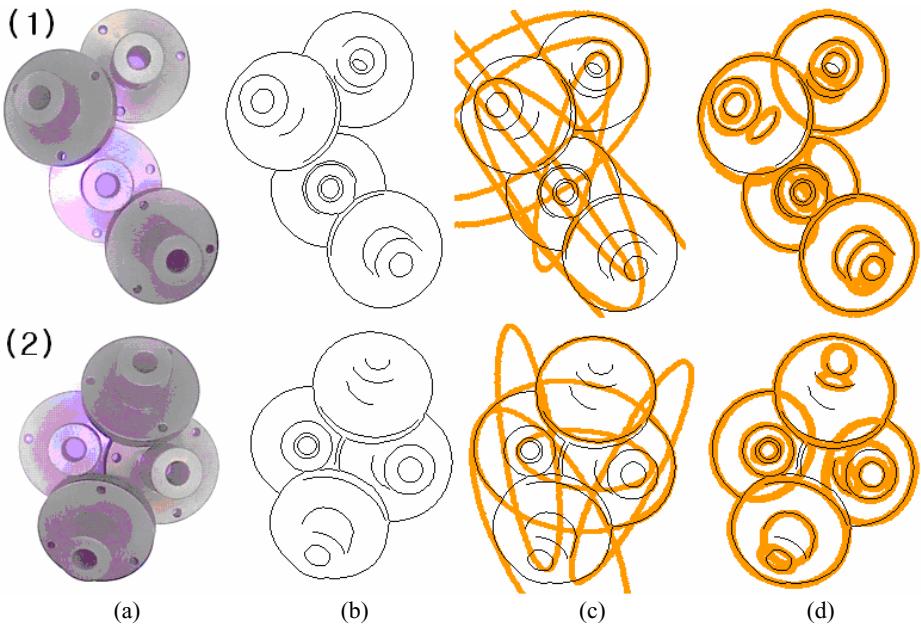


Fig. 17. Ellipse detection in real images, RHT and Xie's algorithm can not detect ellipse (a) Industrial parts image, (b) Edge image, (c) Result of FGED, (d) Result of proposed method

worse than theirs. Only the results image generated by the proposed method in Fig. 15(e) has extracted exactly 4 ellipses with the smallest error. The images in the lower row show the result images containing 7 ellipses. They show that the other algorithms except the proposed algorithm has detected too many ellipses or failed in detecting the ellipses.

The experiment results obtained from the synthetic images are summarized in Fig. 16 and Table 1. Figure 16(a) and (b) shows the computation time and detection

Table 1. Comparing the performance of RHT, Xie's algorithm, FGED and proposed method

| Number of ellipse in image | Computation Time (sec) | | | | Accuracy | | | |
|----------------------------|------------------------|--------|-------|-----------------|----------|-------|-------|-----------------|
| | RHT | Xie's | FGED | Proposed method | RHT | Xie's | FGED | Proposed method |
| 1 | 6.77 | 51.503 | 4.131 | 1.702 | 0.97 | 0.981 | 0.998 | 1.000 |
| 2 | 16.5 | 83.910 | 13.62 | 5.945 | 0.96 | 0.977 | 0.982 | 0.997 |
| 3 | 29.4 | 287.06 | 27.54 | 8.718 | 0.86 | 0.753 | 0.879 | 0.990 |
| 4 | 48.2 | 1386.5 | 46.69 | 9.398 | 0.69 | 0.713 | 0.725 | 0.988 |
| 5 | 51.4 | 1386.5 | 61.29 | 10.74 | 0.62 | 0.456 | 0.520 | 0.973 |
| 6 | 64.9 | 1419.9 | 81.20 | 10.95 | 0.39 | 0.390 | 0.409 | 0.953 |
| 7 | 71.4 | 1452.3 | 85.94 | 11.60 | 0.36 | 0.334 | 0.355 | 0.925 |
| 8 | 76.5 | 1985.8 | 97.85 | 14.36 | 0.30 | 0.329 | 0.351 | 0.920 |
| 9 | 79.4 | 2105.1 | 180.9 | 19.56 | 0.24 | 0.293 | 0.312 | 0.901 |

Table 2. Parameter values of ellipses detected in Fig. 17(d)

| Parameter values of ellipses in Fig. 17(d-1) | | | | | Parameter values of ellipses in Fig. 17(d-2) | | | | | | |
|--|------------|------------|-------------|----------|--|--------------------------|------------|-------------|----------|--|-------|
| Center | Major axis | Minor axis | Orientation | Accuracy | Center | Major axis | Minor axis | Orientation | Accuracy | | |
| (170, 85) | 51 | 47 | 42.03297 | 0.9212 | (181, 127) | 50 | 44 | 71.47059 | 0.9201 | | |
| (146, 238) | 51 | 47 | -59.8352 | 0.9284 | (61, 165) | 50 | 44 | 79.94118 | 0.9138 | | |
| (84, 199) | 54 | 47 | 81.59341 | 0.9564 | (132, 195) | 22 | 22 | 77.82353 | 0.9299 | | |
| (66, 210) | 24 | 20 | -64.7802 | 0.8914 | (130, 195) | 44 | 38 | -14.2941 | 0.9222 | | |
| (98, 120) | 34 | 20 | -54.8901 | 0.9057 | (108, 103) | 44 | 41 | -25.9412 | 0.8399 | | |
| (150, 243) | 14 | 10 | -76.6484 | 0.9366 | (112, 109) | 22 | 22 | 77.82353 | 0.9080 | | |
| (180, 76) | 24 | 20 | 56.86813 | 0.9013 | (132, 197) | 13 | 9 | -61.9412 | 0.9250 | | |
| (66, 210) | 14 | 10 | 53.9011 | 0.9048 | (112, 109) | 13 | 9 | -46.0588 | 0.9328 | | |
| (122, 137) | 24 | 24 | 89.5055 | 0.9003 | (134, 175) | 16 | 9 | 30.17647 | 0.9051 | | |
| (172, 85) | 24 | 20 | 33.13187 | 0.9524 | (196, 127) | 16 | 9 | 57.70588 | 0.9399 | | |
| (130, 137) | 27 | 24 | -5.43956 | 0.9075 | (205, 117) | 9 | 9 | 56.64706 | 0.9009 | | |
| (140, 227) | 17 | 10 | -31.1538 | 0.9011 | (68, 155) | 13 | 6 | 49.23529 | 0.9552 | | |
| (124, 137) | 10 | 10 | -45.989 | 0.9117 | (37, 169) | 13 | 13 | -71.4706 | 0.9142 | | |
| (182, 74) | 14 | 10 | 43.02198 | 0.9161 | (112, 111) | 9 | 9 | 43.94118 | 0.8604 | | |
| (90, 194) | 14 | 7 | 49.94506 | 0.9397 | | | | | | | |
| Computation Time (sec) : | | | | | 12.227 | Computation Time (sec) : | | | | | 9.503 |

accuracy in terms of the number of ellipses in the original image. As expected from the result images in Fig. 15, the results of all algorithms when the number of ellipses in an image is small look similar. However, as the number of ellipses in an image increases, the performance differences among the algorithms also increase significantly. The computation time of the proposed method increases twice proportional to the number of ellipses, but those of the others do four to ten times proportional to. The accuracy of the proposed method does not change much even when the number

of ellipses increases, but those of the others drop significantly as the number of ellipses increases.

Figure 17 shows the result images applied to the real images containing industrial parts randomly positioned. Differently from the synthetic images, the edge images contain many noise edges generated by shadows or by shapes. Because there exists many edge pixels inside the boundary ellipses, most other algorithms generate the result image as shown in Fig. 17(c) containing large error, but the proposed method does almost correct ones. The accuracy of the proposed algorithm on the real images is summarized in Table 2. It shows that proposed method extracts accurately the most ellipses in the real image with a reasonable time expense.

6 Conclusion

This paper has proposed a new ellipse detection algorithm using the RHT based on line segments. The algorithm intended to solve the problems of RHT which has a tendency of detecting false ellipses with edge pixels pertained to different ellipses and whose computational complexity depends on the number of edge pixels. These problems have been solved in this paper by reducing the number of inputs to RHT by grouping the edge pixels by line segments. The experimental results have shown that the proposed algorithm is superior to three conventional algorithms, RHT [5], Xie's algorithm [7] and FGED [8], by at least two times on average in terms of accuracy and processing time, even when ellipses are overlapped in an input image.

References

1. Hough, P.V.C.: Method and Means for Recognizing Complex Patterns. U.S. Patent 3069654 (December 18, 1962)
2. Yuen, H.K., Illingworth, J., Kittler: Detecting partially occluded ellipses using the Hough Transform. *Image and Vision Computing* 7(1), 31–37 (1989)
3. Kiryati, N., Eldar, Y., Bruckstein, A.M.: A probabilistic Hough Transform. *Pattern Recognition* 24(4), 303–316 (1991)
4. Jeng, S.-C., Tsai, W.-H.: Scale and orientation-invariant generalized Hough Transform-A new approach. *Pattern Recognition* 24(11), 1034–1051 (1991)
5. Xu, L., Oja, E., Kultane, P.: A new curve detection method: Randomized Hough Transform (RHT). *Pattern Recognition Letters* 11(5), 331–338 (1990)
6. McLaughlin, R.A.: Randomized Hough Transform: better ellipse detection. *Digital Signal Processing Applications* 1, 409–414 (1996)
7. Xie, Y., Ji, Q.: A new efficient ellipse detection method. In: *International Conference on Pattern Recognition*, vol. 2, pp. 957–960 (2002)
8. Elmowafy, O.M., Fairhurst, M.C.: Improving ellipse detection using a fast graphical method. *Electronics Letters* 35(2), 135–137 (1999)
9. McLaughlin, R.A., Alder, M.D.: The Hough transform versus the UpWrite. *Pattern Analysis and Machine Intelligence* 20(4), 396–400 (1998)
10. Cheng, Z., Liu, Y.: Efficient technique for ellipse detection using restricted randomized Hough transform. *Image Processing and Pattern Recognition* 2, 714–718 (2004)

11. Lutton, E., Martinez, P.: A genetic Algorithm for the detection 2D geometric primitives in image. In: Proc. 12th international conference on pattern recognition, October 1994, pp. 526–528 (1994)
12. Yao, J., Kharna, N., Grogono: Fast, robust GA-based ellipse detection. In: Proc. 17th International Conference on Pattern Recognition, ICPR 2004, vol. 2, pp. 859–862 (2004)
13. Hahn, K., Han, Y., Hahn, H.: Ellipse detection using a randomized Hough transform based on edge segment merging scheme. In: ISPRRA 2007 (2007)
14. Kälviäinen, H., Hirvonen, P.: Connective randomized Hough transform. In: The 9th Scandinavian conference on Image analysis, pp. 15–26 (1996)
15. Kälviäinen, H., Hirvonen, P., Xu, L., Oja, E.: Comparisons of probabilistic and non-probabilistic Hough transforms. In: The third European conference on Computer Vision, vol. 2, pp. 352–360 (1994)

Solving Decentralized Continuous Markov Decision Problems with Structured Reward

Emmanuel Benazera

Universität Bremen
Fachbereich 3 - AG/DFKI Robotic Lab
Robert-Hooke-Str 5 D-28359 Bremen, Germany
benazera@informatik.uni-bremen.de

Abstract. We present an approximation method that solves a class of Decentralized hybrid Markov Decision Processes (DEC-HMDPs). These DEC-HMDPs have both discrete and continuous state variables and represent individual agents with continuous measurable state-space, such as resources. Adding to the natural complexity of decentralized problems, continuous state variables lead to a blowup in potential decision points. Representing value functions as Rectangular Piecewise Constant (RPWC) functions, we formalize and detail an extension to the Coverage Set Algorithm (CSA) [1] that solves transition independent DEC-HMDPs with controlled error. We apply our algorithm to a range of multi-robot exploration problems with continuous resource constraints.

1 Introduction

Autonomous exploratory robots roam unknown environments where uncertainty is pervasive. For a single robot, a prevalent direct or indirect consequence of the uncertainty is observed in the form of highly varied resource consumption. For example, the terrain soil or slope directly affects a rover's battery usage: this potentially leads to high failure rates in the pursuit of rewarded objectives. In such a case, the best strategy is an optimal trade-off between risk and value. It comes in the form of a tree of actions whose branches are conditioned upon resource levels [6]. The multiagent problems can be represented as decentralized hybrid Markov decision problems (DEC-HMDPs). Versions of dynamic programming (DP) that solve centralized HMDPs use functional approximations to the true value functions. They can find near-optimal policies with controlled error [4,7,8].

Discrete decentralized MDPs (DEC-MDPs) have been proved to be of high computational complexity [2], ranging from PSPACE to NEXP-complete. A handful of recent algorithms can produce optimal solutions under certain conditions. [3] proposes a solution based on Dynamic Programming (DP), while [12] extends point based DP to the case of decentralized agents, and [13] applies heuristic search. Of particular interest to us in this paper, work in [1] has focused on *transition-independent* decentralized problems where agents do not affect each other's state but cooperate via a joint reward signal instead. However,

these algorithms are not targeted at the solving of DEC-HMDPs and cannot deal efficiently with the continuous state-space.

In this paper, we study the solving of transition independent DEC-HMDPs. Our theoretical approach remains within the brackets of [1]. This approach is attractive because it allows to consider each agent individually: the core idea is to compute the *coverage set* of each agent, that is the set that regroups the agent strategies such that all of the possible behaviors of the other agents are optimally covered. Therefore the continuous decision domains of individual agents can be decoupled during the computations. We extend the Coverage Set Algorithm (CSA) to DEC-HMDPs. We show how to aggregate those continuous Markov states for which an agent coverage set is identical. This eases the computations drastically as the algorithm considers far fewer points than a naive approach. The true joint value function of the multiagent problem is approximated with controllable precision. Our contribution is twofold. First, we bring the formulation and the computational techniques from the solving of HMDPs to that of DEC-HMDPs. Second, we solve problems of increasing difficulty in a multiagent version of the Mars rover domain where individuals act under continuous resource constraints. Results shed light on the relation between the computational complexity of the problems, and the level of collaboration among agents that is expressed in the domain.

In Section 2 and 3 we detail computational techniques for continuous and decentralized problems respectively, and formulate the general problem. In section 4 we present our solution algorithm. Section 5 provides a thorough analysis of the algorithm on decentralized planning problems with various characteristics. While to our knowledge this is the first algorithm to near optimally solve this class of DEC-HMDPs, much work remains, and we briefly outline future research threads in section 6.

2 Decision Theoretic Planning for Structured Continuous Domains

A single agent planning problem with a multi-dimensional continuous resource state is a special case of a hybrid MDP (HMDP).

2.1 Hybrid Markov Decision Process (HMDP)

A factored Hybrid Markov Decision Process (HMDP) is a factored Markov decision process that has both continuous and discrete states.

Definition 1 (Factored HMDP). *A factored HMDP is a tuple (N, X, A, T, R, n_0) where N is a discrete state variable [1], $X = X_1, \dots, X_d$ is a set of continuous variables, A is a set of actions, T is a set of transition functions, R is a reward*

¹ Multiple discrete variables plays no role in the algorithms described in this paper and for simplifying notations we downsize the discrete component to a single state variable N .

function. n_0 is the initial discrete state of the system, and $P_{n_0}(x)$ the initial distribution over continuous values.

The domain of each continuous variable $X_i \in X$ is an interval of the real line, and $X = \otimes_i X_i$ is the hypercube over which the continuous variables are defined. Transitions can be decomposed into the discrete marginals $P(n' | n, x, a)$ and the continuous conditionals $P(x' | n, x, a, n')$. For all (n, x, a, x') it holds $\sum_{n' \in N} P(n' | n, x, a) = 1$ and $\int_{x' \in X} P(x' | n, x, a, n') = 1$. The reward is assumed to be a function of the arrival state only, and $R_n(x)$ denotes the reward associated with a transition to state (n, x) . The reward function in general defines a set of goal formulas $g_j, j = 0, \dots, k$. Thus the non-zero $R_n(x)$ are such that $n = g_j$, and noted as the goal reward $R_{g_j}(x)$.

We consider a special case of HMDP in which the objective is to optimize the reward subject to resource consumption. Starting with an initial level of non-replenishable resources, an agent’s action consumes at least a minimum amount of one resource. No more action is possible once resources are exhausted. By including resources within the HMDP state as continuous variables, we allow decision to be made on resource availability. Typically time and energy are included as resources. This model naturally gives rise to *over-subscribed* planning problems, i.e. problems in which not all the goals are feasible by the agent under resource constraints. Over-subscribed problems have been studied in [11,14] for deterministic domains, and in [9] for stochastic domains. Here, it leads to the existence of different achievable goal sets for different resource levels. Each goal can be achieved only once (no additional utility is achieved by repeating the task), and the solving of the problem leads to a tree-like policy whose leaves contain achieved goals and branches depend on resource levels.

2.2 Optimality Equation

The Bellman optimality equation for a bounded-horizon HMDP is given by

$$\begin{aligned}
 &V_n(x) = 0 \text{ when } (n, x) \text{ is a terminal state.} \\
 &V_n(x) = \max_{a \in A_n(x)} \left[\sum_{n' \in N} P(n' | n, x, a) \int_{x'} P(x' | n, x, a, n') (R_{n'}(x') + V_{n'}(x')) dx' \right] \tag{2.1}
 \end{aligned}$$

where $A_n(x)$ is the set of eligible actions in state (n, x) , and a terminal state such that $A_n(x) = \emptyset$. Given a HMDP with initial state (n_0, x_0) , the objective is to find a policy $\pi : (N \times X) \rightarrow A$ that maximizes the expected cumulative reward. Given a policy π , $P_n(x | \pi_i)$ represents the state distribution over resources in state n under the policy π . It is given by

$$P_n(x) = P_{n_0}(x) \text{ when } (n, x) \text{ is the initial state.}$$

$$P_n(x | \pi_i) = \sum_{(n', a) \in \omega_n} \int_{X'} P_{n'}(x' | \pi_i) P(n | n', x', a) P(x | n', x', a, n) dx' \tag{2.2}$$

where $\omega_n = \{(n', a) \in N \times A : \exists x \in X, P_{n'}(x | \pi_i) > 0, \pi_i(x) = a\}$.

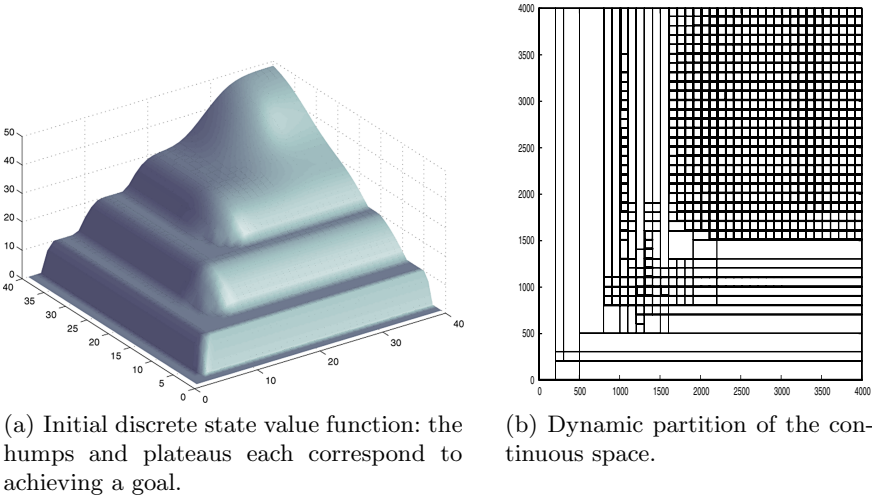


Fig. 1. Initial discrete state value function for an HMDP with two goals and two resources

Dynamic programming for structured HMDPs. Computationally, the challenging aspect of solving an HMDP is the handling of continuous variables, and particularly the computation of the continuous integral in Bellman backups and equations 2.1 and 2.2. Several approaches exploit the structure in the continuous value functions of HMDPs [4,7,8]. Typically these functions appear as a collection of humps and plateaus, each of which corresponds to a region in the state space where similar goals are pursued by the policy. The steepness of the value slope between plateaus reflects the uncertainty in achieving the underlying goals. Figure 1(a) pictures such a value function for a three goals, two resources problem. Taking advantage of the structure relies on grouping those states that belong to the same plateau, while naturally scaling the discretization for the regions of the state space where it is most useful such as in between plateaus. This is achieved by using discretized continuous action effects to partition the continuous state-space, thus conserving the problem structure. It contrasts with the naive approach that consist in discretizing the state space regardless of the relevance of the partition. Figure 1(b) shows the state grouping result for the value function on the left: discretization is focused in between plateaus, where the expected reward is the most subjected to action stochastic effects.

Theorem 1 (From [4]). *For an HMDP such that $V_{n'}(x)$ is Rectangular Piecewise Constant (RPWC), then $V_n(x)$ computed by Bellman backup (Equation 2.1) is also RPWC.*

Our implementation follows [4] and represents both value functions and continuous state distributions as kd-trees [5]. Tree leaves are RPWC values and probabilities. We have implemented the two main operations that solve equa-

tions 2.1 and 2.2 respectively, and that we refer to as the *back-up* and *convolution* operators.

3 Decentralized HMDPs (DEC-HMDPs)

Now, we consider any number of agents, operating in a decentralized manner in an uncertain environment, and choosing their own actions according to their local view of the world. The agents are cooperative, i.e. there is a single value function for all the agents. This value is to be maximized, but the agents can only communicate during the planning stage, not during the execution, so they must find an optimal joint policy that involves no exchange of information.

Definition 2 (Factored Locally Fully Observable DEC-HMDP). *An m -agents factored DEC-HMDP is defined by a tuple $(N, X, A, T, R, \hat{N}_0)$. N is a set of m discrete variables N_i that refer to each agent i discrete component, and n_i denotes a discrete state in N_i . $X = \bigotimes_{i=1}^m X_i$ is the continuous state space, and x_i denotes a continuous state in state-space X_i . $A = A_1 \times \dots \times A_m$ is a finite set of joint actions. $T = T_1 \times \dots \times T_m$ are transitions functions for each agent i . R is the reward function, and $R_n(x)$ denotes the reward obtained in joint state $(n; x)$. $N_0 = \{n_0^i\}_{i=1, \dots, m}$ are initial discrete states, with initial distributions over continuous values $\{P_{n_0^i}(x)\}_{i=1, \dots, m}$.*

In this paper, we focus on the class of *transition-independent* DEC-HMDPs where an agent actions have no effects on other agent states. However agents are not reward-independent.

3.1 Reward Structure

The reward function for a transition independent DEC-HMDP is decomposed into two components: a set of individual local reward functions that are independent; a global reward the group of agents receives and that depend on the actions of multiple agents. We assume a set of identified goals $\{g_1, \dots, g_k\}$, each of which is known and achievable by any of the m agents. Basically, a *joint reward structure* ρ is the distribution of a global reward signal over combinations of agents over each goal. In other words, ρ maps the achievement of a goal by possibly multiple agents to a global reward (or penalty), added to the system global value. For example, in the case of multiple exploratory robots, it would model a global penalty for when several robots try to achieve the same goal. The joint reward structure articulates the effect of individual agent actions on the global value function of the system. Thus the joint reward value for a set of given policies, one per agent, is a linear combination of the reward signals and the probability for each agent to achieve each of the goals. For simplifying the notations, we formally study the case of a single combination of agents per goal:

$$JV(\rho, x_1, \dots, x_m \mid \pi_1, \dots, \pi_m) = \sum_{j=0}^k c_j(x_1, \dots, x_m) \prod_{l=1}^m P_{g_j}(x_l \mid \pi_l) \quad (3.1)$$

where the c_j are the global reward signals of ρ , and the $P_{g_j}(x \mid \pi_i)$ naturally derive from equation 2.2 such that $P_{g_j}(x \mid \pi_i) = \sum_{n \text{ s.t. } n|=g_j} P_n(x \mid \pi_i)$. Now, the global value GV of a joint policy for all agents can be expressed as the joint gains of individuals and the global signals.

$$GV(x_1, \dots, x_m \mid \pi_1, \dots, \pi_m) = \sum_{i=1}^m V_{n_i^0}(x_i) + JV(\rho, x_1, \dots, x_m \mid \pi_1, \dots, \pi_m) . \quad (3.2)$$

The optimal joint policy thus writes

$$\{\pi_1^*(x_1), \dots, \pi_m^*(x_m)\} = \operatorname{argmax}_{\pi_1, \dots, \pi_m} GV(x_1, \dots, x_m \mid \pi_1, \dots, \pi_m) .$$

3.2 Cover Set Algorithm (CSA)

The influence of other agents on an agent’s policy can be captured by the probabilities these agents have to achieve each of the goals and thus rip some of the reward. We denote *subscription space* the $|k|$ -dimensional space of agent probabilities over goal achievements. In [1] this space is referred to as the parameter space. In a factored DEC-HMDP, the continuous state often conditions the reachability of the goals. Therefore the parameter space is resizable and related to the goal *subscription* of each agent. Points in the subscription space are referred to as subscription points.

Definition 3 (Subscription Space). *For two agents, the subscription space for k goals $\{g_j\}_{j=1, \dots, k}$ is a k -dimensional space of range $[0, 1]^k$. Each policy π over joint continuous state x corresponds to a subscription point s such that $s = [P_{g_1}(x), \dots, P_{g_k}(x)]$.*

Following [1], and for m MDPs, the CSA has three main procedures. Procedure *augment*(MDP_i, ρ, s), that creates an augmented MDP: an agent’s MDP with reward enriched with the joint reward ρ evaluated at a given subscription point s . Procedure *CoverageSet*(MDP_i, ρ) that computes the *optimal coverage set*, that is the set of all optimal policies for one agent, considering any possible policy applicable by the other agents. This set is found by sequentially solving series of augmented MDPs at carefully chosen subscription points. Given agent’s coverage sets, procedure *CSA*($MDP_1, \dots, MDP_m, \rho$) finds the optimal joint policy. The optimal joint policy returns the maximal global value.

Algorithm 1 puts the three main steps of the CSA together for a two agents problem. The algorithm remains similar for a larger number of agents. Unfortunately, by lack of space, we need to refer the reader to [1] for the necessary foundations and algorithmic details of the CSA.

4 Solving Transition-Independent DEC-HMDPs

This section extends the CSA from DEC-MDPs to DEC-HMDPs.

- 1: Input: MDP_1, MDP_2, ρ .
- 2: *optimal coverage set* $\leftarrow CoverageSet(MDP_1, \rho)$ (Algorithm 2).
- 3: **for all** policies π_1 in *optimal coverage set* **do**
- 4: *Augment*(MDP_2, ρ, s_1) with s_1 subscription point computed from π_1 .
- 5: policy $\pi_2 \leftarrow$ solve augmented MDP_2 .
- 6: $GV^* = \max(GV^*, GV(\pi_1, \pi_2))$.
- 7: return GV^* and optimal joint policy (π_1^*, π_2^*) .

Algorithm 1. Cover Set Algorithm for 2 agents 1 (CSA(MDP_1, MDP_2, ρ))

4.1 Policy Value in the Subscription Space

We assume the $R_{g_j}(x)$ and $c_j(x)$ of relation 3.1 are Rectangular Piecewise Constant (RPWC) functions of x . An RPWC is a rectangular partition of the state-space by a set of rectangles, each with a constant value. The agent transitions T_i are approximated with discrete (dirac) or RPWC functions, depending on the approximation method 4.7. DP leads to value functions that are represented by tuples (d_j, Δ_j) where the d_j are real, and the Δ_j form a rectangular partition of the state-space. Figure 3(a) depicts RPWC value functions. Following 1 an augmented $HMDP_i^{s_l}$ for agent i , for a subscription point s_l computed from a policy of agent l is such that:

- Its reward for goal g_j becomes: $R'_{g_j}(x_i) = R_{g_j}(x_i) + P_{g_j}(x_i | \pi_l)c_j(x_i)$.
- Given that individual agents' continuous state-spaces are independent, $P_{g_j}(x_i | \pi_l) = \int_{X_l} P_{g_j}(x_l)dx_l$ and $c_j(x_i) = \int_{X_l} c(x_i, x_l)dx_l$.

Given that individual agents' resource spaces are independent, $P_{g_j}(x_i | \pi_l) = \int_{X_l} P_{g_j}(x_l)dx_l$.

Theorem 2 (Adapted from 1). *If $R_{g_j}(x)$, $c_j(x)$ are RPWC for all goals $j = 0, \dots, k$ and states $n \in N$, the value of a policy π_i for $HMDP_i^{s_l}$ is Piecewise Linear (PWL) in the subscription space.*

Proof. From theorem 1, and suming over all goals:

$$V_{n_i^0}^{s_l}(x_i) = \sum_{j=0}^k P_{g_j}(x_i | \pi_i) R'_{g_j}(x_i) = \sum_{j=0}^k P_{g_j}(x_i | \pi_i) \left[R_{g_j}(x_i) + P_{g_j}(x_i | \pi_l) c_k(x_i, x_l) \right] \quad (4.1)$$

$$V_{n_i^0}^{s_l}(x_i) = V_{n_i^0}(x_i) + JV(\rho, x_i, x_l | \pi_i, \pi_l) \quad (4.2)$$

The optimal coverage set (OCS) is defined as the set of optimal solutions to augmented HMDPs.

Definition 4 (Optimal Coverage Set (OCS)). *The OCS for agent i is the set of optimal policies to augmented $HMDP_i^{s_l}$ such for any point s_l of the subscription space:*

$$OCS_i = \{\pi_i \mid \exists s_l, \pi = \underset{\pi'_i}{\operatorname{arg\,max}}(V_{\hat{n}_0}^{s_l}(x_i))\} \tag{4.3}$$

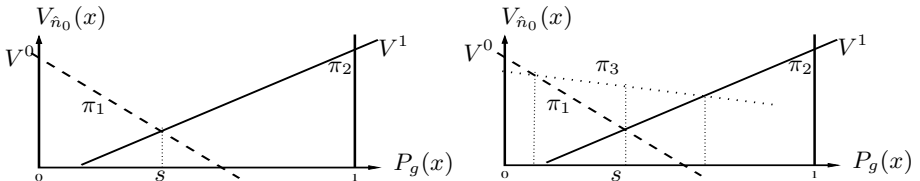
Relation 4.2 shows that solutions to augmented HMDPs form a set of linear equations if goal rewards and joint reward signals are RPWC. The solving of the linear equations yield the $P_{g_j}(x_i|\pi_l)$ that are the subscription points of interest. Given a policy $\pi(x_i)$ with value $V_{n_i^0}(x_i)$ for all $x_i \in X_i$, we note *planes*($\pi_i(X_i)$) the set of hyperplanes (planes in short) defined over the subscription space by equation 4.2

4.2 Computational Solution to the Discrete Problem

Let us consider an MDP with a *single resource fixed point*. Since the subscription space is continuous, there are infinitely many joint reward values w.r.t. this space. Thus there are infinitely many MDPs to be considered for each individual agent. However, those that are relevant correspond to policies of other agents. Thus there is a finite number of MDPs of interest. It follows that the coverage set must be finite. Theorem 2 ensures that the value of an agent policy w.r.t. other agent strategies is linear the subscription space of these agents' policies. This is also easily seen from relations 3.1 and 3.2, and it implies that the relevant subscription points can be found by solving sets of linear equations. These equations define hyperplanes that are agent optimal policy values w.r.t to the subscription space. Hyperplane intersections yield the subscription points of interest. Computing augmented MPDs at these points recursively builds the optimal coverage set of an agent. At these points, the agent must change policy w.r.t. to the subscription space. A case-study computation is pictured on figure 2.

4.3 Computational Solution to the Continuous Problem

Relaxing the hypothesis of a single fixed resource point produces an infinite number of policy values per subscription point. The original CSA cannot directly



(a) Corner points correspond to policies of other agents that achieve g with probability 0 and 1 respectively. V^0 and V^1 are values of the optimal policies for augmented MDP_i at corner points.

(b) Augmenting MDP_i at intersection point s yields yet another policy, and plane in the subscription space. Recursively intersecting and finding policies yields a convex set of planes that is the optimal cover set.

Fig. 2. Cover Set computation for an agent i over single goal, and a single resource point

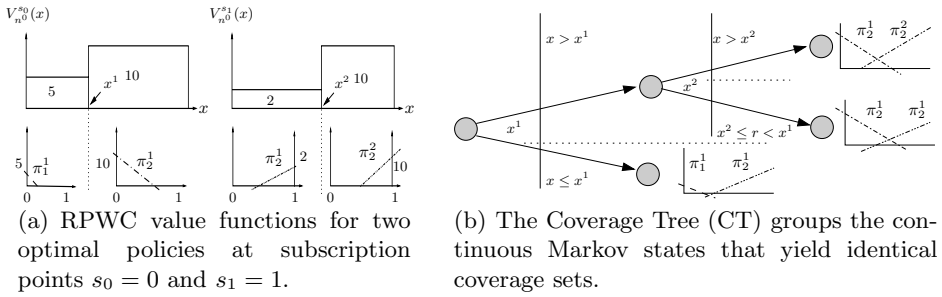


Fig. 3. Grouping the OCS over continuous Markov states with a Coverage Tree

deal with this problem since it would be forced to consider infinitely many linear equations. A possible extension to the CSA for solving equation 4.2 is to consider a high number of naively selected resource points. In practice this can work for problems of reduced size. But most notably, it does not take advantage of the structure of the continuous space w.r.t. to problem. However, several situations occur that can be exploited to build a more scalable solution:

1. As exposed in section 2, value functions in equation 4.2 exhibit a structure of humps of plateaus where continuous points can be regrouped.
2. Several agent policies have similar values over part of the resource space.
3. Similarly, agent policies can be dominated in sub-regions of the resource space.

Situation 1 is already exploited by the DP techniques mentioned in section 2. Situation 2 can potentially lead to great saving in computation, and this is precisely what our algorithm is designed for. To see this, consider a two agents problem with multiple goals. Imagine agent 1 often performs the same goal first as it lies near its initial position, and that it does so before dealing with remaining targets. In this case, many of the solution policies to augmented HMDPs could be regrouped over resources while w.r.t. different subscription points. In other words, identical sets of linear equations can cover entire regions of the continuous state-space. Situation 3 considers the dual problem where policies, and thus linear equations, are dominated in localized regions of the continuous space. Eliminating these policies locally is key to computational efficiency.

4.4 Implementation

Our implementation represents and manipulates value functions and probability distributions as kd-trees [5]. The number of continuous dimensions does not affect this representation. Each $V_{n_i}^{s_i}(x_i)$ calculated with equation 4.2 is also captured by a kd-tree. Therefore, an OCS translates into a special kd-tree whose leaves contain the linear equations in the subscription space defined by 4.2. We refer to this tree as the Coverage Tree (CT). Each node of the CT corresponds to a partition of the continuous state-space of an agent. Moving from the root

```

1: Initialize boundaries to  $P_{g_j} = 0; P_{g_j}(x) = 1$  for all  $j = 0, \dots, k$ .
2: Initialize coverage set to  $\emptyset$ , planes=empty CS, subscription points=find intersections of boundaries.
3: repeat
4:   for all  $s$  in subscription points do
5:     Remove  $s$  from subscription points.
6:     Do Augment( $HMDP_i, \rho, s$ ).
7:      $\pi_i^*(X_i) \leftarrow$  solve augmented  $HMDP_i$ .
8:     Do planes  $\leftarrow$  planes  $\cup$  planes( $\pi_i^*(X_i)$ ).
9:     Do Depth First Search in planes and prune dominated planes in leaves.
10:    If  $\pi_i^*(x_i)$  is not dominated over  $X$ , add  $\pi_i^*(x_i)$  to the coverage set.
11:    subscription points  $\leftarrow$  find intersection points of planes  $\cup$  boundaries.
12: until subscription points is empty

```

Algorithm 2. Coverage Set computation for $HMDP_i$ ($CoverageSet(HMDP_i, \rho)$).

toward the leaves yields smaller regional tiles. Within a tree leaf lay the linear equations that define the coverage set for continuous Markov states within this tile. Figure 3(b) pictures the CT for two RPWC functions at different subscription points, on figure 3(a). The linear equations are solved locally within each tile. Access to the leaves requires a depth-first search of the CT. The max operator of relation 4.3 leads to a pruning that is performed within each leaf. It thus removes dominated equations. Dominance is computed by solving a linear program. Whenever a policy is dominated in all leaves, it can be eliminated from the OCS.

Algorithm 2 computes the optimal coverage set of an HMDP. It recursively grows a set of subscription points of interest at which it solves augmented HMDPs. *planes* is represented by a CT. It groups continuous Markov states that are covered by identical sets of linear equations (step 8). It solves these equations (step 11), augments HMDPs (step 6) and solves them (step 7) by performing DP. Step 8 is a crucial source of complexity since it unionizes all policy planes in the subscription space over the continuous state-space. This union of planes is a kd-tree intersection [10]. The sequential processing of subscription points yields an equivalently denser mesh of continuous regions, and thus a deeper CT. To mitigate the blowup in the number of tree leaves, leaves with identical coverage sets are merged.

5 Results

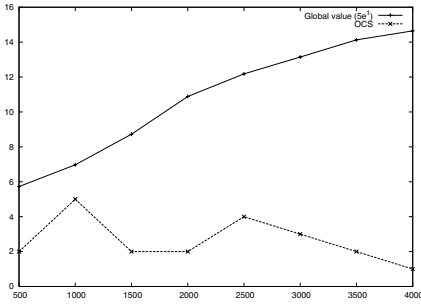
Experimental Domain. We consider a multi-robot extension to the Mars rover domain [6]. Robots have internal measurable continuous resource states and navigate among a set of locations. Some are goal locations where robots earn reward for performing analyses. Both navigation and analyses stochastically consume individual continuous resources, time and/or energy. Continuous state is irreversible since no action can be performed when resources are exhausted.

Therefore an optimal policy for a robot is in the form of a tree whose branches are conditioned upon resource values [6]. The Mars rover domain has been widely studied and results can be found in [1,9,4,7,8]. We study two variations of the reward structure: a *non collaborative* (NCL) form where for each goal that is achieved more than once the system suffers a penalty that is equal to that goal reward; a *collaborative* (CL) form where full reward can only be obtained by a succession of unordered analyses, one per robot.

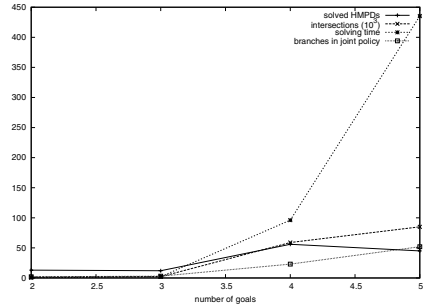
Planning results. We run our planner on several problems with varied parameters. Table 1 present results on NCL and CL problems. NCL problems exhibit a more complex structure than CLs: larger cover sets, high number of intersections, and more HMDPs solved. However, they produce optimal plans that are more compact and with less branching on resources than those produced for CL problems. This can be explained as follows. First, individual agents involved in a NCL problem are more dependent on the strategy of others, since they are forced to avoid goals possibly achieved by others. This leads to more intersected planes, and more meaningful subscription points. On the other hand, collaborative agents are facing more goals, and thus choose strategies with more branches,

Table 1. Results on exploration problems. m : number of agents; X :resource dimensions; G :number of goals; rds :reachable discrete states; rms :reachable Markov states given the problem's action discretization; ssp :subscription space size; ocs :optimal cover set size; I :number of computed plane intersections; $HMDPs$:number of solved HMDPs; pp :number of pruned policies; ph :number of pruned planes; $time$:solving time; b :number of branches in the optimal joint policy; $size$:number of actions in the optimal joint policy.

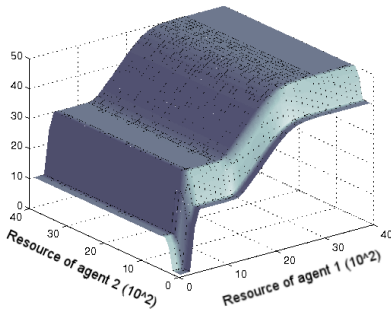
| Model | m | X | G | rds | rms | ssp | ocs | I | HMDPs | pp | ph | time (s) | br | size |
|-----------------------------------|---|---|---|------------------|------------------|-----|-----|--------|-------|------|--------|----------|----|------|
| Non Collaborative problems | | | | | | | | | | | | | | |
| sp2_1d | 2 | 1 | 2 | $\approx 2^{10}$ | $\approx 2^{18}$ | 2 | 3 | 420 | 16 | 10 | 12 | 0.29 | 2 | 22 |
| sp2_2d | 2 | 2 | 2 | $\approx 2^{10}$ | $\approx 2^{25}$ | 2 | 4 | 2560 | 38 | 30 | 3549 | 25.56 | 7 | 39 |
| sp3_1d | 2 | 1 | 3 | $\approx 2^{15}$ | $\approx 2^{24}$ | 3 | 27 | 46270 | 84 | 30 | 18055 | 104 | 4 | 36 |
| sp3_1d_3a | 3 | 1 | 3 | $\approx 2^{23}$ | $\approx 2^{30}$ | 9 | 17 | 770484 | 186 | 145 | 10762 | 233 | 11 | 87 |
| sp3_2d | 2 | 2 | 3 | $\approx 2^{15}$ | $\approx 2^{24}$ | 3 | 8 | 107555 | 19 | 3 | 5457 | 22.4 | 8 | 48 |
| sp4_1d | 2 | 1 | 4 | $\approx 2^{19}$ | $\approx 2^{28}$ | 4 | 31 | 121716 | 147 | 85 | 41816 | 1007 | 21 | 241 |
| sp5_1d | 2 | 1 | 5 | $\approx 2^{21}$ | $\approx 2^{27}$ | 5 | 32 | 746730 | 164 | 75 | 15245 | 2246 | 18 | 94 |
| sp5_1d_3a | 3 | 1 | 5 | $\approx 2^{32}$ | $\approx 2^{37}$ | 15 | 85 | 170M | 2735 | 1768 | 128570 | 44292 | 27 | 160 |
| Collaborative problems | | | | | | | | | | | | | | |
| sp2_1d | 2 | 1 | 2 | $\approx 2^{10}$ | $\approx 2^{18}$ | 2 | 2 | 300 | 8 | 4 | 18 | 0.16 | 2 | 22 |
| sp2_2d | 2 | 2 | 2 | $\approx 2^{10}$ | $\approx 2^{25}$ | 2 | 4 | 2240 | 52 | 44 | 5040 | 65.3 | 8 | 62 |
| sp3_1d | 2 | 1 | 3 | $\approx 2^{15}$ | $\approx 2^{24}$ | 3 | 4 | 21455 | 21 | 13 | 4946 | 30.55 | 8 | 78 |
| sp3_1d_3a | 3 | 1 | 3 | $\approx 2^{23}$ | $\approx 2^{30}$ | 9 | 17 | 592020 | 181 | 136 | 10642 | 164 | 11 | 101 |
| sp3_2d | 2 | 2 | 3 | $\approx 2^{15}$ | $\approx 2^{24}$ | 3 | 2 | 44905 | 13 | 4 | 7290 | 33.4 | 44 | 234 |
| sp4_1d | 2 | 1 | 4 | $\approx 2^{19}$ | $\approx 2^{28}$ | 4 | 7 | 83916 | 49 | 35 | 14084 | 359 | 35 | 313 |
| sp5_1d | 2 | 1 | 5 | $\approx 2^{21}$ | $\approx 2^{27}$ | 5 | 13 | 107184 | 62 | 38 | 5858 | 820 | 65 | 303 |
| sp5_1d_3a | 3 | 1 | 5 | $\approx 2^{32}$ | $\approx 2^{37}$ | 15 | 113 | 125M | 3679 | 1497 | 149074 | 72221 | 74 | 380 |



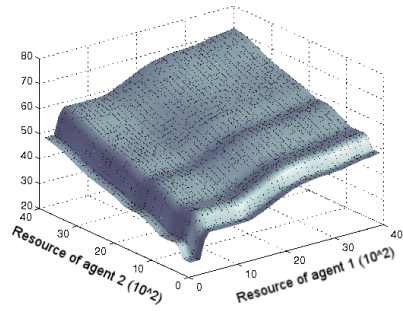
(a) Global value and optimal cover set size w.r.t. initial resources on a 3 goals problem with a single resource.



(b) Planning result w.r.t. the number of goals. We used a 5 goals problem and removed goals while keeping all locations.



(c) Non-Collaborative problem: agent 2 leaves two goals to agent 1. This accounts for the clear asymmetry in the value function.



(d) Collaborative problem: the symmetry of the function comes from the fact that both agents have incentive to visit all goals.

Fig. 4. Planning results (1)

and thus actions. However, in CL problems, goals are less constrained by other agents behavior, and greedy policies of individuals are more likely to be globally optimal. Overall, the small size of the optimal cover sets (OCS) is striking. The observed difference between that of CL and NCL problems is explained by the number of meaningful points in the subscription space. In NCL problems, agents must distribute the goal visits among themselves. A consequence is that numerous joint policies achieve the same maximum global plan value. In CL problems, the set of optimal joint policies (and global Nash equilibria) is reduced since agents have interest in visiting all goals.

A good measure of the efficiency of our algorithm is given by the number of pruned policies and planes. A naive approach that considers the isolated continuous Markov states would not group the planes, and thus not be able to perform the two pruning steps. Clearly, using the problem structure drastically eases the computational effort.

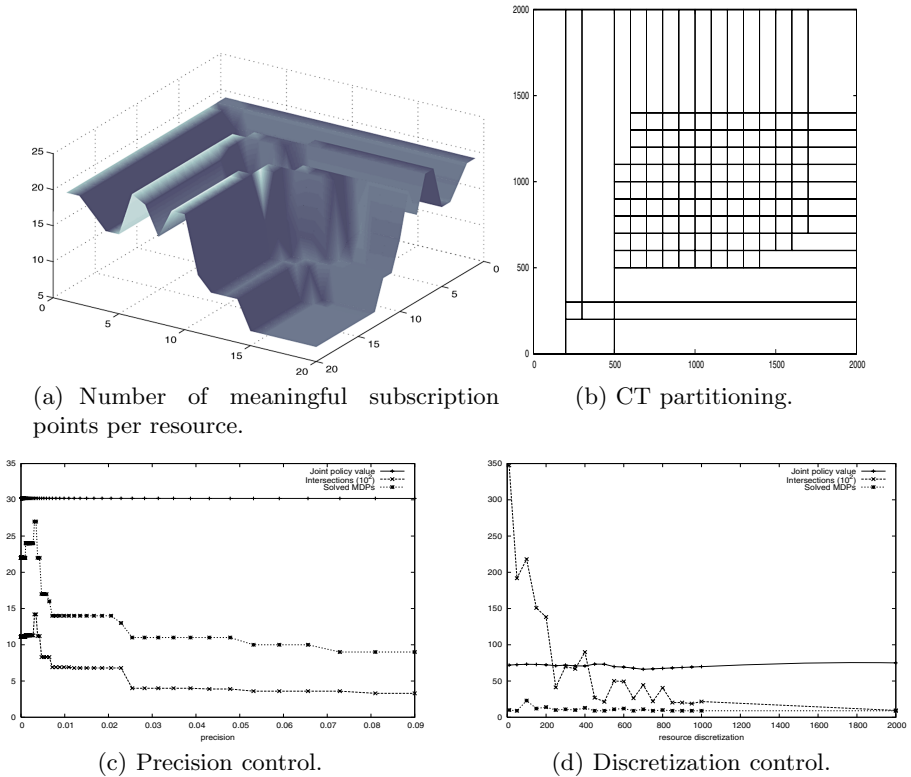


Fig. 5. Planning results (2)

Subscription space analysis. In over-subscription problems, the amount of initial resources determines the reachable states and goals. The size of the OCS is not a linear function of the initial resources (figure 4(a)). This is because the difficulty of a decision problem varies with the level of resources: as risk is high, optimal policies for agents with low resources tend to exhibit many branches. High level resources in general lead to simpler, because less constrained, policies. Thus similarly, the size of the OCS scales with the difficulty of achieving a good risk vs. reward trade-off. However, the number of goals, that is the size of the subscription space is expected to be a main driver of problems' complexity. Figure 4(b) pictures solving results w.r.t. the number of goals. Clearly, the solving time is crucially affected. Interestingly, the number of solved HMDPs drops with more goals. This is a typical consequence of the over-subscribed nature of the considered problems: adding more goals can ease up the local decision of individual agents by dominating the expected reward of more risky goals.

Global value function. Figures 4(c) and 4(d) respectively report the global value functions for both CL and NCL versions of a three goals problem with

two agents, and a single resource dimension per agent. The expected reward is better balanced among agents by the solution policy to the CL problem.

Subscription points per resource. The efficiency of our planner can be studied by looking at the resource partitioning achieved by the CST. Our planner regroups resources with identical cover sets. As seen above, decision is more difficult in certain regions of the resource space than in others. It follows that meaningful subscription points are not uniformly distributed in the resource space. Figure 5(a) pictures the number of meaningful points for a NCL problem with two resources. Interestingly, regions that correspond to high resource levels support very few subscription points. This means that for these levels, an individual agent can drive the game and is almost independent. Figure 5(b) pictures the partition of the resource space achieved by the CT for the same problem. As can be seen, the upper right corner of high resource levels is captured by a single tile.

Error control. Figure 5(d) reports on the algorithm’s behavior when varying the discretization step of the T_i . As expected, the number of plane intersections jumps exponentially as the step closes zero. The plan value does not vary significantly and this shows that the algorithm returns solutions that are close to the optimal. Similarly, varying the numerical precision on real values d_j does not affect the plan value significantly. This is reported on figure 5(c). These results are encouraging, and we believe our algorithm efficiently reduces the computational burden of finding near-optimal plans for DEC-HMDPs, and does so while remaining very close to the true functions.

6 Conclusion

We have formalized the extension of the CSA to agents modelled as HMDPs. To our knowledge, this paper reports on the first multiagent planner for agents with continuous independent state-spaces. Performances are promising, but much research remains, mostly on non transition independent DEC-HMDPs. We note that our technique does not apply straightforwardly to existing planners for DEC-MDPs and DEC-POMDPs [3,13,12]. This is because agent action spaces cannot be easily decoupled. However, we have reasons to believe that policies with equal expected reward in local continuous regions are often found around Nash equilibria. Therefore the grouping of individual agent policies appears to be a good prospect. We also note that other techniques for solving HMDPs such as [8] use analytical building blocks. They yield a usable partition of the continuous state-space and can be used to further improve our planner at no cost.

Acknowledgements. Emmanuel Benazera is supported by the DFG under contract number SFB/TR-8 (A3).

References

1. Becker, R., Zilberstein, S., Lesser, V., Goldman, C.V.: Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research* 22 (2004)
2. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research* 27(4) (2002)
3. Hansen, E., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: *Proceedings of the Nineteenth National Conference on Artificial Intelligence* (2004)
4. Feng, Z., Dearden, R., Meuleau, N., Washington, R.: Dynamic programming for structured continuous Markov decision problems. In: *Proceedings of the Twentieth International Conference on Uncertainty In Artificial Intelligence*, pp. 154–161 (2004)
5. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Mathematical Software* 3(3), 209–226 (1977)
6. Bresina, J., Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D., Washington, R.: Planning under continuous time and uncertainty: A challenge in ai. In: *Proceedings of the Eighteenth International Conference on Uncertainty In Artificial Intelligence* (2002)
7. Li, L., Littman, M.L.: Lazy approximation for solving continuous finite-horizon mdps. In: *Proceedings of the Twentieth National Conference on Artificial Intelligence* (2005)
8. Marecki, J., Koenig, S., Tambe, M.: A fast analytical algorithm for solving markov decision processes with real-valued resources. In: *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (2007)
9. Mausam, Benazera, E., Brafman, R., Meuleau, N., Hansen, E.A.: Planning with continuous resources in stochastic domains. In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 1244–1251 (2005)
10. Naylor, B., Amanatides, J., Thibault, W.: Merging bsp trees yields polyhedral set operations. In: *Computer Graphics (SIGGRAPH'90)* (1990)
11. Smith, D.: Choosing objectives in over-subscription planning. In: *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, pp. 393–401 (2004)
12. Szer, D., Charpillet, F.: Point-based dynamic programming for dec-pomdps. In: *Proceedings of the Twenty First National Conference on Artificial Intelligence* (2006)
13. Szer, D., Charpillet, F., Zilberstein, S.: Maa*: A heuristic search algorithm for solving decentralized pomdps. In: *Proceedings of the Twentieth National Conference on Artificial Intelligence* (2005)
14. van den Briel, M., Do, M.B., Sanchez, R., Kambhampati, S.: Effective approaches for partial satisfaction (over-subscription) planning. In: *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 562–569 (2004)

Options in Readylog Reloaded – Generating Decision-Theoretic Plan Libraries in Golog

Lutz Böhnstedt, Alexander Ferrein, and Gerhard Lakemeyer

RWTH Aachen University
Knowledge-Based Systems Group
Ahornstrasse 55
52056 Aachen

lutzboehnstedt@gmx.de, {ferrein, gerhard}@cs.rwth-aachen.de

Abstract. READYLOG is a logic-based agent programming language and combines many important features from other GOLOG dialects. One of the features of READYLOG is to make use of decision-theoretic planning for specifying the behavior of an agent or robot. In this paper we show a method to reduce the planning time for decision-theoretic planning in the READYLOG framework. Instead of planning policies on the fly over and over again, we calculate an abstract policy once and store it in a plan library. This policy can later be re-instantiated. With this plan library the on-line planning time can be significantly reduced. We compare computing policies on the fly with those stored in our plan library with examples from the robotic soccer domain. In the 2D soccer simulation league we show the significant speed-up when using our plan library approach. Moreover, the use of the plan library together with a suitable state space abstraction for the soccer domain makes it possible to apply macro-actions in an otherwise continuous domain.

1 Introduction

The logic-based agent programming language Golog is a language for autonomous robots or agents acting in real-world domains. One of its advantages is that it supports deliberation in form of projecting the world state into the future or perform decision-theoretic (DT) planning to choose the next action to be performed by the agent in a rational way. In recent years several extensions to the original GOLOG dialect have been proposed to increase the expressiveness of the language to cope with real-world problems. Such extensions deal with sensors, continuous change, probabilistic projections, and decision-theoretic planning, to name but a few. The language READYLOG [1] integrates many of the useful GOLOG features proposed in the literature. Its usefulness and applicability as a programming language for agents acting in dynamic real-time domains has been shown with robotic soccer and service robotics applications [1,2,3,4].

In this paper we investigate the decision-theoretic planning approach adopted originally from DTGOLOG [5] and extend the forward-search algorithm which is used in READYLOG. Decision-theoretic planning in READYLOG, roughly, works

as follows. Given an input program which leaves open several action alternatives for the agent is interpreted and an optimal policy for the input program is generated. Formally, a Markov Decision Process (MDP, cf. e.g. [6]) is solved. The transition function between states of the Markov chain is given by Reiter’s variant of the basic action theory formalized in the underlying situation calculus [7,8], the policy is calculated with an optimization theory consisting of a reward function and possibly transition probabilities for the action outcomes (cf. also [5,9]).

This means that each time a program is interpreted in a decision-theoretic fashion all outcomes of the used stochastic actions are expanded, for each choice point the optimal value of being in this particular state of the MDP has to be calculated, because these values have to be calculated relative to the world situation when the optimization is invoked. The question we posed was whether there is an efficient way to store intermediate results of previous calculations of an optimal policy in such a way that it can be re-used later. This re-using of previously calculated parts of a policy should also save computation time, though at the cost of higher memory consumption. But considering an agent acting in a dynamic domain, where decisions have to be taken quickly, this trade-off seems to be worthwhile. In this paper we show that this is indeed feasible. It is possible to store the computation of a policy independent from a particular world situation in an efficient way such that it can be re-used later on; and moreover, re-used with savings in the computation time. We show our modifications to the forward-search algorithm used in READYLOG which replaces the standard MDP value iteration method and show how a decision-theoretic plan library can be established for a particular application domain in a bootstrap fashion. With this plan library it is in particular simple to extend an existing macro-actions or options (as these are sometimes called in the MDP context) approach in READYLOG [10]. With our approach we also overcome limitations of the original approach proposed in [10] as the original proposal relied on explicit state enumeration which is not necessary any longer with our novel approach. Together with a suitable state space abstraction options become applicable also in continuous domains like robotic soccer where the original approach failed.

The rest of the paper is organized as follows. In Section 2 we introduce the language READYLOG in greater detail, focusing on decision-theoretic planning and options in their current form. Section 3 presents the modified forward-search algorithm which is capable to generate and store abstract DT policies while Section 4 focuses on the re-instantiation of these abstract policies and on building the DT plan library. Further, we show how the previously proposed options approach can be modeled in our new framework. In Section 5 we show several experimental results from the robotic soccer domain where a soccer plan library is exemplarily established. We demonstrate the run-time gain with our new method by comparing the library instantiation with planning the same policies each time from scratch. We conclude with Section 6.

2 Readylog

READYLOG [21], a variant of GOLOG, is based on Reiter’s variant of the situation calculus [8,7], a second-order language for reasoning about actions and their effects. Changes in the world are only due to actions so that a situation is completely described by the history of actions starting in some initial situation. Properties of the world are described by *fluents*, which are situation-dependent predicates and functions. For each fluent the user defines a successor state axiom specifying precisely which value the fluent takes on after performing an action. These, together with precondition axioms for each action, axioms for the initial situation, foundational and unique names axioms, form a so-called *basic action theory* [8].

GOLOG has imperative control constructs such as loops, conditionals [11], and recursive procedures, but also less standard constructs like the nondeterministic choice of actions. Extensions exist for dealing with continuous change [12] and concurrency [13], allowing for exogenous and sensing actions [14] and probabilistic projections into the future [15], or decision-theoretic planning [5] which employs Markov Decision Processes (MDPs). READYLOG integrates these extensions in one agent programming framework. In the following we focus on the decision-theoretic planning with READYLOG.

2.1 DT Planning with Readylog

To illustrate how READYLOG calculates an optimal policy from a given input program we give an navigation example from a toy maze domain. A robot should navigate from its start position S to a goal position G . It can perform one of the actions from the set $A = \{go_right, go_left, go_up, go_down\}$. Each of the actions brings the robot to one of its neighboring locations. The actions are stochastic, that is there exists a probability distribution over the effects of the action. Each action takes the agent to the intended field with probability of p , with probability $1 - p$ the robot will arrive at any other adjacent field. The maze shown is the well-known Maze66 domain from [16]. In our example $p = 0.7$ which means that the action *right* will succeed with probability 0.7, and with probability of 0.1 nature chooses one of the actions *left*, *up*, and *down*. The robot cannot go through the walls, if it tries, though, the effect is that it does not change its position at all.

Accordingly, the basic action theory consists of the fluents *loc*, *start* and *goal*, and the stochastic actions *go_right*, *go_left*, *go_up*, and *go_down*. As these are stochastic actions we have to provide the predicates *choice*(A, a, s) and *prob*(n, a, s). This means that we have to define a set of deterministic actions r, l, u, d from which nature chooses when performing one of our navigation actions. For ease of notation we assume that the outcomes for each action remain the same in each situation, i.e.

$$choice(go_right) = choice(go_left) = choice(go_up) = choice(go_down) \stackrel{def}{=} \{r, l, u, d\}.$$

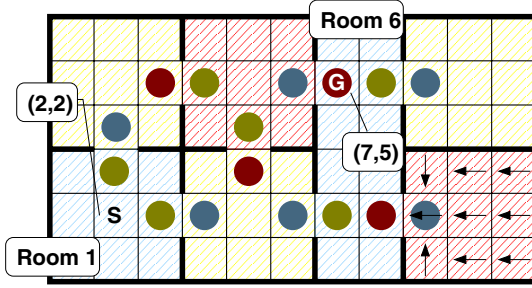


Fig. 1. Options for the Maze66 domain from [16]

The probability for each outcome is then

$prob(r, go_right) = prob(l, go_left) = prob(u, go_up) = prob(d, go_down) \stackrel{def}{=} 0.7$ and $prob(n, A, s) \stackrel{def}{=} 0.1$ for the remaining action pairs [16]. The successor state axiom for the location fluent is defined as

$$\begin{aligned}
 loc(do(a, s)) = (x, y) &\equiv \\
 \exists x', y'. loc(s) = (x', y') \wedge & \\
 ((a = r \wedge x = x' + 1 \wedge y = y') \vee (a = l \wedge x = x' - 1 \wedge y = y') \vee & \\
 (a = u \wedge x = x' \wedge y = y' + 1) \vee (a = d \wedge x = x' \wedge y = y' - 1) \vee & \\
 (a \neq r \wedge a \neq l \wedge a \neq u \wedge a \neq d \wedge x = x' \wedge y = y')) &
 \end{aligned}$$

The fluents *start* and *goal* are situation independent and encode only the position of the start and the target position. In our example $start(s) = (1, 1)$ and $goal(s) = (7, 5)$. The reward function is defined as $reward(s) = +1$ if $loc(s) = goal(s)$ and -1 otherwise. To find the optimal path from *S* to *G* the robot is equipped with the program

```

proc navigate
  solve(while  $\neg loc = goal$  do
    ( $go\_right \mid go\_left \mid go\_up \mid go\_down$ )
  endwhile, h)
endproc

```

With the *solve* statement decision-theoretic planning is initiated. The interpreter switches into an off-line mode and optimizes the program given as the argument of the solve-statement up to horizon *h*. The “|” represent nondeterministic choices of actions. At these choice points the interpreter selects the best action alternative. The READYLOG interpreter does this via predicates *BestDo*

¹ Again, for ease of notation, we do not distinguish between different partitions of the state space. As all probabilities have to sum up to 1 for each action, the probability mass of a stochastic action has to be redistributed over the possible outcomes. This means that at position (1,1) only the actions *r* and *u* are possible and thus the probability of the outcome for *r* is 0.875 and for the outcome *u* is 0.125.

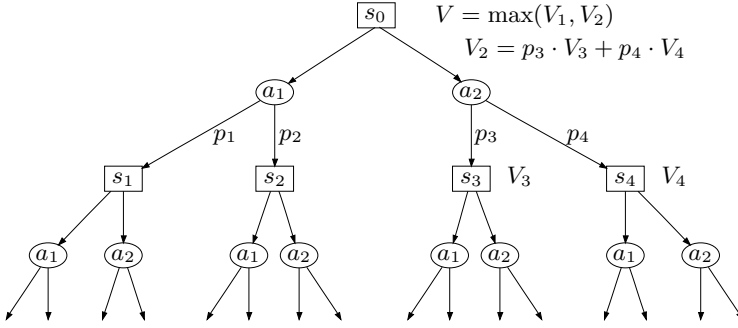


Fig. 2. Decision tree search in READYLOG

which implement the forward-search algorithm (Fig. 2). For space reasons we will not show the whole definition of the algorithm here. For a detailed discussion of *BestDo* we refer to [5,2]. As long as the robot is not at the goal location (and the horizon is not reached) READYLOG loops over the nondeterministic choice statement. At each iteration the interpreter expands a subtree for each of the actions inside the choice statement. As each of the actions are stochastic ones, again for each outcome of each action the interpreter branches over the nature’s choices. This goes on until either the agent is located at the goal position or the horizon is reached. At the leaves of the computation tree over *BestDo* (at the end of the recursion) the agent receives the reward for these final situations. Then, “going up” the computation tree for nondeterministic choices, the best alternative is evaluated and chosen for the policy. An illustration of the computation tree is given in Fig. 2. Since it is not known in advance which outcome is chosen by nature at execution time, the policy need to cover all possibilities, which is realized by nested conditionals. Coming up to the root node the computation terminates returning the policy, the value for the policy, and its probability of success.

To give an example for *BestDo* we show how a stochastic action is interpreted in the following.

$$\begin{aligned}
 BestDo(a; p, s, h, \pi, v, pr) &\stackrel{def}{=} \\
 &\exists \pi', v'. BestDoAux(choice'(a), a, p, s, h, \pi', v', pr) \wedge \\
 &\pi' = a; senseEffect(a); \pi' \wedge v = reward(s) + v'
 \end{aligned}$$

The resulting policy is $a; senseEffect(a); \pi'$. The pseudo action *senseEffect* is introduced to fulfill the requirement of full observability of the underlying MDP (cf. e.g. [6] for details about MDP theory). The remainder policy π' branches over the possible outcomes and the agent must be enabled to sense the state it is in after having executed this action. The remainder policy is evaluated using the predicate *BestDoAux*. The predicate *BestDoAux* for the (base) case that there is one outcome is defined as

$$\begin{aligned}
& \text{BestDoAux}(\{n_k\}, a, \delta, s, h, \pi, v, pr) \stackrel{\text{def}}{=} \\
& \neg \text{Poss}(n_k, s) \wedge \text{senseCond}(n_k, \varphi_k) \wedge \pi = \varphi_k?; \text{Stop} \wedge v = 0 \wedge pr = 0 \vee \\
& \text{Poss}(n_k, s) \wedge \text{senseCond}(n_k, \varphi_k) \wedge \\
& \quad \exists \pi', v', pr'. \text{BestDo}(\delta, do(n_k, s), h, \pi', v', pr') \wedge \\
& \quad \pi = \varphi_k?; \pi' \wedge v = v' \cdot \text{prob}(n_k, a, s) \wedge pr = pr' \cdot \text{prob}(n_k, a, s)
\end{aligned}$$

If the outcome action is not possible, the *Stop* action is inserted into the policy and no further calculations are conducted. Otherwise, if the current outcome action is possible the remainder policy π' for the remaining program is calculated. The policy π consists of a test action on the condition φ_k from the *senseCond* predicate with the remainder policy π' attached. *senseCond* define mutually exclusive conditions to distinguish between the possible outcomes n_1, \dots, n_k of a stochastic action. The case for more than one remaining outcome action is defined as

$$\begin{aligned}
& \text{BestDoAux}(\{n_1, \dots, n_k\}, a, p, s, h, \pi, v, pr) \stackrel{\text{def}}{=} \\
& \neg \text{Poss}(n_1, s) \wedge \text{BestDoAux}(\{n_2, \dots, n_k\}, p, s, h, \pi, v, pr) \vee \\
& \text{Poss}(n_1, s) \wedge (\exists \pi', v', pr'). \text{BestDoAux}(\{n_2, \dots, n_k\}, p, s, h, \pi', v', pr') \wedge \\
& \quad \exists \pi_1, v_1, pr_1. \text{BestDo}(p, do(n_1, s), h - 1, \pi_1, v_1, pr_1) \wedge \text{senseCond}(n_1, \varphi_1) \\
& \quad \pi = \text{if } \varphi_1 \text{ then } \pi_1 \text{ else } \pi' \text{ endif } \wedge \\
& \quad v = v' + v_1 \cdot \text{prob}(n_1, a, s) \wedge pr = pr' + p_1 \cdot \text{prob}(n_1, a, s)
\end{aligned}$$

The robot is now endowed with a conditional program which tells it, for each of the positions it can reach, which is the best action to advance to the goal position. One has to remark that the policy yields only an action for the projected locations the robot reached during planning, and up to the given fixed horizon. It does not have any idea which action to take at location, say (10, 4). This might at first sight seem to be a disadvantage, but on second thought this turns out as one of the advantages of READYLOG. With standard solution techniques to MDPs like value iteration one would have a solution for each of the locations, but for the cost that value iteration has to iterate several times over all states. With READYLOG this can be avoided by only expanding the reachable successor locations.

2.2 Options with Readylog

Macro-actions in the decision-theoretic context are referred to as options, based on a definition by [17]. The idea is, roughly, to define a partition of the state space of the MDP and find an optimal solution for this partition. Macro-actions based on the work of [17][16] have been introduced into READYLOG by [10]. Here, the idea is to find sub-tasks, solve these sub-tasks in an optimal way, and apply these macro-actions and their solution, resp., for a more complex problem. [10] shows an exponential speed-up when using options in READYLOG compared to using stochastic actions only.

In our grid world example we have seen that in the full planning approach the agent can choose between four actions which have four possible outcomes each.

Thus, at each stage of the algorithm 16 nodes have to be expanded. Solving the navigation task for large domains becomes infeasible, even with the forward-search approach used in READYLOG. Therefore one has to identify appropriate sub-tasks to reduce the complexity of the task.

Sub-tasks for finding the way to the goal are to leave certain rooms and enter other ones. Accordingly, to reach the goal from the start state “S” one possibility is to execute the action sequence *leaveroom_1_north*; *leaveroom_2_east*; *leaveroom_4_east*. Figure 1 depicts the situation. There are two possibilities to leave room 1: through the northern or the eastern entrance. The blue spots in Figure 1 depicts the two possibilities. The other entrance fields are marked with spots in the respective room color. This example shows that only three actions are needed to reach the goal, instead of minimal 8 actions when using the basic actions which we introduced previously. With these actions, obviously, we can reduce the number of expanded nodes in the calculation tree when searching for the optimal policy for the navigation problem.

Clearly, one could define basic actions for leaving a room. Then, one additionally has to specify the behavior the agent should take when it is located inside a room. But this is not needed as we can make use of decision-theoretic planning. This, moreover, yields optimal behavior for leaving a room. We can relax the original problem to the problem of leaving Room 1, solve this problem and save the policy, the value, and the probability of success. Later, when solving the original problem we could use the results of solving the MDP which leaves room 1. Figure 1 shows the solution of one of the identified sub-tasks for Room 6. The arrows represent the optimal policy for leaving Room 6.

As Room 1 has two different doors to neighboring rooms we have to define two different options, one for leaving through the northern door, and one for leaving the room through the eastern door. Why do we have to distinguish between both doors? The reason is even if we want to leave the room through the northern door, it might be the case that the agent ends up in room 3 due to failing basic actions. Consider the agent being located on position (3, 2). The optimal policy should take the agent to position (3, 3) with a *go_up* action. But if the action fails and nature chooses a *go_right* action we end up in room 3. This is obviously not what we wanted. Therefore, this case should be declared as a failure case for the macro action *leaveroom_1_north*. For each room we have to identify one macro action for each door, solve the MDP of the sub-task and store the result. The results can then be re-used for solving the task of reaching the goal position.

3 Solving Decision-Theoretic Plans in an Abstract Way

As we have sketched in the previous section READYLOG implements the forward-search algorithm introduced in DTGOLOG [5]. This means that from a program including nondeterministic choices a policy is calculated. A policy is a conditional program where nondeterministic choices are substituted by the best alternative according to the background optimization theory. For each agent’s choice point the forward-search algorithm selects the best alternative, for each nature’s

choice point given by stochastic actions a conditional over the possible outcomes is introduced. The calculations of values and probabilities rely on the current situation term, i.e. the reward is given w.r.t. a particular situation. Therefore, the algorithm can decide optimal choices at choice points.

The idea for generating a plan library is now the following. In a run of the forward-search we do not calculate explicit numeric values for the reward function but keep it as terms. Basically, we store the whole computation tree for a respective input program. Later, when instantiating a plan from the plan library, we can establish the optimal policy, the values and probabilities of all outcomes of the policy. Thus, we make use of the trade-off between space and time. It turns out that with re-instantiating the abstract terms and re-evaluating the optimal choices one can save a significant amount of computation time compared with the on-line interpretation of a decision-theoretic program. For the implementation of the calculation of policies in an abstract way, we have to modify several *BestDo* predicates in such a way that choices are not taken but all possible continuation policies are calculated. When calculating these policies for a conditional **if** φ **then** a_1 **else** a_2 **endif**, for instance, we have to calculate both branches, where φ and $\neg\varphi$ holds. For space reasons we cannot give the complete specification. But as the implementation is quite straight-forward, we can omit the other predicate definitions giving only our modifications of *BestDo* for stochastic actions as an example.

As in the previous section, we first have to expand the outcomes of an stochastic action with the *choice* predicate. Note that unlike the similar definition in Sect. 2 we calculate the value in an abstract fashion. This is denoted by the formula $v = +(-(\text{reward}(s), \text{cost}(a, s)), v')$ in prefix notation.

$$\begin{aligned} \text{BestDoM}(a; p, s, h, \pi, v, pr) &\stackrel{\text{def}}{=} \\ \exists \pi', v'. \text{BestDoMAux}(\text{choice}'(a), a, p, s, h, \pi', v', pr) \wedge \\ \pi' = a; \text{senseEffect}(a); \pi' \wedge v = +(-(\text{reward}(s), \text{cost}(a, s)), v') \end{aligned}$$

As in the previous case for each stochastic outcome n_1, \dots, n_k we have to calculate the appropriate continuation policy. The difference to the similar predicate in Sect. 2 is that we do not check for action preconditions and do not distinguish if an action is possible or not. The reason for this change is that we cannot decide whether or not the respective outcome action is possible as we are not given a concrete situation where we could evaluate the predicate. This must be checked when executing the so calculated policy. Again, the value as well as the probability of success is calculated as a term depending on situation s , not as a concrete value.

$$\begin{aligned} \text{BestDoMAux}(\{n_1, \dots, n_k\}, a, p, s, h, \pi, v, pr) &\stackrel{\text{def}}{=} \\ \exists \pi', v', pr'. \text{BestDoMAux}(\{n_2, \dots, n_k\}, p, s, h, \pi', v', pr') \wedge \\ \exists \pi_1, v_1, pr_1. \text{BestDoM}(p, \text{Do}(n_1, s), h - 1, \pi_1, v_1, pr_1) \wedge \text{senseCond}(n_1, \varphi_1) \\ \pi = \mathbf{if} \varphi_1 \mathbf{then} \pi_1 \mathbf{else} \pi' \mathbf{endif} \wedge \\ v = +(v, \cdot(v_1, \text{prob}(n_1, a, s))) \wedge pr = +(pr', \cdot(pr_1, \text{prob}(n_1, a, s))) \end{aligned}$$

$$\begin{aligned}
\text{BestDoMAux}(\{n_k\}, a, \delta, s, h, \pi, v, pr) &\stackrel{\text{def}}{=} \\
&\text{senseCond}(n_k, \varphi_k) \wedge \exists \pi', v', pr'. \text{BestDoM}(\delta, \text{do}(n_k, s), h, \pi', v', pr') \wedge \\
&\pi = \varphi_k?; \pi' \wedge v = \cdot(v', \text{prob}(n_k, a, s)) \wedge pr = \cdot(pr', \text{prob}(n_k, a, s))
\end{aligned}$$

As we have stressed before, the difference of our *BestDoM* definitions w.r.t. the original *BestDo* is that (1) all possible continuation policies have to be calculated, and (2) the value as well as the probability of success are handed over as terms. Later, when the policy is instantiated in a concrete situation, we could evaluate the value and probability term to get the real numbers. Compared to planning, this simple evaluation of abstract values is able to bring a computational benefit. In DT planning the whole search tree has to be explored during runtime, numerical values have to be calculated and compared to choose the highest one. Abstract planning in contrast already provides the whole tree and an abstract value. Thus, here the on-line time consumption of DT planning can be reduced. To illustrate this again, we give the abstract value for the agent calculating one step of the (simplified) policy to leave room 1 from the start position “S” through the northern door (see Fig. [4](#)):

$$\begin{aligned}
v = &+(-(\text{reward}(\text{do}(\text{go_up}, s)), \text{cost}(\text{do}(\text{go_up}, s))), \cdot(\text{prob}(\text{go_up}, \text{det_up}, s), \\
&\text{reward}(\text{do}(\text{det_up}, s)), \cdot(\text{prob}(\text{go_up}, \text{noop}, s), \text{reward}(\text{do}(\text{noop}, s)))), \\
&+(-(\text{reward}(\text{do}(\text{go_right}, s)), \text{cost}(\text{do}(\text{go_right}, s))), \cdot(\text{prob}(\text{go_right}, \text{det_right}, s), \\
&\text{reward}(\text{do}(\text{det_right}, s)), \cdot(\text{prob}(\text{go_right}, \text{noop}, s), \text{reward}(\text{do}(\text{noop}, s)))), \dots
\end{aligned}$$

In our implementation in Prolog which is generated from the macro description the following policy results:

$$\begin{aligned}
(\text{poss}(\text{go_up}, S) \rightarrow ((\text{has_val}(\text{pos}, V_6, S), V_6 = [V_{11}, V_{12}]), V_{14} \text{ is } V_{12} + (1), \dots), \\
V_{111} = [\text{go_up}, \text{if}(\text{pos} = [V_{11}, V_{13}], [], [\text{if}(\text{pos} = [V_{11}, V_{12}], [], [])]); V_{111} = [], !), \\
(\text{poss}(\text{go_right}, S) \rightarrow ((\text{has_val}(\text{pos}, V_{19}, S), V_{19} = [V_{24}, V_{25}]), V_{27} \text{ is } V_{24} + (1), \dots), \\
V_{110} = [\text{go_right}, \text{if}(\text{pos} = [V_{26}, V_{25}], [], [\text{if}(\text{pos} = [V_{24}, V_{25}], [], [])]); V_{110} = [], !), \\
(\text{poss}(\text{go_down}, S) \rightarrow ((\text{has_val}(\text{pos}, V_{32}, S), V_{32} = [V_{37}, V_{38}]), V_{40} \text{ is } V_{38} - (1), \dots)
\end{aligned}$$

`has_val` is a predicate to evaluate a fluent, and `poss` checks whether the precondition of an action holds. For ease of presentation the example is simplified as the basic actions only have one failure case, namely a *noop* action where the agent will stay on the same position. As one can see from the small examples above the policies grow large even for small toy problems. Actually, if the optimal solution to leave room 1 is encoded as an option, i.e. as a macro-action, the option cannot be represented in a compact way as each possible outcome for each action has to be provided when calculating abstract policies and values. Though, as we will show in Sect. [5](#) one can buy the larger space consumption by less computation time.

4 Generating a DT Plan Library

In the previous section we presented our idea of solving decision-theoretic problems in an abstract way implemented by a new predicate *BestDoM*. Independent from a particular situation, this predicate generates, from a given program, a list of abstract policies with a corresponding abstract representation of the value function. In a given situation, this abstract representation can be easily re-instantiated with numerical values, which makes it possible to choose and execute the highest valued abstract policy. Our idea is now to use this abstract plan instead of DT planning and to build a DT PLAN LIBRARY, a library that contains policies that already were calculated and used before. These stored policies are then provided for re-use if the agent comes to similar states again.

This then is the basic idea of our new options approach: an option is essentially a READYLOG program with a solve statement which encodes the contents of the option (similar to the *navigate* program in Section 2). This means that with this program the partition of the state space of the option is induced. The states reachable with the program form the state space of the option. Note that we use the term state here and use it similar as is done in related action formalisms like FLUX [18]. A finite subset of instantiated fluents form the state representation for the option and must be given by the user. This state description must contain enough information to evaluate test and action preconditions mentioned in the input program from which the option is calculated. For the maze domain our state representation comprises only the location fluent. It is sufficient for navigating between the rooms in the maze. In the soccer domain, on the other hand, we need to find the fluents which are important for the option. For a free kick option, for instance, taking the positions of all 22 players on the field into account is too much. Only the players near the ball might be of importance. In the MDP theory this is referred to as factored representations of the state space.

To calculate a policy for an option the following steps have to be conducted:

1. *Off-line preprocessing*
 - (a) Calculate an abstract policy for each solve statement occurring in the behavior specification.
 - (b) Replace each solve statement with its abstract policy in the specification.
2. *On-line execution*
 - (a) look up the policy, value, and probability of success for the option in the DT PLAN LIBRARY.
 - (b) If the option is not contained in the library, instantiate the option in the particular situation and store the value and the probability of success together with the current world state in the library.

In the off-line part we pre-process each occurrence of a solve statement in our agent high-level specification with the *BestDoM* predicates shown in the previous section. As the result we obtain for each solve statement an abstract policy as presented before.

When executing an option in a particular situation we first query our DT PLAN LIBRARY if for the current world situation there exists an instantiated

```

getState;
while  $\varphi_m$  do
  if DT PLAN LIBRARY has entry for current state  $s$  then
    | get_bestPolicy( $s$ , DT PLAN LIBRARY,  $\pi$ );
    | execute( $\pi_s$ );
  else
    | evaluate( $s$ , AbstractValues,  $\pi_s$ );
    | execute( $\pi_s$ );
    | store( $(s, \pi_s, v, pr)$ , DT PLAN LIBRARY);
  end
  execute( $a_{sense}$ );
end

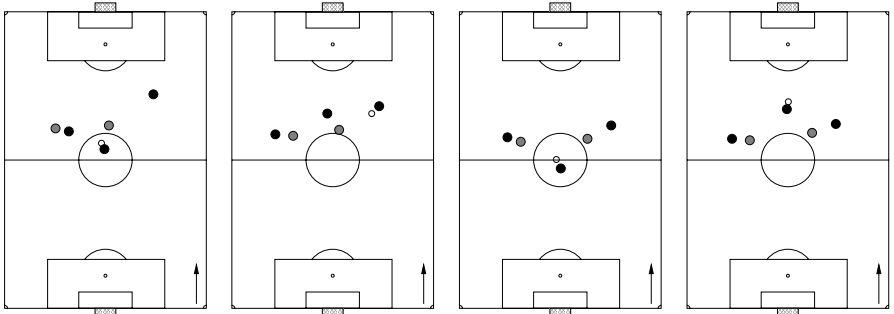
```

Algorithm 4.1. The algorithm executed by a macro-action

policy for the option currently to be executed. If so, we simply take this policy from the library and execute it. If there does not exist a policy for the option in the current world situation, we have to generate it. We take the situation independent abstract policy for the option and substitute the situation terms with the actual situation. Similarly, we evaluate the value and success probability of the option given the current world situation. With a particular situation we can re-evaluate the precondition axioms of actions, if-conditions, and nondeterministic choices of the abstract policy and obtain one fully instantiated policy which is the same as if we would have calculated it on the fly. To gain computation speed for the next time when the agent wants to execute the option in this particular situation we store the fully instantiated policy, the value and the success probability together with the world state. Thus, the next time the option is to be executed in the very same situation, we simply look up the policy without the need to calculate anything at all.

The on-line execution is again illustrated in Algorithm 4.1. The predicate `getState` calculates the current world state based on fluent values as described above. The predicate `get_bestPolicy` performs the lookup operation, the predicate `evaluate` evaluates the abstract plan tree returning a fully instantiated policy π_s , which is then executed with `execute(π)`. The `store` predicate saves the instantiated policy, the value, and the success probability together with the current world situation in the DT PLAN LIBRARY for the next time it is needed. The action a_{sense} is a sensing action which is executed to sense the actual state the agent is in when trying to execute the option. The logical formula φ_m is a condition which checks if the option is executable. This condition can be viewed as a precondition for the option. This precondition is part of the specification of the option and must be provided by the user.

The main difference to the original options approach in READYLOG [10] is that we do not use standard value iteration to calculate the behavior policy but make use of the forward-search algorithm which is also used for decision-theoretic planning in the framework. The advantage is that with our new approach we gain flexibility in the sense that the input program from which the option is calculated determines the state space of the option. Determining the sub-tasks in toy



(a) First setting for outplaying opponents. (b) The macro-action chooses to pass and go. (c) Second setting for outplaying opponents. (d) The macro-action chooses to dribble.

Fig. 3. Outplay Opponent; (a)-(b): “pass and go”.(c)-(d): “dribble”

domains like the maze domain is rather easy (leaving room through doors). But in realistic domains like robotic soccer this task is not that easy and therefore it helps to define the option by the input program.

5 Experimental Results in the RoboCup Domain

In this section we present the results we obtained by testing our approach in a continuous real-time domain. The presented results are from the 2D Soccer Simulation league where two teams of 11 soccer agents compete against each other. The *Soccer Server* [19] provides the simulation environment where each agent can submit basic commands like *kick*, *dash*, or *turn* and receives a world model in form of visible landmarks like the goals or the corner posts.

We defined two macro-actions here to show the applicability of our approach in such a demanding environment. For restricting the state space of the soccer domain we make use of a qualitative world model which abstracts from the infinite quantitative state space. In [3] Schiffer et al. proposed a qualitative world model for this soccer domain. The playing field is divided into grid cells, where each cell of this grid clusters infinite many coordinates. For each of these cells one quantitative representative (the center of this cell) is provided. Such a cell is then represented by just one contained position. So instead of the ‘real’ positions, we use these representatives as state variables. This leads to a discretization of the state space and makes it possible for the macro-action to ‘recognize’ a state again. Without this abstraction we would preclude the re-use of policies stored in the DT PLAN LIBRARY as it is very improbable that an agent reaches, say, the position (12.0226, 5.8475) on the pitch twice. Note, however, that this abstraction alone is still not enough to make the options as defined in [10] applicable in this domain.

The first macro-action is designed to *outplay opponents* as shown in Fig. 3(a) and Fig. 3(b). Facing attacking opponents, the ball leading agent either dribbles or passes the ball to a teammate. If the macro-action chooses the pass, the agent afterwards moves to a free position to be a pass receiver again. The second action aims to *create a good scoring opportunity* to shoot a goal. The agent in ball possession can dribble with the ball if the distance to the opponent's goal is too far. Near the goal the agent can shoot directly to the goal or pass to a teammate that is in a better scoring position. We compared macro-actions with DT planning for the game settings we created as test scenarios. Besides the computational behavior we also investigated the action's choices in specific states. We want to remark that by using the proposed state abstraction we commit to some sort of bias in the representation of the environment that is reflected in the state variables. In fact, these variables have to ensure that the macro-action correctly chooses a policy fitting the game setting. Situations which require different policies have to be represented by different states. This can be handled by the granularity of the qualitative world model. The provided grid is adjustable by hand and can be created fine-grained enough to distinguish significant changes in the game setting. In the first setting depicted in Fig. 3(a) the ball leading agent directly faces an opponent. In this state the macro-action evaluates a pass to the uncovered teammate and a subsequent move action as best policy (Fig. 3(b)). In the second setting given in Fig. 3(c) both opponents cut the possible pass-ways for the ball leading agent to its teammates. The state representation 'detects' this difference and evaluates a new policy. The agent dribbles towards the opponent's goal (Fig. 3(d)). Qualitatively there is no difference in the behavior between the on the fly DT planning and the macro approach as both rely on the same READYLOG input programs. What can be observed, though, is that the macro-action approach needs less time to come to a decision.

We considered three strategies: (a) using DT planning to cope with the task, (b) using the macro-action, but only by evaluating a policy in each step², and (c) using the macro-action with the DT PLAN LIBRARY that was generated in the last step. We conducted 20 iterations per setting. Using the planning approach the agent needed 0.1 seconds on average to calculate a policy. With the evaluation strategy (b) only 0.8 seconds are needed. This is a speed-up compared to planning of about 20 %. The time for off-line computations in this example was about 0.02 seconds for each macro. Even taking this preprocessing time into account our macro approach yields reasonable speed-ups. Of course, preprocessing more and more complex macro-actions consumes more off-line computation time. But as this time does not need to be spent on-line this off-line computation time can be justified. The macro-action based on the DT PLAN LIBRARY clearly outperforms DT planning. In each test-run, for both macro-actions, the executing system constantly returns the minimum of measurable time of 0.01 seconds for searching the best plan in the DT PLAN LIBRARY. In

² Each policy evaluated in this step is stored in the DT PLAN LIBRARY, so we can use this stored knowledge in the next step (c).

fact, this is a mean time saving of over 90 %. In tests in the ROBOCUP 3D simulations which we conducted in another test scenario these savings showed an impact on the quality of the agent’s behavior in real game situations. Caused by a reduced reaction time the agent showed fewer losses of the ball during the game. Moreover the team created more opportunities to score than in test runs without using the macro-actions. This goes along with a space consumption of about 10 kB for each defined macro-action. Our examples reflect the task in the ROBOCUP for finding a suitable policy in a split second. In practice, our policies are quite short, since computing larger policies is not (yet) reasonable as the world changes unpredictably for larger horizons. In our application examples the larger space consumption does not play a role. For more complex macros the space consumption of the exponentially growing computation tree has to be further investigated. It is due to future work to examine how well our method scales.

6 Conclusion

In this paper we showed how a significant speed-up can be gained for calculating policies in the logic-based framework READYLOG. The basic idea is to calculate and store all possible calculation branches of the used forward-search algorithm together with their values and probabilities in an abstract fashion. When the agent faces a world situation which it already encountered before it can simply draw on a previously calculated policy, re-instantiate it with the current situation it is in and gains an optimal policy instantaneously. While previous proposed approaches like options in READYLOG also resulted in an exponential speed-up [10] for toy domains they fail for continuous domains like robotic soccer. The reason is that the method presented in [10] relies on an explicit state enumeration for calculating the result of an option. Even when soccer state space abstractions [3] are used the problem of deciding the states of an option remains. With our approach of a plan library on the other hand, this problem is no longer apparent. Which states belong to an option is implicitly given by the input READYLOG program. Nevertheless, the options approach from [10] can be easily modeled equivalently with our plan library. The appeal of the plan library lies for one in the speed-up of agent’s decision making, for another in its simplicity. By simply exploiting the well-known trade-off between space and time we can speed-up decision-making in READYLOG significantly.

Acknowledgments

This work was supported by the German National Science Foundation (DFG) in the Priority Program 1125, *Cooperating Teams of Mobile Robots in Dynamic Environments*. We would like to thank the anonymous reviewers for their helpful comments.

References

1. Ferrein, A., Fritz, C., Lakemeyer, G.: Using golog for deliberation and team coordination in robotic soccer. *KI Künstliche Intelligenz* (1) (2005)
2. Ferrein, A., Fritz, C., Lakemeyer, G.: On-line Decision-Theoretic Golog for Unpredictable Domains. In: *Proc. KI-04* (2004)
3. Schiffer, S., Ferrein, A., Lakemeyer, G.: Qualitative world models for soccer robots. In: *Qualitative Constraint Calculi Workshop at KI-06* (2006)
4. Schiffer, S., Ferrein, A., Lakemeyer, G.: Football is coming home. In: Chen, X., Liu, W., Williams, M.-A. (eds.) *Proceedings of the International PCAR Symposium* (2006)
5. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: Decision-Theoretic, High-Level Agent Programming in the Situation Calculus. In: *Proc. AAAI-00* (2000)
6. Puterman, M.: *Markov Decision Processes: Discrete Dynamic Programming*. Wiley, New York, USA (1994)
7. McCarthy, J.: *Situations, Actions and Causal Laws*. Technical report, Stanford University (1963)
8. Reiter, R.: *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge (2001)
9. Soutchanski, M.: *High-Level Robot Programming in Dynamic and Incompletely Known Environments*. PhD thesis, University of Toronto, Toronto, Canada (2003)
10. Ferrein, A., Fritz, C., Lakemeyer, G.: Extending DTGolog with Options. In: *Proc. IJCAI'03* (2003)
11. Levesque, H.J., Reiter, R., Lesperance, Y., Lin, F., Scherl, R.B.: GOLOG: A logic programming language for dynamic domains. *J. of Log. Progr.* 31(1-3) (1997)
12. Grosskreutz, H., Lakemeyer, G.: cc-Golog – An Action Language with Continuous Change. *Logic Journal of the IGPL* (2002)
13. De Giacomo, G., Lesperance, Y., Levesque, H.J.: ConGolog, A concurrent programming language based on situation calculus. *Artificial Intelligence* 121(1–2), 109–169 (2000)
14. De Giacomo, G., Levesque, H.: An incremental interpreter for high-level programs with sensing. In: Levesque, H.J., Pirri, F. (eds.) *Logical foundation for cognitive agents: contributions in honor of Ray Reiter*, pp. 86–102. Springer, Berlin (1999)
15. Grosskreutz, H.: Probabilistic projection and belief update in the pgolog framework. In: *Proc. CogRob-00 at ECAI-00* (2000)
16. Hauskrecht, M., Meuleau, N., Kaelbling, L.P., Dean, T., Boutilier, C.: Hierarchical Solution of Markov Decision Processes using Macro-actions. In: *Proc. UAI* (1998)
17. Sutton, R.S., Precup, D., Singh, S.P.: Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence* 112(1-2), 181–211 (1999)
18. Thielscher, M.: FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming* 5(4-5), 533–565 (2005)
19. Noda, I., Matsubara, H., Hiraki, K., Frank, I.: Soccer server: a tool for research on multi-agent systems. *Applied Artificial Intelligence* 12 (1998)

On the Construction and Evaluation of Flexible Plan-Refinement Strategies

Bernd Schattenberg, Julien Bidot*, and Susanne Biundo

Institute for Artificial Intelligence
Ulm University, Germany
firstname.lastname@uni-ulm.de

Abstract. This paper describes a system for the systematic construction and evaluation of planning strategies. It is based on a proper formal account of refinement planning and allows to decouple plan-deficiency detection, refinement computation, and search control. In adopting this methodology, planning strategies can be explicitly described and easily deployed in various system configurations.

We introduce novel domain-independent planning strategies that are applicable to a wide range of planning capabilities and methods. These so-called *HotSpot* strategies are guided by information about current plan defects and solution options. The results of a first empirical performance evaluation are presented in the context of hybrid planning.

1 Introduction

In hybrid planning – the combination of hierarchical-task-network (HTN) planning with partial-order causal link (POCL) techniques – the solution of planning problems requires the integration of plan synthesis based on primitive operations with the use of predefined plans that implement abstract actions. As a consequence, causal reasoning and task expansion interleave during the plan-development process.

So far, only few research has been devoted to search strategies in hybrid planning; nor have existing strategies ever been systematically compared and assessed in this context. For most of today’s planning systems, this is due to the fact that search strategies are only implicitly defined and indirectly controlled via parameters. In these frameworks, search strategies and planning algorithms are strongly interdependent and changing one of them inevitably affects the other.

Furthermore, all these search strategies are basically fixed; i.e., in each plan-generation cycle, they follow a pre-defined schema indicating at which kind of deficiency to look first, and which changes to apply to the plan preferably. In POCL planning, Joslin & Pollack proposed Least-Cost Flaw Repair, a method that repairs first the flaws associated with the minimal number of modifications [1]. Peot & Smith presented strategies that consist of delaying the treatment

* This work has been supported by a grant from the Ministry of Science, Research and the Arts of Baden-Württemberg (Az: 23-7532.24-14-19).

of causal threats for which multiple solutions exist [2]. Their results have been picked up by Schubert & Gerevini who proposed simple but effective plan metrics for an A^* heuristic [3]. In HTN planning, Tsuneto et al. described the *fewest-alternatives first* heuristic for selecting task expansions in the UMCP system [4]. McCluskey introduced the *Expand then Make Sound* method [5] in which the expansion of an abstract task is followed by repairing the plan’s causal structure.

For studying search strategies, we use the hybrid planning approach presented in Schattenberg et al. [6]. It provides a formal framework in which the plan generation process is decomposed into well-defined functions for flaw detection and the generation of plan modifications, respectively. Furthermore, the approach decouples search strategies from planning algorithms. The strategies are quite flexible and can be applied to large varieties of system configurations. Recently, they have been integrated into a planning-and-scheduling system [7].

2 Refinement-Based Planning

We employ a hybrid planning formalism based on sorted first-order logic [6].

An operator or primitive *task schema* $t(\bar{\tau}) = (\text{prec}(t(\bar{\tau})), \text{add}(t(\bar{\tau})), \text{del}(t(\bar{\tau})))$ specifies the *preconditions* and the *positive* and *negative effects* of the task. Preconditions and effects are sets of literals, $\bar{\tau} = \tau_1, \dots, \tau_n$ are the task parameters. A ground instance of a task schema is called an *operation*. A *state* is a finite set of ground atoms, and an operation $t(\bar{c})$ is called *applicable* in a state s , if the positive literals of $\text{prec}(t(\bar{c}))$ are in s and the negative are not. The result of applying the operation in a state s is a state $s' = (s \cup \text{add}(t(\bar{c}))) \setminus \text{del}(t(\bar{c}))$. The applicability of sequences of operations is defined inductively over state sequences as usual.

Abstract actions are represented by *complex tasks*. In hybrid planning these show preconditions and effects exactly like the primitive tasks. The associated state transition semantics is based on axiomatic state refinements that relate task preconditions and effects across various abstraction levels [8].

A *partial plan* or *task network* is a tuple $P = (TE, \prec, VC, CL)$ with the following sets of *plan components*: TE is a set of plan steps or *task expressions* $te = l : t(\bar{\tau})$ where l is unique label and $t(\bar{\tau})$ is a (partially) instantiated task. \prec is a set of *ordering constraints* that impose a partial order on the plan steps in TE . VC is a set of *variable constraints* $v \doteq \tau$ and $v \not\doteq \tau$, which *codesignate* and *non-codesignate* variables occurring in TE with variables or constants. Furthermore, it contains *sort restrictions* of the form $v \dot{\in} Z$ and $v \not\dot{\in} Z$, where Z is a sort symbol, that restrict further codesignations. CL is a set of *causal links* $\langle te_i, \phi, te_j \rangle$, indicating that a task te_i *establishes* a precondition of task te_j . Causal links are used in the usual sense as a book-keeping means for maintaining the causal structure.

Task networks are also used as pre-defined *implementations* of complex tasks. An *expansion method* $em = (t(\bar{\tau}), (TE_{em}, \prec_{em}, VC_{em}, CL_{em}))$ relates such a complex task schema $t(\bar{\tau})$ to a task network.

A domain model for hybrid planning is specified by $D = (\mathbf{T}, \mathbf{EM})$, that is a set of task schemata \mathbf{T} and a set \mathbf{EM} of expansion methods for implementing

the complex tasks in T . Please note that there are in general multiple methods provided for each complex task schema.

A *hybrid planning problem* is the structure $\pi = (D, s_{init}, s_{goal}, P_{init})$. It consists of a domain model D and an initial task network P_{init} . A partial plan $P = (TE, \prec, VC, CL)$ is a solution to a problem specification, if and only if TE consists of primitive task expressions only, and P is executable in s_{init} and generates a state s_{end} such that $s_{goal} \subseteq s_{end}$ holds. Plan P is thereby called *executable* in a state s and *generates* a state s' , if all ground linearizations of P , that means all linearizations of all ground instances of the task expressions in TE that are compatible with \prec and VC , are executable in s and generate a state $s'' \supseteq s'$.

In our framework, violations of the solution criteria are made explicit by so-called *flaws*, data structures that represent critical components of a partial plan, i.e. those that prevent the plan from being a solution. The flaws serve to classify sub-problems and to guide the search for a solution.

Definition 1 (Flaw). *Given a problem specification π and a partial plan P that is not a solution to π , a flaw $\mathbf{f} = \{\alpha_1, \dots, \alpha_n\}$ consists of a set of plan components α_i of P . It represents a defect in which these components are involved.*

The set of all flaws is denoted by F and subsets F_x represent classes of flaws. For a partial plan $P = (TE, \prec, VC, CL)$ the violation of the solution criteria is mirrored by flaw classes such as the following. The class $F_{\text{CausalThreat}}$, e.g., contains flaws $\mathbf{f} = \{(te_i, \phi, te_j), te_k\}$ describing causal threats, i.e., indicating that a task te_k is possibly being ordered between plan steps i and j ($te_k \not\prec^* te_i$ and $te_j \not\prec^* te_k$) and there exists a variable substitution σ that is consistent with the equations and in-equations imposed by the variable constraints in VC such that $\sigma(\phi) \in \sigma(\text{del}(te_k))$ for positive literals ϕ and $\sigma(|\phi|) \in \sigma(\text{add}(te_k))$ for negative literals. This means, the presence of te_k in P as it stands will possibly corrupt the executability of at least some ground linearizations of P .

Flaw classes also cover the presence of abstract actions in the plan, ordering and variable constraint inconsistencies, unsupported preconditions of actions, etc. The complete class definitions can be found in [6]. It can be shown that these flaw definitions are complete in the sense that for any given planning problem π and plan P that is not flawed, P is a solution to π .

For the definition of the refinement operators, we make use of an explicit representation of change to the plan data structure: plan modifications. This representation has two major impacts on the hybrid planning framework: First, it makes changes to the plan analyzable and with that the available options for a search strategy become comparable qualitatively and quantitatively. Second, new functionality can instantly be integrated as soon as it is translated into the plan modification structure.

Definition 2 (Plan Modification). *For a given partial plan P and domain model D , a plan modification is defined as the structure $\mathbf{m} = (E^+, E^-)$. E^+ and E^- are sets of elementary additions and deletions of plan components over P and D .*

E^\oplus and E^\ominus are assumed to be disjoint and $E^\oplus \cup E^\ominus \neq \emptyset$. Furthermore, we require plan modifications to be consistent in the sense that all elements in the deletion set are components of plan P and all elements in the addition set are new components.

The set of all plan modifications is denoted by \mathbb{M} and grouped into modification classes \mathbb{M}_y . The application of plan modifications is characterized by the generic plan transformation function $app : \mathbb{M} \times \mathbb{P} \rightarrow \mathbb{P}$, which takes a plan modification $\mathbf{m} = (E^\oplus, E^\ominus)$ and a plan P , and returns a plan P' that is obtained from P by adding all elements of E^\oplus and removing those in E^\ominus . As an example, the class $\mathbb{M}_{\text{AddCL}}$ contains plan modifications $\mathbf{m} = (\{\langle te_i, \phi, te_j \rangle, v_1 \doteq \tau_1, \dots, v_k \doteq \tau_k\}^\oplus, \{\}^\ominus)$ for manipulating a given partial plan $P = (TE, \prec, VC, CL)$ by adding causal links. The plan steps te_i and te_j are in such a modification in TE and the codesignations represent variable substitutions. They induce a VC' -compatible substitution σ' with $VC' = VC \cup \{v_1 \doteq \tau_1, \dots, v_k \doteq \tau_k\}$ such that $\sigma'(\phi) \in \sigma'(\text{add}(te_i))$ for positive literals ϕ , $\sigma'(|\phi|) \in \sigma'(\text{del}(te_i))$ for negative literals, and $\sigma'(\phi) \in \sigma'(\text{prec}(te_j))$.

The complete collection of refinement operations for hybrid planning is introduced in [6]. This also covers the expansion of abstract plan steps and the insertion of new plan steps, ordering constraints, and variable (in-) equations. These plan modifications are the canonical plan transformation generators in a refinement-based hybrid planner: starting from an initial task network, the current plan is checked against the solution criteria, and while these are not met, refinements are applied. If no applicable modification exists, backtracking is performed. In order to make the search systematic and efficient, the algorithm should focus on those modification steps which are appropriate to overcome the deficiencies in the current plan. Based on the formal notions of plan modifications and flaws, a generic algorithm and planning strategies can be defined. A strategy specifies *how* and *which* flaws in a partial plan are eliminated through appropriate plan modification steps. We therefore need to define the conditions under which a plan modification can *in principle* eliminate a given flaw.

A class of plan modifications $\mathbb{M}_y \subseteq \mathbb{M}$ is called *appropriate* for a class of flaws $\mathbb{F}_x \subseteq \mathbb{F}$, if and only if there exist partial plans P which contain flaws $\mathbf{f} \in \mathbb{F}_x$ and modifications $\mathbf{m} \in \mathbb{M}_y$, such that the refined plans $P' = app(\mathbf{m}, P)$ do not contain these flaws anymore.

It is easy to see that the plan modifications perform a strict refinement, that means, a subsequent application of any modification instances cannot result in the same plan twice; the plan development is inherently acyclic. Given that, any flaw instance cannot be re-introduced once it has been eliminated. This qualifies the appropriateness relation as a valid strategic advice for the plan generation process and motivates its use as the trigger function for plan modifications: the α modification triggering function relates flaw classes with their potentially solving modification classes. As an example, causal threat flaws can be solved by expanding abstract actions which are involved in the threat (by overlapping task implementations), by promotion or demotion, or by separating variables through inequality constraints [8]: $\alpha(\mathbb{F}_{\text{CausalThreat}}) = \mathbb{M}_{\text{ExpTask}} \cup \mathbb{M}_{\text{AddOrdCstr}} \cup \mathbb{M}_{\text{AddVarCstr}}$.

Please note, that α states nothing about the relationship between the actual flaw and modification instances.

The modification triggering function allows for a simple plan generation schema: based on the flaws that are found in the current plan, the calculation of plan modifications is motivated. A strategy component can then freely choose among all applicable plan modifications in order to try to solve one of the particular problems that have been identified. If any flaw remains unanswered by triggered modification generating entities, no appropriate refinement candidate exists and the plan can be discarded. There are even flaw classes that do not trigger any modification. Flaw instances that represent ordering cycles and variable inconsistencies obviously cannot be resolved by our refinement operators and do therefore not trigger any modification: $\alpha(\mathbf{F}_{\text{OrdInconst}}) = \alpha(\mathbf{F}_{\text{VarInconst}}) = \emptyset$.

The above mentioned triggering function completely separates the computation of flaws from the computation of modifications, and in turn both computations are independent from search-related considerations. The system architecture relies on this separation and exploits it in two ways: module invocation and interplay are specified through the α -trigger, while reasoning about search can be performed on the basis of flaws and modifications without taking their actual computation into account. Hence, we map flaw and modification classes directly onto groups of modules which are responsible for their computation.

Definition 3 (Detection Modules). *A detection module x is a function that, given a partial plan P , a domain model D , and a problem specification π , returns all flaws of type x that are present in the plan: $f_x^{\text{det}} : \mathbf{P} \times \mathbf{D} \times \Pi \rightarrow 2^{\mathbf{F}_x}$.*

Definition 4 (Modification Modules). *A modification module y is a function which computes all plan modifications of type y that are applicable to a given plan P and that are appropriate for given flaws with respect to a given domain model: $f_y^{\text{mod}} : \mathbf{P} \times 2^{\mathbf{F}_x} \times \mathbf{D} \rightarrow 2^{\mathbf{M}_y}$ for $\mathbf{M}_y \subseteq \alpha(\mathbf{F}_x)$.*

Please note that plan modifications carry a reference to the flaw instance they address, i.e., any plan modification is unambiguously linked with its triggering flaw.

While the plan deficiency detectors and the refinement generators provide the basic plan generation functionality, strategy functions can be designed for reasoning about which paths in the refinement space to pursue. To this end, we split up reasoning about search into two compartments: The first component is an option evaluation that is performed in the local view of the currently processed plan; it reasons about the detected flaws and proposed refinements in the current plan and assesses the modifications. The second component is responsible for the global view on the refinement space and evaluates the alternative search options.

We begin with the definition of a strategic function that selects all plan modifications that are considered to be worthwhile options, thereby determining the ordered set of successors for the current plan in the plan refinement space. In doing so, the following function also determines the branching behavior of the upcoming refinement-based planning algorithm.

Definition 5 (Modification-Selection Module). *Given a plan, a set of flaws, and a set of plan modifications, a modification-selection module is a function $f^{modSel} : \mathcal{P} \times 2^{\mathcal{F}} \times 2^{\mathcal{M}} \rightarrow 2^{\mathcal{M} \times \mathcal{M}}$ that selects some (or all) of the plan modifications and returns them in a partial order for application to the passed plan.*

Strategies discard a plan P , if any flaw remains unaddressed by the associated modification modules. That means, we reject any plan P , over any domain model D and for any planning problem π , if for any f_x^{det} and $f_{y_1}^{mod}, \dots, f_{y_n}^{mod}$ with $\mathcal{M}_{y_1} \cup \dots \cup \mathcal{M}_{y_n} = \alpha(\mathcal{F}_x)$ the following holds: $\bigcup_{1 \leq i \leq n} f_{y_i}^{mod}(P, D, f_x^{det}(P, D, \pi)) = \emptyset$.

The second aspect of search control concerns the selection of those plans that are to be processed next by the detection and modification modules. These unassessed partial plans, the leaves of the search tree, are usually called the *fringe*. In other words, concrete implementations of the following module are responsible for the general search schema, ranging from uninformed procedures such as depth-first, breadth-first, etc., to informed, heuristic schemata.

Definition 6 (Plan-Selection Module). *A plan-selection module is a function that returns a partial order of plans for a given sequence of plans. It is described as $f^{planSel} : \mathcal{P}^* \rightarrow 2^{\mathcal{P} \times \mathcal{P}}$.*

Building on the components defined so far we can now assemble a hybrid planning system. A software artifact that implements the generic refinement algorithm (Alg. [1](#)) is making the flaw detection and modification generating modules operational by stepwise collecting plan deficiencies, collecting appropriate modifications, selecting worthwhile modifications, and finally selecting the next plan in the fringe of the search tree. Please note that the algorithm is formulated independently from the deployed modules, since the options to address existing flaws by appropriate plan modifications is defined via α .

3 Search Strategies

A large number of search strategies can be realized in the proposed refinement-planning framework by sequencing the respective selection modules and using the returned partially ordered sets of modifications, respectively plans, to modulate preceding decisions: if the primary strategy does not prefer one option over the other, the secondary strategy is followed, and so on, until finally a random preference is assumed. The following single-objective heuristics constitute the building blocks for more sophisticated strategy compositions.

Unflexible Strategies: Previous work has presented the translation of some existing search strategies into our hybrid planning framework and showed that most of them are unflexible in the sense that they represent a fixed preference schema on the flaw type they want to get eliminated primarily and then select appropriate modification methods. For example, it is very common to care for the plan to become primitive first and then to deal with causal interactions.

Algorithm 1. The generic refinement planning algorithm.

Require: Sets of modules $\mathfrak{Det} = \{f_1^{det}, \dots, f_d^{det}\}$ and $\mathfrak{Mod} = \{f_1^{mod}, \dots, f_m^{mod}\}$
Require: Selection modules f^{modSel} and $f^{planSel}$

```

plan( $P_1 \dots P_n, D, \pi$ ): { $P_1$  is the plan that is worked on}
if  $n = 0$  then
3:   return failure
    $P \leftarrow P_1$ ;   Fringe  $\leftarrow P_2 \dots P_n$ 
    $F \leftarrow \emptyset$ 
6:   for all  $f_x^{det} \in \mathfrak{Det}$  do {Flaw detection}
      $F \leftarrow F \cup f_x^{det}(P, D, \pi)$ 
     if  $F = \emptyset$  then
9:       return  $P$ 
        $M \leftarrow \emptyset$ 
       for  $x = 1$  to  $d$  do {Modification generation}
12:     $F_x = F \cap \mathbf{F}_x$  {Process flaws class-wise as returned by corresponding  $f_x^{det}$ }
       for all  $\mathbf{f} \in F_x$  do
         for all  $f_y^{mod} \in \mathfrak{Mod}$  with  $\mathbf{M}_y \subseteq \alpha(\mathbf{F}_x)$  do
15:           $M \leftarrow M \cup f_y^{mod}(P, \mathbf{f}, D)$ 
           if  $\mathbf{f}$  was un-addressed then
              $P_{next} \leftarrow f^{planSel}(\text{Fringe})$ 
18:          return  $\text{plan}(P_{next} \circ (\text{Fringe} \setminus P_{next}), D, \pi)$ 
           for all  $\mathbf{m}$  in  $\text{linearize}(f^{modSel}(P, F, M))$  do {Strategic choices}
             Fringe  $\leftarrow \text{app}(\mathbf{m}, P) \circ \text{Fringe}$ 
21:    $P_{next} \leftarrow \text{first}(\text{linearize}(f^{planSel}(\text{Fringe})))$ 
       return  $\text{plan}(P_{next} \circ (\text{Fringe} \setminus P_{next}), D, \pi)$ 

```

We will not recapitulate the results obtained in [6], but rather give a short, systematic overview over the possible instances of unflexible selection schemas for modification and plan selection.

A traditional form of modification selection is either to prefer or to disfavor categorically specific classes of plan modifications; e.g., we prefer the expansion of tasks to task insertions or we try to avoid an assignment of variables to constants “as long as possible.” Systems with built-in strategies of this kind are typically provided with hand-tailored heuristics such that certain plan properties modulate the preference schema to some extent, but for now we focus on purely modification-based forms of decisions.

In the presented framework, the preference of a modification class is encoded as follows: Let \mathbf{M}_{Pref} be the preferred modification class by modification-selection module $f_{\text{Pref}MC}^{modSel}$, then the corresponding function is defined as

$$\mathbf{m}_i < \mathbf{m}_j \in f_{\text{Pref}MC}^{modSel}(P, \{\mathbf{f}_1, \dots, \mathbf{f}_m\}, \{\mathbf{m}_1, \dots, \mathbf{m}_n\}) \text{ if } \mathbf{m}_i \in \mathbf{M}_{\text{Pref}} \text{ and } \mathbf{m}_j \notin \mathbf{M}_{\text{Pref}}$$

for all plans $P \in \mathbf{P}$, sets of flaws $\mathbf{f}_1, \dots, \mathbf{f}_m \in \mathbf{F}$, and sets of plan modifications $\mathbf{m}_1, \dots, \mathbf{m}_n \in \mathbf{M}$. Avoiding the selection (and consequently the application) of instances of particular modification classes is realized by an analogue selection module that is returning the inverse of the above definition: $f_{\text{Dis}fMC}^{modSel}$.

Plan selection can also be based upon the availability of refinement options for a plan. For simplification, the presented algorithm performs three consecutive sections: one loop for the flaw detection, one loop for the modification generation, and finally the strategic choices. In the little more complicated implementation of the framework, after applying the selected plan modifications to the current plan (line 20), for every new plan in the fringe, flaws and modifications are calculated in advance, so that the following two plan-selection modules can be defined for the preference of a modification class:

$$P_i < P_j \in \mathcal{J}_{PrefMC}^{planSel}(\{P_1, \dots, P_n\}) \text{ if } |mods(P_i) \cap M_{Pref}| > |mods(P_j) \cap M_{Pref}|$$

The function $mods : P \rightarrow 2^M$ thereby returns all modifications that have been published for a given plan. This plan selection can be formulated in relation to the plan size; e.g., it is related to the number of plan steps:

$$P_i < P_j \in \mathcal{J}_{PrefMRatio}^{planSel}(\{P_1, \dots, P_n\}) \text{ if } \frac{|mods(P_i) \cap M_{Pref}|}{|TE_i|} > \frac{|mods(P_j) \cap M_{Pref}|}{|TE_j|}$$

The plan selection can also be defined inversely, thereby evading partial plans for which undesired refinement options are available.

Strategies that concentrate on the flaw situation rather than on the available refinement options are more “problem-oriented” and are typical for agenda-based planning algorithms, which collect the plan defects and then decide which one to tackle first. Let F_{Pref} be the preferred class of flaws and let $modFor : F \times P \rightarrow 2^M$ be the function that returns all modifications that have been answered to a given flaw in a given plan. A modification strategy module for preferring F_{Pref} is then defined as:

$$\begin{aligned} m_i < m_j \in \mathcal{J}_{AddrFC}^{modSel}(P, \{f_1, \dots, f_m\}, \{m_1, \dots, m_n\}) \\ \text{if } 1 \leq x_i, x_j \leq m, 1 \leq i, j \leq n, m_i \in modFor(f_{x_i}, P), m_j \in modFor(f_{x_j}, P) \\ \text{and } f_{x_i} \in F_{Pref}, f_{x_j} \notin F_{Pref} \end{aligned}$$

Like for the modification-dependent strategies, a flaw avoiding modification-selection module can be set up, and also corresponding absolute and relative plan selections are available.

Flexible Modification-Selection Strategies: *Flexible* strategies are capable of operating on a more general level by exploiting flaw/modification information: they are neither *flaw-dependent* as they do not primarily rely on a flaw type preference schema, nor *modification-dependent* as they do not have to be biased in favor of specific modification types. A representative is the modification-selection strategy *Least Committing First (LCF)*. It works according to the least-commitment principle and has proven to be very successful in the context of hybrid planning [6]. It is a generalized variant of *Least-Cost Flaw Repair* [1] and selects those modifications that address flaws for which the smallest number of alternative modifications has been found:

$$\begin{aligned}
 & \mathfrak{m}_i < \mathfrak{m}_j \in f_{LCF}^{modSel}(P, \{\mathbf{f}_1, \dots, \mathbf{f}_m\}, \{\mathfrak{m}_1, \dots, \mathfrak{m}_n\}) \\
 & \text{if } 1 \leq x_i, x_j \leq m, 1 \leq i, j \leq n, \mathfrak{m}_i \in \text{modFor}(\mathbf{f}_{x_i}, P), \mathfrak{m}_j \in \text{modFor}(\mathbf{f}_{x_j}, P) \\
 & \text{and } |\text{modFor}(\mathbf{f}_{x_i}, P)| < |\text{modFor}(\mathbf{f}_{x_j}, P)|
 \end{aligned}$$

It can easily be seen that this is a *flexible* strategy, since it does not depend on the actual types of issued flaws and modifications: it just compares answer set sizes in order to keep the branching in the search space low.

The so-called *HotSpot-based modification-selection* strategies have their origin in the observation that the least-commitment principle is in practice a very efficient search schema, although it does not take the actual structure of the compared options into account. It is, for example, not very intuitive to say that choosing one of two large task expansions represents less commitment than choosing one of three variable codesignations. The expansion obviously makes considerably more changes to the plan structure, and we expect that the “amount of change” has side effects on the overall flaw and modification availability in the future. Lifting the least-commitment schema on the plan structure level led to the development of the HotSpot notion, that is plan components that are referred to by multiple flaws and modifications [6]. We are now going to extend this strategy family by new modification-selection modules and will later introduce plan selection that is based on HotSpot calculations.

We begin with flaw-based modification selections, that means plan modifications are chosen based on the number of cross-references the addressed flaw has with other flaws. The simplest form of a HotSpot is the number of references to shared plan components. The following module avoids modifications that address flaws that contain frequently referenced components:

$$\begin{aligned}
 & \mathfrak{m}_i < \mathfrak{m}_j \in f_{DirUniHS}^{modSel}(P, F, M) \\
 & \text{if } \mathbf{f}_{x_i}, \mathbf{f}_{x_j} \in F, \mathfrak{m}_i \in M \cap \text{modFor}(\mathbf{f}_{x_i}, P), \mathfrak{m}_j \in M \cap \text{modFor}(\mathbf{f}_{x_j}, P) \\
 & \text{and } \sum_{\mathbf{f} \in F \setminus \mathbf{f}_{x_i}} |\mathbf{f} \cap \mathbf{f}_{x_i}| < \sum_{\mathbf{f} \in F \setminus \mathbf{f}_{x_j}} |\mathbf{f} \cap \mathbf{f}_{x_j}|
 \end{aligned}$$

Direct HotSpot calculations offer a first insight into inter-flaw relationships but do not capture relatively trivial indirect dependencies. We therefore introduce the notion of plan *sub-components* which are the structures that constitute plan components. E.g., a task expression is a component of a plan, but may also be a sub-component of an ordering constraint. In order to represent HotSpot elements on the sub-component level, the modification selection has to be adapted as follows: Given a function *comp* that returns all sub-components of a set of plan components, we define the indirect uniform HotSpot selection as the function:

$$\begin{aligned}
 & \mathfrak{m}_i < \mathfrak{m}_j \in f_{IndUniHS}^{modSel}(P, F, M) \\
 & \text{if } \mathbf{f}_{x_i}, \mathbf{f}_{x_j} \in F, \mathfrak{m}_i \in M \cap \text{modFor}(\mathbf{f}_{x_i}, P), \mathfrak{m}_j \in M \cap \text{modFor}(\mathbf{f}_{x_j}, P) \\
 & \text{and } \sum_{\mathbf{f} \in F \setminus \mathbf{f}_{x_i}} |\text{comp}(\mathbf{f}) \cap \text{comp}(\mathbf{f}_{x_i})| < \sum_{\mathbf{f} \in F \setminus \mathbf{f}_{x_j}} |\text{comp}(\mathbf{f}) \cap \text{comp}(\mathbf{f}_{x_j})|
 \end{aligned}$$

The extension of the HotSpot focus on sub-components is that of finding “transitive relationships” between HotSpot components on the component and sub-component level: HotZones. The idea is to build an initial map of all HotSpots in a plan, and then to re-evaluate each individual HotSpot according to the HotSpots in its neighborhood:

$$\begin{aligned}
 & \mathbf{m}_i < \mathbf{m}_j \in f_{HZone}^{modSel}(P, \{\mathbf{f}_1, \dots, \mathbf{f}_m\}, \{\mathbf{m}_1, \dots, \mathbf{m}_n\}) \\
 & \text{if } 1 \leq x_i, x_j \leq m, 1 \leq i, j \leq n, \mathbf{m}_i \in modFor(\mathbf{f}_{x_i}, P), \mathbf{m}_j \in modFor(\mathbf{f}_{x_j}, P) \\
 & \text{and } h(\mathbf{f}_{x_i}, \{\mathbf{f}_1, \dots, \mathbf{f}_m\}, 0) < h(\mathbf{f}_{x_j}, \{\mathbf{f}_1, \dots, \mathbf{f}_m\}, 0)
 \end{aligned}$$

The recursive evaluation of the HotSpot value with respect to the HotSpot values of flaws that share components is defined as follows (constant parameter θ modulates the slope of the decay in the influence neighbors’ values with increasing distance d):

$$h(\mathbf{f}, F, d) = \begin{cases} 0 & \text{for } F = \emptyset \\ \sum_{\mathbf{f}_i \in F \setminus \mathbf{f}} |\mathbf{f} \cap \mathbf{f}_i| \cdot e^{-\frac{d}{\theta}} + h(\mathbf{f}_i, F \setminus \{\mathbf{f}, \mathbf{f}_i\}, d + 1) & \text{else} \end{cases}$$

The HotSpot strategies defined so far solely focused on the flaws’ commonalities. As already suggested in [6], overlappings in plan modifications can also be analyzed for commitment estimates. A direct HotSpot calculation is of course possible, however not very promising, since elementary additions never share components directly (all of them are new) and deletion overlappings are seldom (only occur in alternative expansions). We therefore define the modification-selection module $f_{ModBasedHS}^{modSel}$ analogously to the indirect uniform HotSpot function such that it treats the respective domain model entities as sub-components of the referenced components (by doing so, two task insertion modifications overlap if they share the same task schema). A second element of the heuristic is that the influence of transitive references decays (like in the HotZone calculation) because of the strong interdependencies of the domain model components. We refer to the experimental results, however omit this module due to lack of space.

In first experiments with learning strategies, we found in adaptive HotSpot calculation a straightforward enhancement of the uniform calculations. The adaptive variant tries to predict the conflict potential between two flaw classes in the current refinement space exploration and modulates the calculated HotSpot values with that estimate.

$$\begin{aligned}
 & \mathbf{m}_i < \mathbf{m}_j \in f_{DirAdaptHS}^{modSel}(P, F, M) \\
 & \text{if } \mathbf{f}_{x_i}, \mathbf{f}_{x_j} \in F, \mathbf{m}_i \in M \cap modFor(\mathbf{f}_{x_i}, P), \mathbf{m}_j \in M \cap modFor(\mathbf{f}_{x_j}, P) \\
 & \text{and } \sum_{\mathbf{f} \in F \setminus \mathbf{f}_{x_i}} |\mathbf{f} \cap \mathbf{f}_{x_i}| \cdot g(\mathbf{f}, \mathbf{f}_{x_i}) < \sum_{\mathbf{f} \in F \setminus \mathbf{f}_{x_j}} |\mathbf{f} \cap \mathbf{f}_{x_j}| \cdot g(\mathbf{f}, \mathbf{f}_{x_j})
 \end{aligned}$$

Modification-selection module $f_{DirAdaptHS}^{modSel}$ works exactly like the uniform module except every summand is multiplied with the estimate function $g : F \times F \rightarrow \mathbb{R}$ which is defined as follows:

$$g(\mathbf{f}_a, \mathbf{f}_b) = \frac{\text{count}(\mathbf{F}_a, \mathbf{F}_b)}{\max_{\mathbf{f}_x, \mathbf{f}_y \in \mathcal{F}} (\text{count}(\mathbf{f}_x, \mathbf{f}_y))} \cdot g_{user}(\mathbf{f}_a, \mathbf{f}_b) \text{ for } \mathbf{f}_a \in \mathbf{F}_a, \mathbf{f}_b \in \mathbf{F}_b$$

Function *count* records the accumulated amount of HotSpot overlappings between the corresponding flaw classes according to the results of the $f_{DirUniHS}^{modSel}$ computations that have been obtained in the current plan generation and normalizes the value with the current maximum of all count records. The second function g_{user} is a user-defined modulation table in which certain conflict weights can be set. This mechanism allows us to mark, e.g., a HotSpot relationship between an open precondition flaw and an abstract task flaw to be less critical than the one between an open precondition and a causal threat.

Like for the uniform selections, the direct adaptive modification selection can also be formulated as an indirect working module with calls to the respective indirect uniform HotSpot calculations.

Flexible Plan-Selection Strategies: The simplest way of selecting plans is choosing the first or last plan in the fringe, which are the conventional uninformed plan-selection modules $f_{First}^{planSel}$ for a depth-first and $f_{Last}^{planSel}$ for a breadth-first search scheme. If a “depth-first-flavor” is to be added in a combination of strategies, the $f_{LongerHistoryFirst}^{planSel}$ selection prefers plans that have been obtained by a longer sequence of modification applications (the inverse strategy favors leaves of shorter refinement paths). Other simple plan metrics that can be used to estimate the stage of development for a given plan are based on the size of the constraint sets and in particular on the number of plan steps.

An effective plan-selection module that is based on plan metrics is the following:

$$P_i < P_j \in f_{ConstrPlans}^{planSel}(P_1, \dots, P_n) \text{ if } \frac{(|\prec_i| + |VC_i| + |CL_i|) \cdot |TE_j|}{|TE_i| \cdot (|\prec_j| + |VC_j| + |CL_j|)} > 1$$

The idea behind this selection function is to focus on plans that are more constrained than others and that are therefore more likely to either get completed or turn out a failure. Since processed plans tend to get processed again, it is more of a depth-first search but it is less vulnerable to getting trapped in useless task-insertion paths (for solely inserting tasks reduces the heuristic value).

In the fixed-strategy section, we already mentioned flaw- and modification-based plan metrics. More general forms of that schema compare the relative sizes of detected flaw sets or those of proposed modification sets:

$$P_i < P_j \in f_{LessFlawsPerTask}^{planSel}(P_1, \dots, P_n) \text{ if } \frac{|flaws(P_i)|}{|TE_i|} < \frac{|flaws(P_j)|}{|TE_j|}$$

As our experiments have shown, the heuristic to develop plans that have less flaws per plan step is a very effective estimate (although not admissible in general). Moreover, plan metrics such as $flaws : P \rightarrow \mathbb{N}$ or the size of the plan component sets are computationally extremely cheap.

The last selection functions is related to an A^* -heuristic proposed in [3], which is used as a UCPOP plan-selection. In our framework, such a strategy is modelled as follows:

$$P_i < P_j \in f_{S+OC}^{planSel}(P_1, \dots, P_n) \text{ if } \frac{|TE_i| + |\text{flaws}(P_i) \cap F_{OpenPrec}|}{|TE_j| + |\text{flaws}(P_j) \cap F_{OpenPrec}|} < 1$$

For plan selection, a set of HotSpot-based metrics can be derived from the presented modification-selection functions. We will briefly describe the ideas because their arithmetics are presented above. The first two strategies consider the flaw-based HotSpot situation in the plans. Plan selections $f_{DirUniHS}^{planSel}$ and $f_{IndUniHS}^{planSel}$ accumulate the direct, respectively indirect HotSpot values for each plan and relate these values to the total number of detected flaws. The result is in both cases an average of the direct and indirect HotSpot values for flaws and gives a good estimation of the degree of connectivity between defects in plans.

The HotZone concept can also be transferred to plan-selection task, namely in two ways: the simpler form is identifying the plan with the smallest maximal HotZone value. The $f_{LeastHZone}^{planSel}$ module performs for each plan the calculations of f_{HZone}^{modSel} for each flaw, keeps the maximum HotZone value for each plan, and then prefers the plans accordingly. Similarly to the other HotSpot heuristics, this strategy has also the option to prefer strong or weak interactions in the plan.

The second HotZone derivative is $f_{FewerHZones}^{planSel}$, a module that prefers plans with fewer HotZone *clusters*. Such clusters are identified by performing the f_{HZone}^{modSel} computation: When following the transitive overlappings of components, the set of flaws can be partitioned accordingly into independent sub-sets. The number of identified sub-sets is the number of HotZones.

Following the modification-based modification selection, the examination of plan modifications can contribute to the plan selection as well: Corresponding to the direct uniform HotSpot plan-selection, $f_{FewerModHotSpotRatio}^{planSel}$ is a strategy module that accumulates the computed modification overlappings as provided by $f_{ModBasedHS}^{modSel}$ and relates these values to the total number of plan modifications issued for a particular plan. The average overlapping of plan modifications is intended to estimate the amount of interdependencies between the available flaw resolution proposals on a particular plan, and these interdependencies in turn indicate potentially conflicting refinement options in the future.

4 Evaluation

Most of the selection modules are not suitable for being used as singular strategies, because they cover only particular aspects of information about the search space. Consequently, we combine strategies into sequences of selection modules, as described above. This technique covers a wide area of search control in planning: e.g., performing modification selection, modulated by the available alternative modifications, as it is done in the *fewest alternatives first* heuristic for task-expansion selection of the UMCP system [4], is achieved by employing $f_{PrefMC}^{modSel}(M_{ExpTask})$ as the primary modification selection, and f_{LCF}^{modSel} as the secondary strategy.

The well-known “expand then make sound” (EMS) strategy can be emulated by modification selections (1) f_{AddrFC}^{modSel} with argument $F_{CausalThreat}$, (2) f_{AddrFC}^{modSel} with argument $F_{OpenPrec}$, and (3) f_{PrefMC}^{modSel} with argument $M_{ExpTask}$: the strategy

defers task expansion until the current plan is sound. The reverse combination correlates with the UMCP-style of HTN planning, which expands until the very concrete level and then fixes the causal structure.

Although our framework provides the means, a systematic experimental evaluation of all strategy combinations is not feasible. About 40 modification and 60 plan-selection modules – combined to triples – are far too many configurations to test; even though this includes many trivially fruitless combinations, such as fixed strategies that avoid, respectively prefer the same flaw classes. It also became apparent that all HotSpot variants are only performing well in the “avoidance” mode, and the same holds for plan selections that prefer less constrained or developed plans. Seeking HotSpots implies an early commitment and going after un-constrained plans tends to degrade into breadth-first search.

Our experiment domains were the UMTranslog domain from [6], a hierarchical variant of the Satellite competition domain, and an artificial domain “Criss-Cross”. Its name is derived from the fact that the causal structure includes many interleaving causal dependencies, and plans in that domain have typically many HotSpots. The solution space for CC-problems is relatively sparse, while the larger problems in the Satellite and UMTranslog domains can be solved in multiple ways. UMTranslog and CC share a deep task expansion hierarchy and many (typing-) constraints in their methods. This makes both domains behave exactly the same way during our experiments: every strategy for UMTranslog is performing well in CC and vice versa. Satellite problems are a little bit less constrained and hierarchically structured. This may be the reason why they need different strategies and may also explain, why larger problems, which offer a larger selection of modification options, are sometimes easier to solve for the HotSpot strategies than smaller ones.

For this presentation, we conducted a series of experiments with only binary combinations of modification selection and a plan selection with the modules $f_{LessFlawsPerTask}^{planSel}$ and $f_{FewerModsFirst}^{planSel}$, which is a good compromise between a prediction of the future efforts and the branching factor for the newly explored node. Table 1 shows the results in terms of visited plans for finding a solution. The table contains no pure fixed strategy components, because their performance was between one and two orders of magnitude worse and also afflicted with a high variance. The first row displays our reference strategy from [6].

The Satellite domain turned out to be particularly vulnerable to variance problems and this is reflected in many runs of the smaller problems performing worse than the larger ones. Also because of high sample variances, we used the established *LCF* as a primary selection and used other flexible strategies for modulation. Two interesting observations could be made: using *LCF* for modulation did not work very well in most cases because the primary HotSpot strategies favor independent (in terms of plan structures) flaw situations, and those have in turn more modifications available on average due to a higher degree of freedom. I.e., a secondary *LCF* strategy mostly contradicts the primary HotSpot and cannot stabilize its decisions. The second interesting outcome is the *HZone/LCF* combination. It is a good example for the dependency of

Table 1. Modification-selections for the Satellite and CrissCross domains

| f^{modSel} primary – secondary | Sat1 | Sat2 | Sat3 | Sat4 | CC1 | CC2 | CC3 | CC4 | CC5 |
|----------------------------------|------|------|------|------|-----|-----|-----|-----|-----|
| LCF – PrefMC($M_{ExpTask}$) | 62 | 50 | 957 | 886 | 21 | 33 | 73 | 95 | 190 |
| LCF – DirUniHS | 43 | 48 | 771 | 854 | 21 | 34 | 75 | 92 | 185 |
| LCF – DirAdaptHS | 31 | 42 | 596 | 418 | 21 | 31 | 71 | 95 | 184 |
| LCF – IndUniHS | 24 | 37 | 475 | 367 | 21 | 30 | 69 | 95 | 186 |
| LCF – IndAdaptHS | 33 | 40 | 468 | 482 | 22 | 29 | 71 | 87 | 209 |
| LCF – HZone | 24 | 40 | 424 | 375 | 21 | 30 | 71 | 98 | 188 |
| HZone – LCF | 36 | 68 | 399 | 718 | 22 | 30 | 70 | 99 | 161 |
| LCF – ModBasedHS | 36 | 128 | 945 | 702 | 22 | 33 | 70 | 93 | 204 |

domain characteristics and strategy quality: The HotZone is better informed than *LCF* in a domain with many interdependencies, while *LCF* performs better if that information is scarce. Furthermore, it was the only stable strategy in the experiments for the larger Satellite problems.

Table 2 documents the performance of different single plan selections for a given least-commitment modification selection. It has to be noted that only the first strategy (our reference plan selection) and the “fewer HotZones” produce relatively stable results. High sample variances with encouraging low minima indicate that further experimentation with modulating secondary plan selections have to be conducted.

Table 2. Plan-selections for the crisscross domain with $f_{LCF}^{modSel} - f_{PrefMC}^{modSel}(M_{ExpTask})$

| $f^{planSel}$ | CC1 | CC2 | CC3 | CC4 | CC5 |
|------------------|-----|-----|-----|-----|-----|
| LessFlawsPerTask | 21 | 33 | 73 | 95 | 190 |
| DirUniHS | 22 | 42 | 119 | 159 | 220 |
| IndUniHS | 22 | 38 | 125 | 152 | 210 |
| LeastHZone | 20 | 35 | 124 | 145 | 215 |
| FewerHZones | 21 | 34 | 72 | 94 | 205 |
| FewerModBHS | 22 | 35 | 189 | 365 | 420 |

Our empirical analysis confirms previous results that flexible strategies are suitable domain-independent search procedures. It also raised a number of interesting questions, e.g., why every combination including the FewerHZones plan selection performs very well for CrissCross problems but only poor for Satellite problems, while LeastHZone produces the opposite behaviour. The differences between those two heuristics are very subtle and it has to be clarified, which property of a domain or problem they specifically address.

5 Conclusions and Future Developments

We presented a formal framework for refinement-based planning in which the functionality for generation plans is decomposed into plan-deficiency detection,

plan-modification computation, and (tactical) search reasoning. In particular, we focused on search strategies: how to systematically construct strategic guidance and how to evaluate its performance. Strategies from the literature have been integrated into this framework and novel ones have been developed.

The newly developed search strategies are procedures that are capable of reasoning about the interaction of flaws and plan modifications *in general*, thereby becoming unlimited in their applicability and expandability: flexible strategies can be deployed in any system configuration, ranging from hybrid planning to resource planning, etc.

However, more experimental evidence has to be collected in order to get more insight into the relation between modification and plan selections and their performance in specific domains. Future work also includes the investigation of dynamic strategies, and a more thorough coverage of the available strategy spectrum. In particular, we will deal with strategy performance in an integrated planning-and-scheduling system.

Acknowledgement. The authors would like to thank Andreas Lanz who contributed to the strategy implementation and the experimental evaluation.

References

1. Joslin, D., Pollack, M.: Least-cost flaw repair: A plan refinement strategy for partial-order planning. In: Hayes-Roth, B., Korf, R. (eds.) Proc. of the 12th National Conference on AI, pp. 1004–1009. AAAI (1994)
2. Peot, M.A., Smith, D.: Threat removal strategies for partial-order planning. In: Proc. of the 11th National Conference on AI, pp. 492–499 (1993)
3. Schubert, L.K., Gerevini, A.: Accelerating partial order planners by improving plan and goal choices. In: Proc. of the 7th IEEE International Conference on Tools with AI, pp. 442–450. IEEE Computer Society Press, Los Alamitos (1995)
4. Tsuneto, R., Nau, D., Hendler, J.: Plan-refinement strategies and search-space size. In: Steel, S., Alami, R. (eds.) ECP 1997. LNCS, vol. 1348, pp. 414–426. Springer, Heidelberg (1997)
5. McCluskey, T.L.: Object transition sequences: A new form of abstraction for HTN planners. In: Chien, S., Kambhampathi, R., Knoblock, C. (eds.) Proc. of the 5th International Conference on AI Planning Systems, pp. 216–225. AAAI (2000)
6. Schattenberg, B., Weigl, A., Biundo, S.: Hybrid planning using flexible strategies. In: Furbach, U. (ed.) KI 2005. LNCS (LNAI), vol. 3698, pp. 258–272. Springer, Heidelberg (2005)
7. Schattenberg, B., Biundo, S.: A unifying framework for hybrid planning and scheduling. In: Freksa, C., Kohlhase, M., Schill, K. (eds.) KI 2006. LNCS (LNAI), vol. 4314, pp. 361–373. Springer, Heidelberg (2007)
8. Biundo, S., Schattenberg, B.: From abstract crisis to concrete relief – A preliminary report on combining state abstraction and HTN planning. In: Cesta, A., Borrajo, D. (eds.) Proc. of the 6th European Conference on Planning (2001)

Learning How to Play Hex

Kenneth Kahl, Stefan Edelkamp, and Lars Hildebrand

Computer Science Department
University of Dortmund*

Abstract. In HEX two players try to connect opposing sides by placing pieces onto a rhombus-shaped board of hexagons. The game has a high strategic complexity and the number of possible board positions is larger than in CHESS. There are already some HEX programs of recognizable strength, but which still play on a level below very strong human players. One of their major weaknesses is the time for evaluating a board.

In this work we apply machine learning for the computer player to improve his play by generating an fast evaluation function and lookup procedure for pattern endgame databases. The data structures used are neural networks for the evaluation of a position and limited branching trees to determine if a position can be classified as won or lost.

1 Introduction

In this work we study the strategic game HEX a fully observable two-player zero-sum board game. In the same class of problems we also find CHESS and CHECKERS, which both refer to a long line of AI research. The first article to computer CHESS was published by Shannon [14], while the latest result is that a commercial CHESS playing program has beaten the world champion in a match on a regular PC¹ [19]. Schaeffer et al. [13] could prove that a certain CHECKERS opening (called the *White Doctor*) results in a draw, assuming optimal play.

Hex has been invented by the Danish mathematician Piet Hein in the year 1942, calling it *polygon*. Independently, John F. Nash studied the game in 1947, and wrote an article about it in 1952 [10]. In the same year, Parker Brothers firstly sold the game using the name HEX. For the general audience, HEX has been advertised by Martin Gardner [4,5]. The game is played on a squared-sized rhombus-shaped board, consisting of B hexagons. The board size can be scaled, the classical size is 11×11 . The two players are called *black* and *white*. Alternatively, each player places one piece on the board of his color. The task in the game is to generate a chain of own pieces that connects two opposite sides of the board (see Figure 1). The number of reachable boards accumulate to

$$\binom{121}{1, 0, 120} + \binom{121}{1, 1, 119} + \binom{121}{2, 1, 118} + \dots + \binom{121}{61, 60, 0} \approx 4.7 \times 10^{56}.$$

* The work is supported by DFG in the project ED-74/3.

¹ Deep Fritz won against Vladimir Kramnik with 4:2 running on a Intel dual-core with 3 Ghz and 4 GB RAM.

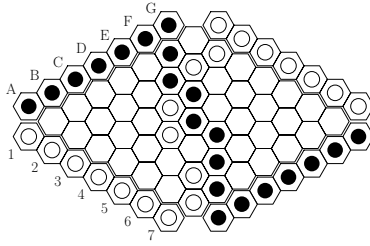


Fig. 1. Terminal position in HEX - game won by black

As intermediate boards may already be classified either to be won for black or for white, the number feasible positions is lower – an upper bound of 2.38×10^{56} has been computed by Browne [2].

As the first player has a big advantage, there is an additional, so-called pie-rule. The first piece is set by player black. Next player white has the choice whether or not he wants to continue the play or swap colors for the rest of the game. The pie-rule is applicable only after the first move. It is well-known that HEX is won by the first player [3]. As the proof is not constructive, for the design of a HEX game playing program the result is of no use. For a growing board, HEX is PSPACE-complete [12]. For board sizes 7×7 , 8×8 and 9×9 without pie-rule winning strategies have been provided by [20], together with a winning strategy for the 7×7 game with pie-rule.

In this paper we apply machine learning algorithms to generate a strong Hex player. On the one hand, we train a neural network with the evaluation function computed by the state-of-the-art program Six². The learned function can be evaluated much faster than the original one, which for a given time slot allows to search the game tree much deeper. For CHECKERS [16] and BACKGAMMON [17] neural networks already yield good players, while for GO so far no strong player could be crated [2]. Moreover, the CHESS program *NeuroChess* could not advance to human play [18]. These approaches learn from the final outcome of games, and they recursively learn the evaluation function. As the second learning mechanism, we will construct a database, in which goals are stored in form of sub-boards (patterns). As a compromise between space and time for insertion and lookup, we propose *limited branching trees*. Additionally, when inserting a goal pattern into the database, symmetric patterns are taken care of.

The paper is structured as follows. First we introduce virtual connections, as they play a central role for the evaluation function and goal detection in the world’s best HEX playing program Six. Then we turn to learning the evaluation function by training neural networks and cross-validation to detect the best network structure. Next we consider limited branching trees as the data structure for storing goal pattern. This subset dictionary structure supports containment queries and provides a fair compromise between searching time and memory consumption. We then turn to experiments that we obtained by integrating

² <http://six.retes.hu>

the above technique into a game playing system. The much larger number of evaluated nodes per second allows deeper searches in the game tree and earlier matches in the databases, and, subsequently, stronger play.

2 Virtual Connections

The current state-of-the-art program Six uses an unusual approach of electrical circuit theory to combine the influence of sub-positions – virtual connections – to larger ones. The program provides an improved implementation of Hexy [1], whose designer has invented the theory of virtual connections.

A *virtual semi-connection* (with respect to a given support cell set) of the board connects two pieces (a.k.a two groups of pieces) provided that the player to close the connection moves first. A (full) *virtual connection* allows to connect pieces even if it is the opponent's turn to move.

An example is given in Figure 2. Black is requested to transform the virtual to a real connection, the shaded cells denote the support. For the first case (a), if it is white to move, he cannot avoid black's connection between the two black pieces on the two shaded cells, but he can escape the connection to the right; if it is black to move the connection is trivial. In the second case (b) white can avoid black's connection by placing a piece on the shaded cell in the center.

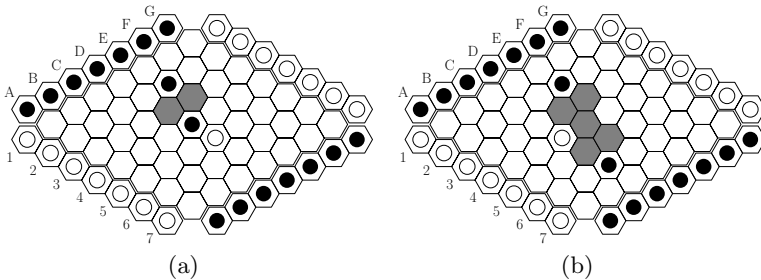


Fig. 2. Virtual connection (left) and virtual semi-connection (right) in HEX

If a virtual connection between opposite borders for one player is established, the game is won (assuming optimal play). The main aspect of evaluating the strength of the position in Six is to find as many virtual (semi-)connections as possible to merge them to more complex patterns and to combine all established connections to an overall game-playing value that is needed to evaluate leaves in the game playing search tree.

Smaller connections are merged to larger ones by applying *and*-, and *or*-rules. The *and*-rule appends two virtual connections, while the *or*-rule combines two virtual semi-connection to one virtual one. Using an analogy to electronic circuit theory, transforming all virtual connections into a single number is based

on computing the resistance for the entire board induced by the resistance for individual cells and connected piece groups.

With this approach Six generates an expressive function to evaluate boards, which itself can look 20 moves (and more) ahead. As a drawback, due to the fact that the evaluation of a single board is quite complex, Six performs a very shallow game tree search, mostly consisting of not more than 1 move for each player. For deeper searches, as e.g. common in CHESS, the time for evaluating one position has to be accelerated.

We decided to improve the performance by applying machine learning. As virtual connections actually serve two purposes, namely evaluating a position and showing whether it is terminating, two different strategies have been developed. Firstly, we monitor the evaluation function and model it using function approximation. A natural choice are neural networks. Secondly, we propose a data structure to store virtual connections that represent a terminal position to generate a pattern database that answers so-called containment queries.

3 Learning the Evaluation Function

The purpose to learn the evaluation function of Six with multi-layered feed-forward neural networks is to reduce the time for evaluating boards in order to be able to search deeper in the game tree. We take an input neuron for every cell of the board. The input cell is 1, if a white piece occupies this cell, 0, if the cell is empty, and -1 , if the cell is occupied by black. In the example of Figure 3 we see a board that is encoded in the following way:

```
0 1 0 0 0 0 0 0 0 0 -1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 -1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 1 3.220380.
```

The first 49 numbers are the inputs for the neural network, while the last number is the evaluation of Six for this board. The neural network has one output neuron, that provides the learned value of the board.

During self- and random play of Six we were able to capture every evaluated board to generate a set of training and validation examples for the neural network. At first we specified the structure of our neural network for the 6×6

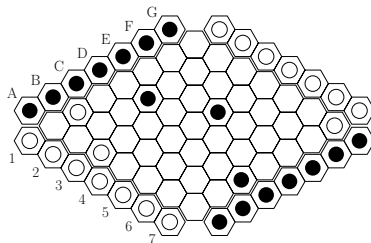


Fig. 3. An evaluated board in HEX

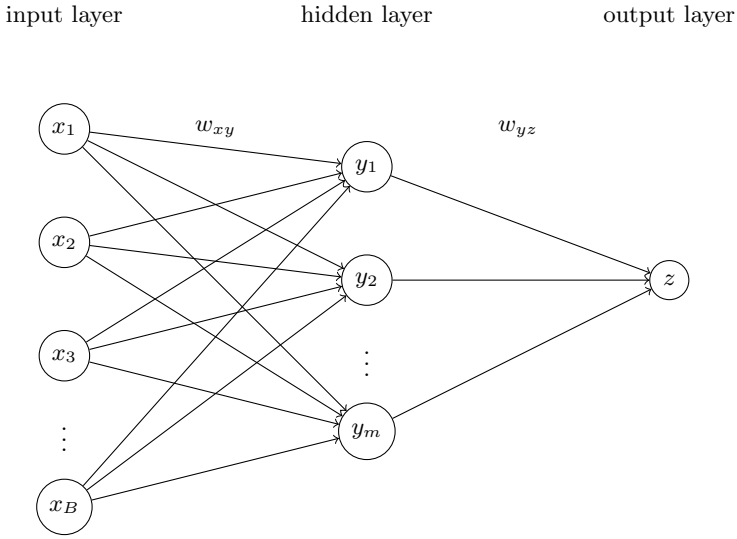


Fig. 4. Structure of the neural network

board. We used a set of 112,737 training examples. After experiments with network structures of up to three hidden layers, we decided to use one hidden layer. We applied a ten-fold cross-validation, where we varied the number of neurons on the hidden layer and the number of epochs, to find a network structure which offers a good generalization. The first criterion for the quality of the generalization was the arithmetic mean of the validation sets, the second criterion was the standard deviation. The best network structure found for the 6×6 board was a network with one hidden layer of $m = 30$ neurons and one output neuron, which was trained for 240 epochs (see Figure 4).

For the 11×11 board we used a set of 142,383 examples. The best network structure we could find is a net with one hidden layer with 50 neurons and one output layer, which was trained over 300 epochs.

Another approach that we implemented was the use of more than one network for evaluating the boards, using a layered partition of the example set. For each search depth (ply) we learned a different evaluation function. Since for the first moves there is only a small number of different boards, the neural network could learn the examples exactly, one property that complies with the experiments.

4 Time Complexity

Now we will show, that our approach outperforms Six. We analyze the complexity for evaluating a board for both programs.

Theorem 1. (*Complexity Neural Network Evaluation*) *Given that the hidden layer in the network has $m < B$ neurons, the running time of the neural network evaluation amounts to at most $O(mB)$.*

Proof. Since the running time for calculating the output of a neural network depends on the inter-connections of the network (it has to be traversed once), we only need to specify the number of network edges. As the network is fully connected, $O(mB)$ steps are executed to evaluate a position.

In the following, we support the observation that the time for evaluating a board in Six is larger than for the neural network with a theoretical comparison. Later on, we will see how the results match with the experimental outcome.

The running time for the evaluation function in Six is dominated by the algorithm *H-Search* [1]. The procedure is rather complex, and takes a maximum number of virtual connections (θ) as well as the maximal number of virtual connections for applying an *or*-rule (θ') as input parameters. For evaluating a board, three steps are executed:

1. The group partition is renewed and the virtual (semi-)connections are adapted.
2. Procedure *H-Search* is executed.
3. The electrical resistance of the circuit is computed.

Suppose a new black piece is placed on the board. If the piece is adjacent to no other black pieces on the board, an individual group is created. If the piece is, however, adjacent to one or more existing one, the groups have to be merged using any union-find data structure. Let B be the number of board cells and θ be a threshold on the number of virtual connections.

Lemma 1. (*Step 1*) *Updating the groups in Six requires $O(B^2\theta)$ operations*

Proof. In Six for each adjacent cell c (out of 6 possible), all groups that contain c are traversed. Since the groups consist of less than B cells the update runs in time $O(B)$. To adapt the virtual and virtual semi-connections, all lists for the support of the group pairs are scanned, which accumulates to $O(B^2\theta)$. Therefore, for step 1 Six requires $O(B) + O(B^2\theta) = O(B^2\theta)$ operations.

Procedure *H-search* updates the lists of virtual (semi-)connections. More precisely, the lists of supports of all virtual connections $VC(g_1, g_2)$ (for all groups g_1, g_2), and all virtual semi-connections $SC(g_1, g_2)$ (for all groups g_1, g_2) are updated. Initially, these lists are empty. Unfortunately, the running time of this approach is large.

Lemma 2. (*Step 2*) *H-Search as implemented in Six requires $O(B^3\theta^2+\theta')$ steps.*

Proof. See [8]

To generate the circuit for each two groups, a wire is inserted into table T , if there is a virtual connection between the two groups by scanning the list of groups (computed by *H-Search*). The list consists of at most B^2 pairs of groups. By playing pieces the number of group pairs can only decrease. According to the Kirchoff's rule, starting with a set of linear equations Six computes the overall resistance [9].

Lemma 3. (Step 3) *The computation of the resistance takes $O(B^3)$ operations.*

Proof. For the insertion of wires all groups are scanned, which takes $O(B^2)$ operations. Solving the set of linear equations can be done in $O(B^3)$ steps.

Theorem 2. (Time Complexity Six) *Computing the evaluation function for requires $O(B^3\theta^{2+\theta'})$ operations.*

Proof. The evaluation function executes the above three steps. Using the lemmas 1, 2 and 3, we arrive at $O(B^2\theta) + O(B^3\theta^2\theta^{\theta'}) + O(B^2) = O(B^3\theta^2\theta^\sigma)$ steps to evaluate a board.

Even if θ and θ' are constants, H-Search requires $O(B^3)$ steps.

5 Goal Pattern Databases

The problem of finding an element in a set of elements such that this element is a subset (or a superset) of the query occurs in many applications, e.g., the matching of a large number of production rules, the identification of inconsistent subgoals in AI planning, and the detection of potential periodic chains in labeled tableau systems for modal logics. Moreover, efficiently storing and searching partial information is central to many learning processes.

5.1 Subset Dictionaries

For state space search the stored sets often correspond to partially specified state vectors or *patterns*. As an example consider the solitaire game SOKOBAN [7], together with a selection of dead-end patterns. As every given state is unsolvable, if the dead-end pattern is a subset of it, we wish to quickly detect, whether or not such dead-end pattern is present in the data structure.

Definition 1. (SUBSET QUERY and CONTAINMENT QUERY Problem, Subset Dictionary) *Let D be a set of n subsets over a universe U . The SUBSET QUERY (CONTAINMENT QUERY) problem asks for any query set $q \subseteq D$ if there is any $p \in D$ with $q \subseteq p$ ($p \subseteq q$). A subset dictionary is an abstract data structure providing insertion of sets to D , while supporting subset and containment queries.*

Since p is a subset of q if and only if its complement is a superset of the complement of q the two query problems are equivalent.

For HEX, we have that each board is an element of U . Inserting a pattern to the goal database amounts to inserting a subset of U to the subset dictionary. In HEX a goal pattern is a virtual connection between both sides of the board (see Figure 7). Subsequently, determining whether or not a state has a match with a stored pattern in the dictionary, is a containment query.

Definition 2. (PARTIAL MATCH) *Let $*$ denote a special don't care symbol that matches every character contained in an alphabet Σ . Given a set D of n vectors over Σ , the PARTIAL MATCH problem asks for a data structure, which for any query $q \in \Sigma \cup \{*\}$ detects if there is any entry p in D such that q matches p .*

The application for this problem is to solve approximate matching problems in information retrieval. A sample application is a crossword puzzle dictionary. A query like $B*T**R$ in the CROSSWORD PUZZLE would be answered with words like BETTER, BITTER, BUTLER, or BUTTER.

Theorem 3. (*Equivalence PARTIAL MATCH and SUBSET QUERY Problems*)
The PARTIAL MATCH problem is equivalent to the SUBSET QUERY problem.

Proof. As we can replace any algorithm for solving the PARTIAL MATCH problem to handle binary symbols by using their binary representation, it is sufficient to consider the alphabet $\Sigma = \{0, 1\}$.

In order to reduce the PARTIAL MATCH to the SUBSET QUERY problem, we replace each $p \in D$ by a set of all pairs (i, p_i) for all $i = 1, \dots, |U|$. Moreover, we replace each query q by a set of all pairs (i, q_i) provided that q is not the don't care symbol $*$. Solving this instance to the SUBSET QUERY problem also solves the PARTIAL MATCH problem.

In order to reduce the SUBSET QUERY to the PARTIAL MATCH problem, we replace each database set with its characteristic vector, and replace query set q by its characteristic vector in which zeros are replaced with don't cares.

As the SUBSET QUERY problem is equivalent to the CONTAINMENT QUERY problem, the latter one can also be solved by algorithms designed for the PARTIAL MATCH problem.

One possible implementation that immediately comes to mind is a *trie*³. It compares a query string with all stored entries. Unfortunately, tries for the PARTIAL MATCH problem can introduce large searching times as each don't care symbol induces a branching.

The next option is to store all possible queries in an array. This solution has constant search time but an obvious problem – the structure can become very large. An alternative to reduce the space complexity for the array representation is to hash the query sets into a smaller table. The lists in the chained hash tables again correspond to database sets. However, the lists have to be searched to filter the elements that match.

5.2 Limited Branching Trees

Our compromise between a trie and a hash table subset dictionary data structure consists of an ordered list of tries. Insertion is similar to ordinary trie insertion with the exception that we maintain a distinctive root for the first element in the sorted representation of the set.

This choice of the data structure adapts *unlimited branching tree* as [6] to our requirements. As the branching factor is bounded, our data structure is referred to as *limited branching trees*, LB trees for short.

³ A trie is a lexicographic search tree structure, in which each node spawns at most $|\Sigma|$ children. The transitions are labeled by $a \in \Sigma$ and are mutually exclusive for two successors of a state. Leaf nodes correspond to stored strings.

The access operation *insert* and *search* in such limited branching trie are realized as follows. Insertion simply traverses the root list to find whether or not a matching root element is present. In case we find a root element the implementation of the ordinary insert routine for the corresponding trie (not shown) is called. In case there is no such element a new one is constructed and inserted to the list. The running time of the algorithm is $O(k + l)$, where k is the size of the current trie list and l the number of elements in the inserted set – plus the time $O(l \log l)$ to sort the elements. As with HEX it is often the case that all elements are selected from the set $\{1, \dots, n\}$ such that the running time is $O(n)$ altogether (given that all elements are distinct, linear sorting algorithms such as bucket sort apply).

Algorithm: Insert

Input: Limited branching tree $L = (T_1, \dots, T_k)$, sorted set $p = \{p_1, \dots, p_l\}$

Output: Modified data structure

```

1 foreach  $i$  in  $\{1, \dots, k\}$  do
2   | if ( $p_1 = \text{root}(T_i)$ ) then
3   |   | return Trie-Insert( $T_i, p$ )
4   |   end
5 end
6 Generate new trie  $T'$  from  $p$ 
7 Insert  $T'$  into list  $L$ 

```

Algorithm 1. Inserting a set in a limited branching tree

In Algorithm 2 we show a possible implementation for the lookup. First all root elements matching the query are retrieved. Then the corresponding tries are searched individually for a possible match with the query. As both the query and the stored set are sorted, the match is available in linear time with respect to the query set. The number of root elements that have to be processed can grow considerably and is bounded by the size of the universe U .

The worst-case running time of the algorithm is $O(km)$, where k is the length of the trie list and where m is the size of the query set – plus the time $O(m \log m)$ to sort the query elements. If all set elements are drawn from the set $\{1, \dots, n\}$ the worst-case running time is bounded by $O(n^2)$.

5.3 Adaption to Hex

For adapting the above structure to a goal pattern database for HEX we remind that a goal pattern represents a virtual connection and consists of a set s of cells and a set of boolean variables b for their occupation. As we store many patterns, we actually maintains a set of sets S for the cells and a set of sets B for the occupation. For each $s \in S$ we have a matching $b \in B$.

Let B be the number of cells on the board. We assume queries of the form $q = (q_s = (q_{s_1}, \dots, q_{s_B}), q_b = (q_{b_1}, \dots, q_{b_B}))$ with $q_{s_i} \in \{1, \dots, B\}$ and $q_{b_i} \in \{0, 1\}$ for $i \in \{1, \dots, B\}$. The answer given is whether or not $s \subseteq q_s$ for one $s \in S$, given that the Boolean assignments in q_b match with the one in b .

Algorithm: Lookup

Input: Limited branching tree $L = (T_1, \dots, T_k)$, sorted query $q = \{q_1, \dots, q_m\}$

Output: Flag indicating whether or not p contained in L with $q \supseteq p$

```

1  $Q \leftarrow \emptyset$ 
2 foreach  $i \in \{1, \dots, k\}$  do
3   | if  $(\text{root}(T_i) \in q)$  then
4   |   |  $Q \leftarrow Q \cup \{T_i\}$ 
5   |   end
6 end
7 foreach  $T_i \in Q$  do
8   | if  $\text{Trie-Lookup}(T_i, q)$  then
9   |   | return true
10  |   end
11 end
12 return false

```

Algorithm 2. Searching for subsets in a limited branching tree

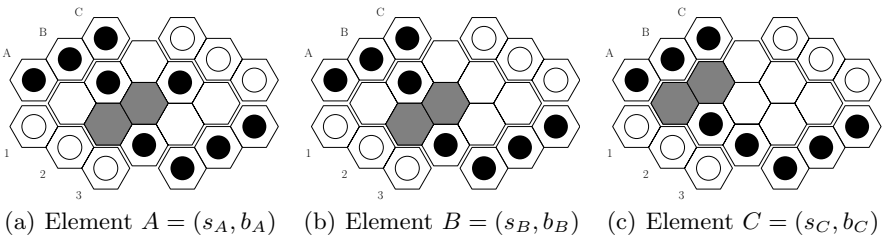


Fig. 5. Goal patterns database example

A node K in the LB tree for the pattern search in HEX represents a cell on the board and consists of three components: the index $I(K)$ of the node K , a sentinel $E(K)$, and a boolean flag $U(K)$ denoting either a black (*true*) or an empty cell (*false*).

As an example, we illustrate the insertion of the patterns of Figure 5 into the empty dictionary. Three patterns, which all show a clear win for black on a 3×3 board, are processed one-by-one. The resulting LB-tree is shown in Figure 6. It consists of two tries, one of which contains a branching.

Beside the color of the pieces there is no difference between a black and a white winning position, such that in order to save memory we only stored black’s winning connections. By swapping the colors and reflecting the board before querying the database, its entries can be reused for white. During the training the opposite strategy applies, all winning connections for white are stored as if they were won by black.

As illustrated in Figure 7 there are many further patterns that can be obtained through translation, reflection, and rotation. For example, the goal pattern in (a) is translated into the pattern in (b). Depending on the width of the pattern,

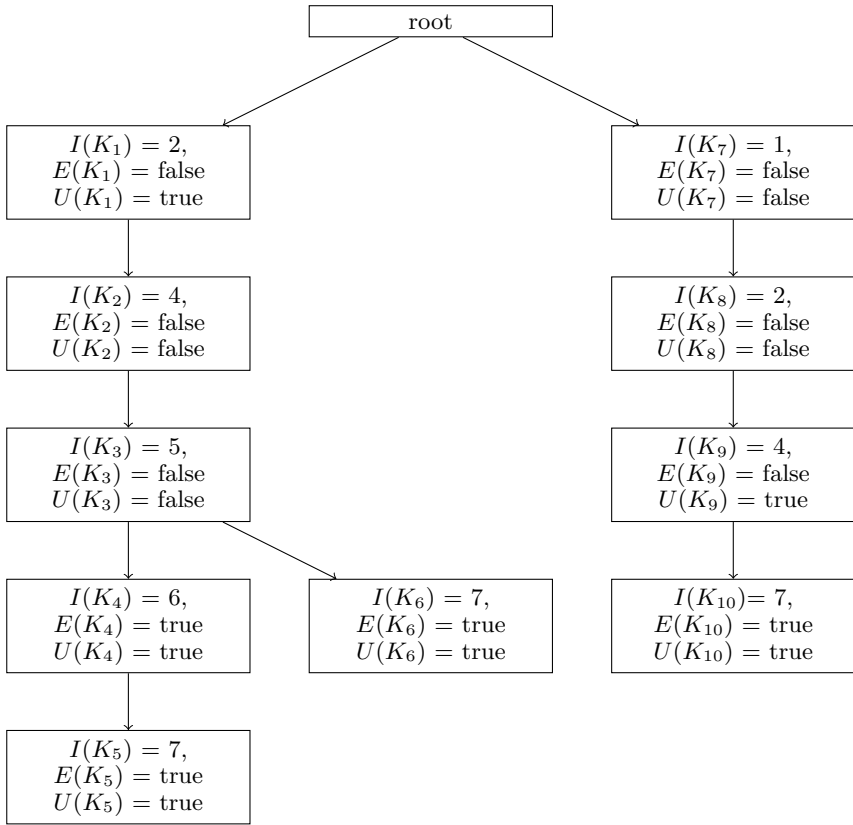


Fig. 6. LB tree for the example

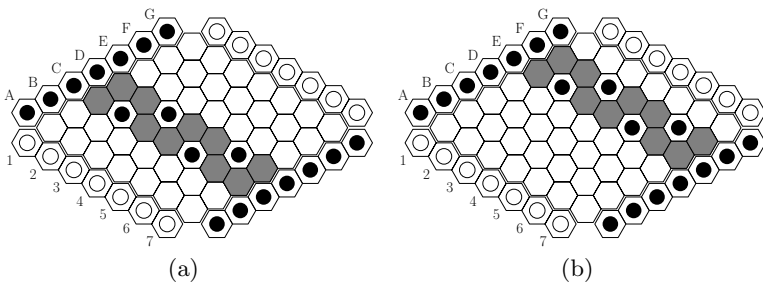


Fig. 7. Hex goal patterns

there are up to $B - 1$ translations possible. For each pattern we have one that is obtained by reflection along the middle axis, such that we insert at most $2(B - 1)$ entries for each established winning position for black. All symmetric patterns are additionally inserted into the limited branching tree.

6 Experiments

We implemented a Hex-playing program on top of Six. Six applies limited depth alpha-beta game tree search. To deal with the large branching factor in HEX only the k -best successors are kept for each depth. We chose the Fast Artificial Neural Network Library (FANN) [11] as the basis for learning the evaluation function and its subsequent use. The Java Native Interface [15] was used to connect a flexible user interface (written in Java) to the underlying system (written in C).

All experiments were run on a Linux PC with 3 GHz Intel Pentium IV with 512 MB RAM. From the range of different board sizes, we selected two case studies: the 6×6 board, which is small enough to completely explore the search tree, and the 11×11 board that is used for tournament play.

The decision in HEX on a 6×6 board can be made after a few moves. The game is won⁴ after three played pieces. Both our HEX-system and Six actually terminate after the third move. For the first move, both systems perform a game-tree search to depth one, where 30 boards are evaluated. In Figure 8 we show a comparison of the evaluation of our system and Six.

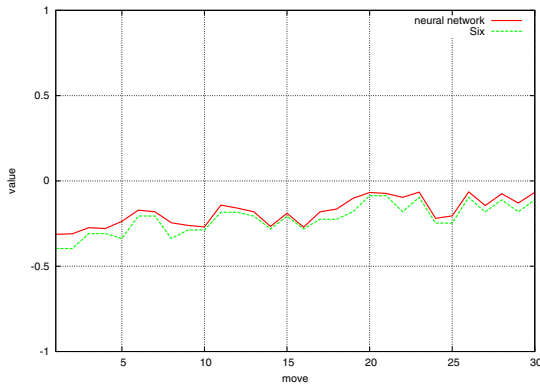


Fig. 8. Comparison between the evaluation function values of our system and Six for a selection of boards

On the one hand, we observe that our system approximates the true value quite good. On the other hand, there is a large difference between the evaluation times. While the evaluation of a board in our system needs less than one millisecond, Six needs up to three seconds and more. In Figure 9 we display the times for evaluating the boards for the first move in Six. The overall time for the first move in our system was 0.57 seconds, while Six required 47.28 seconds.

After the second player has moved, with the third move both systems create a virtual connection between the edges. Our system finds an entry the goal pattern database 0.015 seconds lookup time on the average.

⁴ We say a game is won, if there is a virtual connection between the sides of the board.

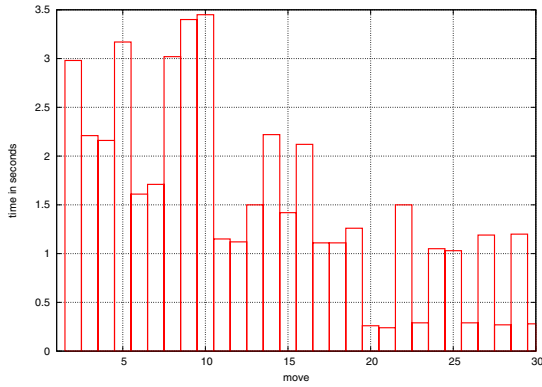


Fig. 9. Distribution on evaluation times for game tree leaves when evaluating the first move in Six

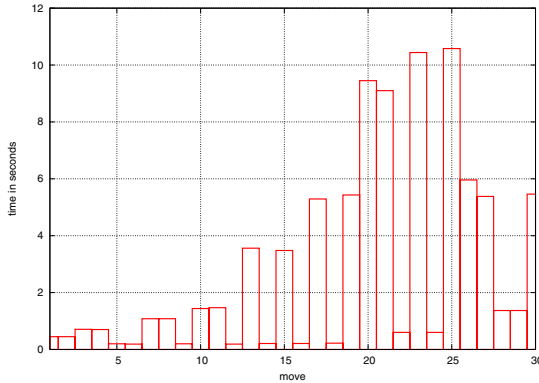


Fig. 10. Distribution on evaluation times for game tree leaves when evaluating the first move on the 11×11 board in Six

Our system can classify the initial state as won for the first player: in a search depth of three without any limit the branching factor on the first move, the root of the game tree evaluates to a definite win for the first player. The search time for this proof was 18.2 seconds, while expanding 1,389 nodes.

Similar results have been obtained for the 11×11 board. The evaluation time of our system is less than one millisecond, while Six needs sometimes more than 10 seconds for evaluating one of the 30 boards for the first move. The time distribution for evaluating these 30 boards for the first move is illustrated in Figure 10. On the same depth limit our system did not find the best move (placing the piece in the middle of the board [2]), so we allowed a search depth

of three. Now the best move was found after 44.26 seconds, which is still faster than the 87.64 seconds that Six needs to calculate the first move with search depth one.

Our system can search at least two levels deeper than Six and, therefore, classify some goal states earlier. The largest amount of time is needed for the lookup in the goal pattern database, with an average of 0.1 seconds.

7 Conclusion

With the combination of a neural network evaluation function and a goal pattern database based on limited branching trees we were able to generate a perfect player for the 6×6 board. For the 11×11 board, a player was generated that is able to choose good moves. Due to the limited range of the first two moves the neural nets computed the evaluation functions precisely without generalization.

We have successfully reduced the time for node expansion and, therefore, increased the search depth. An evaluation on a larger scale (larger number of training example) will produce a much better player. For example, TD-gammon was trained by 1.5 million self-playing games [17].

For a very strong HEX player, the design of a hybrid of Six and our learned system, seems the most plausible avenue for future research. Both systems can concurrently execute game tree search given a fixed time frame. Comparing the evaluations can then exploit the advantage of both approaches. Especially for ending games, our approach searches deeper, which lead to an earlier classification of a board.

References

1. Anshelevich, V.V.: The game of hex: An automatic theorem proving approach to game programming. In: National Conference on Artificial Intelligence (AAAI), pp. 189–194 (2000)
2. Browne, C.: Hex Strategy: Making the Right Connections. A. K. Peters (2000)
3. Gale, D.: The game of hex and the brouwer fixed point theorem. *American Mathematical Monthly* 86, 818–827 (1979)
4. Gardner, M.: *The Scientific American Book of Mathematical Puzzles and Diversions*. Simon and Schuster, New York (1959)
5. Gardner, M.: *The Second Scientific American Book of Mathematical Puzzles and Diversions*. Simon and Schuster, New York (1961)
6. Hoffmann, J., Koehler, J.: A new method to index and query sets. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 462–467 (1999)
7. Junghanns, A.: *Pushing the Limits: New Developments in Single-Agent Search*. PhD thesis, University of Alberta (1999)
8. Kahl, K.: *Maschinelle Lernverfahren für das Strategiespiel Hex*. Diplomarbeit, Universität Dortmund (2007)
9. Litovski, V.B., Zwolinski, M.: *VLSI Circuit Simulation and Optimization*. Kluwer Academic Publishers, Dordrecht (1996)
10. Nash, J.: Some games and machines for playing them. Technical report, Rand Corporation (1952)

11. Nissen, S.: Implementation of a fast artificial neural network library (FANN). Technical report, Department of Computer Science University of Copenhagen (2003)
12. Reisch, S.: Hex ist PSPACE-vollständig. *Acta Informatica* 15(2), 167–191 (1981)
13. Schaeffer, J., Björnsson, Y., Burch, N., Kishimoto, A., Müller, M., Lake, R., Lu, P., Sutphen, S.: Solving checkers. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 292–297 (2005)
14. Shannon, C.E.: Programming a computer for playing chess. *Philosophical Magazine* 41, 256–275 (1950)
15. Java Native Interface., *JavaSoft's Native Interface for Java* (1997)
16. Sutton, R.S.: Learning to predict by the methods of temporal differences. *Machine Learning* 3, 9–44 (1988)
17. Tesauro, G.: Practical issues in temporal difference learning. *Machine Learning* 8, 257–277 (1992)
18. Thrun, S.: Learning to play the game of chess. *Advances in Neural Information Processing Systems*, vol. 7 (1995)
19. Tischbierek, R.: Nur das Schach hat verloren. *Deutsche Schachzeitung* 1, 4–14 (2007)
20. Yang, J., Liao, S., Pawlak, M.: On a decomposition method for finding winning strategy in hex game. In: *Internat. Conf. Application and Development of Computer Games*, pp. 96–111 (2001)

Stochastic Functional Annealing as Optimization Technique: Application to the Traveling Salesman Problem with Recurrent Networks^{*}

Domingo López-Rodríguez¹, Enrique Mérida-Casermeyro¹,
Gloria Galán-Marín², and Juan M. Ortiz-de-Lazcano-Lobato³

¹ Department of Applied Mathematics, University of Málaga, Málaga, Spain
`{dlopez,merida}@ctima.uma.es`

² Department of Electronics and Electromechanical Engineering, University of
Extremadura, Badajoz, Spain
`gloriagm@unex.es`

³ Department of Computer Science and Artificial Intelligence, University of Málaga,
Málaga, Spain
`jmortiz@lcc.uma.es`

Abstract. In this work, a new stochastic method for optimization problems is developed. Its theoretical bases guaranteeing the convergence of the method to a minimum of the objective function are presented, by using quite general hypotheses. Its application to recurrent discrete neural networks is also developed, focusing in the multivalued MREM model, a generalization of Hopfield's. In order to test the efficiency of this new method, we study the well-known Traveling Salesman Problem. Experimental results will show that this new model outperforms other techniques, achieving better results, even on average, than other methods.

1 The Neural Model MREM

A powerful generalization of Hopfield's model appears in the works [11,12], where the model MREM (*Multivalued REcurrent Model*) is presented. This model (in its discrete version) presents two important characteristics which make it very versatile and increase its applicability:

- The output of each neuron, s_i , is a value from the set $\mathcal{M} = \{m_1, m_2, \dots, m_L\}$, which is not necessarily numerical. The state vector of the net is defined as $\mathbf{S} = (s_1, \dots, s_N)$.
- A new concept is introduced: a function f that measures the similarity between the outputs of the neurons. This is the so-called similarity function. So, $f(s_i, s_j)$ represents the similarity between outputs of neurons i and j .

^{*} This work has been partially supported by Junta de Andalucía project number P06-TIC-01615.

Thus, the energy function of this model, which characterizes the behaviour of the net, is as follows:

$$E(\mathbf{S}) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{i,j} f(s_i, s_j) + \sum_{i=1}^N \theta_i(s_i) \tag{1}$$

where $W = (w_{i,j})$ is the synaptic weight matrix, representing the interconnection strength between each pair of neurons, f is the similarity function, and $\theta_i: \mathcal{M} \rightarrow \mathbb{R}$ is a generalization of the thresholds of each neuron.

These characteristics make that certain optimization problems (including the ETSP) have better representation in this model than in the case of the binary or the bipolar model developed by Hopfield, as well as in other multivalued models.

It is clear that MREM generalizes Hopfield's models (with outputs $\mathcal{M} = \{-1, 1\}$ as well as $\mathcal{M} = \{0, 1\}$), if we consider the similarity function given by the product $f(a, b) = ab$.

This model has been successfully applied to several (combinatorial) optimization problems, achieving very good results, which were better than the best known results in literature [13][14][15].

The aim of this work is to present a technique that helps MREM (as well as other optimization techniques) to escape from local minima, improving so its efficiency, by means of a randomized technique that we call Stochastic Functional Annealing.

2 Stochastic Functional Annealing

The Functional Annealing method is designed to help optimization techniques to avoid some local minima of the objective function. In this case, the objective function F (defined over a discrete set V) is substituted by a sequence of functions $\{F_n\}$ such that $\lim_{n \rightarrow \infty} F_n(x) = F(x)$ for all $x \in V$. Then, an optimization technique is used to minimize each of the *approximating* functions F_n .

The point used as initial guess to minimize F_{n+1} is the minimum of F_n , which will be denoted as $x_*^{(n)}$. We iterate by using a stochastic algorithm which, under certain hypotheses, will lead to the minimization of F , although the minimization of F_n is not guaranteed.

This stochastic algorithm is as follows:

1. Begin with $m = 1, n = 1$.
2. Choose an initial guess $x_1^{(n)}$, randomly (if $n = 1$), or by taking $x_1^{(n)} = x_*^{(n-1)}$ (if $n > 1$).
3. Select a point $z \in V$ according to the law of probability $\mathbb{P}_s(z; x_m^{(n)})$.
4. The update $x_{m+1}^{(n)} = z$ will be accepted with probability $\mathbb{P}_a(\Delta F_n)$, where $\Delta F_n = F_n(z) - F_n(x_m^{(n)})$.
5. Repeat steps 3 and 4 until the acceptance of an update $x_{m+1}^{(n)}$. Let $m = m + 1$.
6. If $m \geq K$ (for a fixed K), let $x_*^{(n)} = x_K^{(n)}$, $m = 1$ and $n = n + 1$ and return to step 2.
7. If $m < N$, return to step 3.

Thus, we obtain a sequence $\{x_*^{(n)}\}$ of points in V which approaches to the minima of the corresponding F_n . In the limit, we expect that this sequence approaches a minimum of F .

Note that we are not making an infinite number of iterations in order to minimize F_n . We make only a finite number of them, which implies, due to the stochastic nature of the algorithm, that we are not arriving at the minimum of F_n .

We must observe that there are several possible ways of sampling a point $z \in V$. It does not need to be chosen completely at random. This sampling can be made by taking into account the function values of all points in a certain neighborhood of $x_m^{(n)}$, and assigning to every point a probability which is proportional to its increase of the function value.

In the next section we will impose some conditions to guarantee the minimization of F .

2.1 Convergence Theorems

At least two hypotheses have to be imposed in order to obtain some results of convergence:

Condition of Probabilistic Monotonicity

The next equality must hold

$$\lim_{n \rightarrow \infty} \mathbb{P} \left(F_n(x_{m+1}^{(n)}) > F_n(x_m^{(n)}) \right) = 0$$

for all $m \in \{1, \dots, N - 1\}$.

Condition of Acceptance of the Update $z = x_{m+1}^{(n)}$

We consider the probability of acceptance of $x_{m+1}^{(n)}$ as

$$\mathbb{P}_a(z) = \begin{cases} 1, & \text{if } \Delta F_n < 0 \\ g_n(\Delta F_n) < 1, & \text{if } \Delta F_n \geq 0 \end{cases}$$

where $g_n: \mathbb{R}^+ \rightarrow [0, 1)$ and $\Delta F_n = F_n(z) - F_n(x_m^{(n)})$.

In addition, to simplify the algorithm, let us suppose the following condition:

Condition of Sampling in Neighbourhoods

Given $x = x_m^{(n)}$, we consider the probability of sampling a point $z \in V$ given by

$$\mathbb{P}(\text{sampling } z) = \mathbb{P}_s(z) = \begin{cases} 0, & \text{if } z \notin \mathcal{N}_x \\ a(z) > 0, & \text{if } z \in \mathcal{N}_x \end{cases}$$

where \mathcal{N}_x is a neighbourhood of x .

We must note that if the functional sequence $\{g_n\}$ tends to 0 uniformly, then the condition of acceptance of $z = x_{m+1}^{(n)}$ implies that of probabilistic monotony.

With these hypotheses in mind, we can get some technical results that guarantee the convergence of our algorithm to a minimum of F . However, due to the limitation in the length of this paper, proofs for these results will be omitted.

The following technical results (lemmas, propositions and theorems) establish some quite general conditions to ensure the convergence of our method. These conditions are simplified in the particular case of discrete recurrent networks, in the next section.

Theorem 1. *With probability 1, there exists L such that*

$$L = \lim_{n \rightarrow \infty} F_n(x_*^{(n)}) = \lim_{n \rightarrow \infty} F(x_*^{(n)})$$

Corollary 1. *If ξ is an accumulation point of the sequence $\{x_*^{(n)}\}$, then, with probability 1, the next equality holds:*

$$F(\xi) = \lim_{n \rightarrow \infty} F_n(x_*^{(n)}) = \lim_{n \rightarrow \infty} F(x_*^{(n)})$$

We now proceed to establish the optimality of the accumulation points of $\{x_*^{(n)}\}$.

Theorem 2. *Let $z \in V$ with $F(z) < L$, where*

$$L = \lim_{n \rightarrow \infty} F_n(x_*^{(n)}) = \lim_{n \rightarrow \infty} F(x_*^{(n)})$$

Then, there exists $N \in \mathbb{N}$ such that if $n \geq N$ then

$$\mathbb{P}_s(z) \cdot \mathbb{P}(\text{accept } z = x_2^{(n)}) = 0$$

This technical result gives two interesting corollaries, dealing with the optimality of the accumulation points of $\{x_*^{(n)}\}$, and the convergence of this sequence.

Proposition 1. *Let ξ be an accumulation point of $\{x_*^{(n)}\}$. Then, we have $F(\xi) \leq F(z)$ for all $z \in \mathcal{N}_\xi$. Therefore we can affirm that ξ is a local minimum of F .*

Let us study the time required by this algorithm to visit, at least once, the global optimum x_* .

Suppose that $g_n(\Delta F_n) \geq \delta > 0$ for all n and for all possible $\Delta F_n = F_n(y) - F_n(x)$ with $x, y \in V$ and that $\mathbb{P}_s(z; x) \geq \rho > 0$ for all $z \in \mathcal{N}_x$, for all $x \in V$.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph given by $\mathcal{V} = V$ and the edge $(x, y) \in \mathcal{E}$ if, and only if, $x \in \mathcal{N}_y$.

Suppose that \mathcal{G} is strongly connected, that is, given two nodes (points of V), there always exists a path (sequence of nodes) connecting them by means of edges in \mathcal{E} .

Let D be the diameter of the graph \mathcal{G} , that is, the longest path between to nodes of \mathcal{G} .

Lemma 1. *Under the above hypotheses, given a random initial guess $x_1^{(1)} \in V$, the expected number of steps to visit x_* is less than or equal to $D\rho^{-D}\delta^{1-D}$.*

Proposition 2. *For all $k > 0$, our algorithm visits the global optimum x_* in less than $kD\rho^{-D}\delta^{1-D}$ steps with probability greater than $1 - \frac{1}{k}$.*

But we can say more:

Proposition 3. *For all $k > 0$, our algorithm visits the global optimum x_* in less than $2kD\rho^{-D}\delta^{1-D}$ steps with probability greater than $1 - 2^{-k}$.*

These results are applicable when the algorithm can use some memory to store data, that is, the algorithm is able to remember the best solution up to the current iteration (the *best-solution-so-far*).

When the algorithm has no memory, we look for other results proving convergence.

For the next results, which prove the convergence of the sequence $\{x_*^{(n)}\}$, we need an additional hypothesis on $\{g_n\}$, the sequence that appears in the definition of the probability of acceptance.

Definition 1. *A sequence $\{\varphi_n : [0, \infty) \rightarrow \mathbb{R}\}$ of functions converges ε -uniformly to $\varphi : [0, \infty) \rightarrow \mathbb{R}$ if, for all $\varepsilon > 0$, the sequence $\{\varphi_n|_{[\varepsilon, \infty)}\}$ of functions restricted to the interval $[\varepsilon, \infty)$ converges uniformly to $\varphi|_{[\varepsilon, \infty)}$, that is, if*

$$\lim_{n \rightarrow \infty} \sup_{t \geq \varepsilon} |\varphi_n(t) - \varphi(t)| = 0$$

It is clear that if a sequence converges uniformly, it also converges ε -uniformly, therefore it is a less restrictive hypothesis.

Proposition 4. *If local minima of F are strict, $\{g_n\}$ converges ε -uniformly to 0, and ξ is an accumulation point of $\{x_*^{(n)}\}$, then the next equality holds:*

$$\lim_{n \rightarrow \infty} \mathbb{P}\left(x_*^{(n+1)} = \xi | x_*^{(n)} = \xi\right) = 1$$

But we can still specify a little more:

Corollary 2. *Let us suppose that local minima of F are strict and that $\{g_n\}$ converges ε -uniformly to 0. Let ξ be an accumulation point of $\{x_*^{(n)}\}$ and $\{x_*^{(n_k)}\}$ one subsequence of $\{x_*^{(n)}\}$ such that $\lim_{k \rightarrow \infty} x_*^{(n_k)} = \xi$. Then*

$$\lim_{k \rightarrow \infty} \mathbb{P}(x_*^{(n_k+1)} = \xi) = 1$$

From this result we can deduce the convergence of the sequence $\{x_*^{(n)}\}$.

Theorem 3. *Let us suppose that local minima of F are strict and that $\{g_n\}$ converges ε -uniformly to 0. Let ξ be an accumulation point of $\{x_*^{(n)}\}$. Then*

$$\xi = \lim_{n \rightarrow \infty} x_*^{(n)}$$

Note that we have arrived at the strong convergence of the sequence $\{x_*^{(n)}\}$, not only convergence in probability, to ξ , which is a minimum of F .

3 Application to Recurrent Networks

In this section, we will specify the previous results in the case in which the optimization algorithm used to minimize each F_n is a recurrent neural network. Therefore, we will consider energy functions such as the one given in Eq. (II) for the MREM model.

3.1 Stochastic Functional Annealing Applied to MREM

Now, in order to minimize the energy function E given by Eq. (II), we will consider a sequence of energy functions $\{E_n\}$, defined as in Eq. (2):

$$E_n(\mathbf{S}) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{i,j}^{(n)} f^{(n)}(s_i, s_j) + \sum_{i=1}^N \theta_i^{(n)}(s_i) \tag{2}$$

where $\{W^{(n)} = (w_{i,j}^{(n)})\}$ is a sequence of synaptic weights matrices verifying that $\lim_{n \rightarrow \infty} W^{(n)} = W$, $\{f^{(n)}: \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}\}$ is a functional sequence such that the limit $\lim_{n \rightarrow \infty} f^{(n)}(x, y) = f(x, y)$ for all $x, y \in \mathcal{M}$, and $\{\theta_i^{(n)}: \mathcal{M} \rightarrow \mathbb{R}\}_{n \geq 1}$ is a sequence of threshold (or bias) functions for neuron i , $i = 1, \dots, N$, which converges punctually to θ_i , for all i .

Obviously, $E(\mathbf{S}) = \lim_{n \rightarrow \infty} E_n(\mathbf{S})$ for each state vector $\mathbf{S} \in \mathcal{M}^N$.

For every state vector \mathbf{S} we define a neighbourhood $\mathcal{N}_{\mathbf{S}}$ such that the next state of the net will be chosen from this neighbourhood.

Let us suppose that, to minimize E_n , we iterate and obtain $\mathbf{S}_m^{(n)}$ and make a sampling in its neighbourhood to arrive at the state vector \mathbf{S} . We define the probability of acceptance of the update $\mathbf{S} = \mathbf{S}_{m+1}^{(n)}$ as follows (analogous to what was made in the previous section):

$$\mathbb{P}_a(\mathbf{S}) = \begin{cases} 1, & \text{if } \Delta E_n < 0 \\ g_n(\Delta E_n) < 1, & \text{if } \Delta E_n \geq 0 \end{cases}$$

where $g_n: \mathbb{R}^+ \rightarrow [0, 1)$ and $\Delta E_n = E_n(\mathbf{S}) - E_n(\mathbf{S}_m^{(n)})$.

Thus, for each n , given $\mathbf{S}_1^{(n)}$, we will obtain a finite sequence of state vectors $\{\mathbf{S}_i^{(n)}\}_{i=1, \dots, K}$, since we only iterate K times, to arrive at the state vector $\mathbf{S}_K^{(n)}$ which will be denoted $\mathbf{S}_*^{(n)}$, in order to keep the notation introduced in the previous sections.

So, as a result of applying the previous technical results, we can affirm that:

Theorem 4. *With the previous hypotheses, we have:*

- *There exists a value L which is the common limit of the sequences $\{E_n(\mathbf{S}_*^{(n)})\}$ and $\{E(\mathbf{S}_*^{(n)})\}$.*
- *If \mathbf{S}_* is an accumulation point of $\{\mathbf{S}_*^{(n)}\}$, then*

$$E(\mathbf{S}_*) = \lim_{n \rightarrow \infty} E_n(\mathbf{S}_*^{(n)}) = \lim_{n \rightarrow \infty} E(\mathbf{S}_*^{(n)})$$

and, in addition, $E(\mathbf{S}_*) \leq E(\mathbf{S})$ for all state vector \mathbf{S} belonging to the neighborhood $\mathcal{N}_{\mathbf{S}_*}$ of \mathbf{S}_* .

- If $\{g_n\}$ converges ε -uniformly to zero, and local minima of E are strict, then

$$\mathbf{S}_* = \lim_{n \rightarrow \infty} \mathbf{S}_*^{(n)}$$

3.2 A Particular Case: The Stochastic MREM Model

If we consider $E_n = E$ for all n , we arrive at a stochastic version of MREM.

This version, which will be called sMREM (*stochastic MREM*), will converge to a state of (local) minimal energy, since it is a particular case of the Stochastic Functional Annealing, and it verifies the conditions of Probabilistic Monotonicity, Acceptance of Updates and Sampling in Neighborhoods.

The main drawback of this model sMREM, with respect to its deterministic version, is the amount of computational time needed to achieve that convergence. However, this increase in the computational effort obtains its reward as an increase in the quality of the solution, as we will see in the experimental results.

4 The Euclidean Traveling Salesman Problem

The Euclidean Traveling Salesman Problem (ETSP) is a classical and very well-known issue of study in the field of Operations Research, as well as in Artificial Intelligence, since it has become one of the most popular benchmarks to test the efficiency of optimization-related methods.

We can define this problem as follows: given N cities in the Euclidean space $X_1, \dots, X_N \in \mathbb{R}^2$ and distances $d_{i,j}$ between each pair of cities X_i and X_j , the objective is to find the shortest closed path that visits each city only once.

Real-life applications cover aspects such as automatic route planning for robots, and hole location in printed circuits design [1], as well as gas turbine checking, machine task scheduling or crystallographic analysis [2], among others.

Despite the simplicity of this definition, this problem is one of the most typical representatives of the complexity class of NP-hard problems, showing its high level of difficulty in its resolution. Thus, there is need of algorithms to achieve good approximations to the optimal solution with little time consumption.

For this reason, in addition to classical methods of Operations Research and Optimization, several different algorithms have been developed, including genetic algorithms [3], simulated annealing [4], taboo search [5], and neural networks [6].

Concerning neural networks, the main subject to deal with is to achieve a good representation or formulation of the problem such that its resolution arises as an energy function minimization problem.

In 1985, Hopfield and Tank [7] proposed the first neural network for the study of combinatorial optimization problems (Hopfield's analog model), which was precisely used to solve this problem.

This analog model has more ability to escape from local minima than the discrete model. Some deficiencies are present in both models, as it is the need to fine-tune a high number of parameters in the energy function, as mentioned by Wilson and Pawley [8].

Other approaches are entirely based on Kohonen’s self-organizing maps [9], achieving the best results with the so-called KNIES network [10], in which some statistical measures were incorporated into the original model. This model also presents the drawback of the fine-tuning of a high number of parameters to achieve good results.

In the last few years, the multivalued model MREM has achieved very good results, outperforming KNIES, and presenting the advantage of not needing any adjustment of parameters, quite the opposite to KNIES.

5 The MREM Model for the Travelling Salesman Problem

In order to solve the ETSP with this neural model, two identifications must be made:

- **A network state must be identified to a solution of ETSP**
 A solution of ETSP can be represented as a permutation in the set of numbers $\{1, \dots, N\}$, where N is the number of cities, since it represents the order in which cities are visited. For this reason, the net will be formed by N neurons, each of them taking a value in the set $\mathcal{M} = \{1, \dots, N\}$, such that the state vector $\mathbf{S} = (s_1, \dots, s_N)$ represents a permutation of $\{1, \dots, N\}$ (a *feasible state*). With this representation, $s_i = k$ means that the k -th city will be visited in i -th place.
- **The energy function must be identified to the total distance of the tour**

If we make, in Eq. (II), $f(x, y) = -2d_{x,y}$ and

$$w_{i,j} = \begin{cases} 1 & \text{if } (j = i + 1) \vee ((i = N) \wedge (j = 1)) \\ 0 & \text{otherwise} \end{cases}$$

then the energy function that we obtain is $E(\mathbf{S}) = \sum_{i=1}^{N-1} d_{s_i, s_{i+1}} + d_{s_N, s_1}$, that is, the total distance of the tour represented by the state vector \mathbf{S} .

Computational dynamics is based in beginning with a random feasible initial state vector and updating neurons outputs such that the state vector of the net will always be feasible. To this end, in each iteration, a *2-opt* [16,17,18] update will be made over the current state vector. That is, every pair of neurons, p, q with $p > q + 1$, is studied and the net checks for crosses between the segments $[s_p, s_{p+1}]$ and $[s_q, s_{q+1}]$. In that case, the next inequality holds:

$$d_{s_p, s_{p+1}} + d_{s_q, s_{q+1}} < d_{s_p, s_q} + d_{s_{p+1}, s_{q+1}}$$

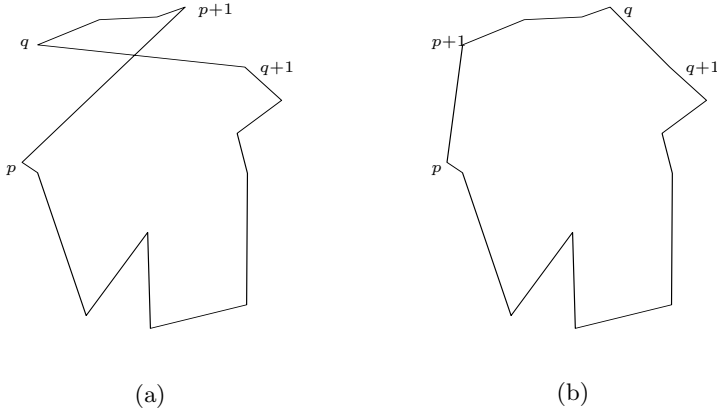


Fig. 1. An example of 2-opt iteration: (a) Original tour. (b) Tour modified by 2-opt technique.

Then, the path from city s_{p+1} to s_q is reversed (see Fig. 1), that is, if \mathbf{S} is the current state, the new state vector \mathbf{S}' will be defined as:

$$s'_i = \begin{cases} s_{q+p+1-i} & \text{if } p+1 \leq i \leq q \\ s_i & \text{otherwise} \end{cases}$$

As an additional technique for improvement, 3-opt updates have also been used: the tour is decomposed in 3 consecutive arcs, A , B and C , which are recombined in all possible ways: $\{ABC, ACB, \hat{A}\hat{B}\hat{C}, \hat{A}\hat{B}\hat{C}, \hat{A}\hat{C}\hat{B}, \hat{A}\hat{C}\hat{B}\}$, where \hat{A} , \hat{B} , \hat{C} are the arcs corresponding to the inversion of A , B , and C , respectively. For example, if $A = (8, 9, 4, 6)$, $B = (1, 5, 3)$ and $C = (2, 7)$, then $\hat{A} = (6, 4, 9, 8)$, $\hat{B} = (3, 5, 1)$ and $\hat{C} = (7, 2)$, and the combination $\hat{A}\hat{C}\hat{B} = (8, 9, 4, 6, 7, 2, 3, 5, 1)$. Note that $\{ABC, \hat{A}\hat{B}\hat{C}, \hat{A}\hat{C}\hat{B}\}$ are 2-opt updates and there is no need to check them again.

The next state of the net will be the combination (chosen from the above list) which decreases most the value of the energy function. For more details, please refer to [12].

6 Functional Annealing for the Resolution of ETSP

In this section, several ways of applying Functional Annealing to ETSP are considered. In each of them we will try to introduce some kind of knowledge about the problem in its resolution, that is, this knowledge will be present in the construction of the sequence $\{E_n\}$.

We will represent this knowledge by means of a transformation of the distances between each pair of cities, that is, we will consider approximating energy functions as in Eq. (2), with $W^{(n)} = W$ for all n , and $f^{(n)}(x, y) = -2d_{x,y}^{(n)}$. The introduction of these functions $d_{x,y}^{(n)}$ will intensify the fact that cities close to

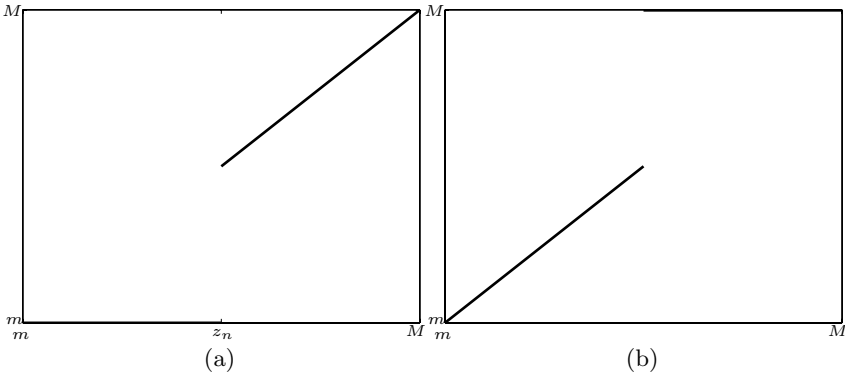


Fig. 2. Graphical representation of the proposed $d_{x,y}^{(n)}$: (a) for FA_1 , (b) for FA_2

each other should be visited consecutively, and that cities which are far away from each other should not. Thus, energy function E_n defined in this way will represent the total distance of the tour associated to the corresponding state vector, but with different measures of the distances between cities.

For simplicity, only a finite number of E_n will be considered: $E_1, E_2, \dots, E_L, E_{L+1} = E$. Let us define a pair of sequences $\{E_n\}$ (it suffices to define $d_{x,y}^{(n)}$) which introduce some of this knowledge in the resolution:

- The first approach will be called FA_1 : In this case, $d^{(n)}$ is defined as

$$d_{x,y}^{(n)} = \begin{cases} d_{x,y}, & \text{if } d_{x,y} \leq \theta_n \\ M, & \text{otherwise} \end{cases}$$

where $\theta_n = m + \frac{(n-1)(M-m)}{L}$, and m and M are, respectively, the minimum and maximum values of $d_{x,y}$ for $x \neq y$.

This election of $d^{(n)}$ makes the algorithm order cities which are close to each other, giving no importance to cities that are far away. That is, in the first few iterations, a partial ordering is induced in the neighborhood of each city. As the neighborhood expands, this series of partial orderings becomes less partial, generating inductively a total ordering that usually represents a better tour in terms of total distance.

- The second approach (FA_2), consists in defining

$$d_{x,y}^{(n)} = \begin{cases} 0, & \text{if } x \in \mathcal{V}_{L-n+1}(y) \vee y \in \mathcal{V}_{L-n+1}(x) \\ d_{x,y}, & \text{otherwise} \end{cases}$$

where $\mathcal{V}_m(X_k)$ is the set formed by X_k and the m nearest cities to X_k , such that $\mathcal{V}_0(X_k) = \{X_k\}$.

This means that, in the first iterations, the model will try to build a global ordering, not focusing in cities that are close to each other, but achieving a rough ordering that can be refined in the following iterations, by taking into account distances between cities close to each other.

In fact, these techniques can help to avoid certain local minima of the energy function E , since the objective function to minimize in each step is E_n , allowing the net to increase temporarily the value of the original energy function.

A graphical representation of $d^{(n)}$ for these two techniques FA₁ and FA₂ can be found in Fig. 2.

7 Experimental Results

Our algorithm has been tested with a wide set of ETSP instances. These instances come from the well-known repository TSPLIB, available on the world wide web and created by Reinelt [19,20]. One of its most important features is that every instance has a record with the distance of the optimal tour, allowing to compare the relative efficiency of our algorithms with respect to MREM. It must be noted that MREM results are better than those of KNIES [12].

Experimental results are shown in Tables 1 and 2. 25 independent executions were made for each instance. The quality measure used in these tables is the percentage of error over the optimum, that is, to compare two results we use their respective relative errors, given by (in percentage):

$$\text{error} = \frac{(\text{Best or average}) \text{ tour length} - \text{Opt}}{\text{Opt}} \cdot 100$$

We have tested the sMREM model and Functional Annealing with dynamics FA₁ and FA₂. The acceptance function g_n considered takes the following form: $g_n(\Delta) = \exp\left(\frac{-\Delta}{T_n}\right)$, where T_n is a ‘temperature’ parameter, converging to 0, what makes $\{g_n\}$ converge ε -uniformly to 0.

Since we only use a finite number of approximating energy functions, we can consider that the temperature parameter T_n decreases linearly from $T_1 = 1$ to $T_L = 0$ and $K = 20$ iterations for each value of the temperature.

The parameter L takes the following values:

- $L = 20$ for sMREM and FA₁, $L = 5$ for FA₂, whose results are shown in Table 1.
- $L = 40$ for sMREM and FA₁, $L = 10$ for FA₂, whose results are shown in Table 2.

We must also specify the way the next state of the net is sampled:

- The increase of energy corresponding to each of the states which are in the neighborhood of the current state, \mathbf{S} , is computed.
- The probability of sampling the state $\mathbf{S}' \in \mathcal{N}_{\mathbf{S}}$ is proportional to the exponential of the opposite of its increment of energy, divided by the temperature at that iteration: $\mathbb{P}_{\mathbf{S}}(\mathbf{S}') \propto \exp\left(\frac{-\Delta_{\mathbf{S},\mathbf{S}'}}{T_n}\right)$.

Once the next state is sampled, it is accepted or not depending on the probability of acceptance \mathbb{P}_a , which is defined in terms of g_n .

Table 1. Comparative results (25 executions per instance) between MREM and the stochastic methods herein proposed for different instances of the Travelling Salesman Problem ($L = 20$ for sMREM and FA₁, $L = 5$ for FA₂)

| Instance | Opt. | KNIES | | | MREM | | | sMREM | | | FA ₁ | | | FA ₂ | | |
|----------|--------|-------|---------|-------|--------|---------|--------|--------|---------|--------|-----------------|---------|--------|-----------------|---------|---|
| | | Best | Average | t | Best | Average | t | Best | Average | t | Best | Average | t | Best | Average | t |
| eil51 | 426 | 2.86 | 2.43 | 3.12 | 0.23 | 1.42 | 11.91 | 0 | 1 | 12.14 | 0.23 | 1.67 | 3.73 | | | |
| st70 | 675 | 1.51 | 1.89 | 9.01 | 0 | 0.94 | 23.09 | 0 | 0.85 | 23.19 | 0 | 1.11 | 8.32 | | | |
| eil76 | 538 | 4.98 | 3.43 | 10.8 | 0.19 | 1.72 | 28.15 | 0 | 1.62 | 28.11 | 0 | 1.8 | 8.87 | | | |
| rd100 | 7910 | 2.09 | 3.02 | 61.7 | 0.62 | 2.99 | 57.66 | 0 | 2.16 | 52.44 | 0.43 | 2.9 | 23.33 | | | |
| eil101 | 629 | 4.66 | 3.51 | 27.76 | 0.48 | 1.78 | 54.44 | 0 | 2.14 | 53.71 | 0.16 | 1.38 | 24.81 | | | |
| lin105 | 14379 | 1.29 | 1.71 | 28.83 | 0.79 | 3.17 | 65.1 | 0.15 | 2.27 | 66.37 | 0 | 2.81 | 29.3 | | | |
| pr107 | 44303 | 0.42 | 0.82 | 49.79 | 0.83 | 1.57 | 76.12 | 0.2 | 0.64 | 71.95 | 0 | 0.45 | 26.59 | | | |
| pr124 | 59030 | 0.08 | 1.23 | 59.51 | 0.26 | 1.87 | 100.98 | 0 | 1.19 | 100.81 | 0.26 | 1.36 | 43.88 | | | |
| bier127 | 118282 | 2.76 | 2.06 | 66.29 | 1.22 | 3.78 | 112.43 | 0.65 | 2.44 | 112.18 | 0.73 | 2.26 | 61.15 | | | |
| kroA200 | 28568 | 5.71 | 3.49 | 6.70 | 318.44 | 4.79 | 7.30 | 474.51 | 3.22 | 4.97 | 377.43 | 4.23 | 260.74 | | | |

Table 2. Comparative results (25 executions per instance) between MREM and the stochastic methods herein proposed for different instances of the Travelling Salesman Problem, second version ($L = 40$ for sMREM and $FA_1, L = 10$ for FA_2)

| Instance | Opt. | KNIES | | | MREM | | | sMREM | | | FA_1 | | | FA_2 | | |
|----------|--------|-------|---------|------|--------|---------|------|--------|---------|------|--------|---------|------|--------|---------|---|
| | | Best | Average | t | Best | Average | t | Best | Average | t | Best | Average | t | Best | Average | t |
| eil51 | 426 | 2.86 | 0.23 | 2.43 | 3.12 | 0 | 1.22 | 22.78 | 0 | 1.03 | 23.25 | 0.47 | 2.19 | 6.66 | | |
| st70 | 675 | 1.51 | 0 | 1.89 | 9.01 | 0 | 0.59 | 43.46 | 0 | 0.58 | 44.1 | 0.59 | 1.43 | 14.26 | | |
| eil76 | 538 | 4.98 | 1.3 | 3.43 | 10.8 | 0 | 1.46 | 52.43 | 0.19 | 1.07 | 53.03 | 0.93 | 2.45 | 16.07 | | |
| rd100 | 7910 | 2.09 | 0 | 3.02 | 61.7 | 1.31 | 4.38 | 103.44 | 0.59 | 2.52 | 97.09 | 0.43 | 2.5 | 36.28 | | |
| eil101 | 629 | 4.66 | 1.43 | 3.51 | 27.76 | 0 | 1.35 | 94.6 | 0.16 | 1.49 | 96.75 | 0.16 | 2.17 | 35.92 | | |
| lin105 | 14379 | 1.29 | 0 | 1.71 | 28.83 | 0.77 | 3.66 | 110.14 | 0.38 | 1.9 | 110.83 | 0 | 1.41 | 42.29 | | |
| pr107 | 44303 | 0.42 | 0.15 | 0.82 | 49.79 | 0.3 | 1.06 | 132.94 | 0.5 | 1.18 | 117.33 | 0.12 | 0.78 | 44.38 | | |
| pr124 | 59030 | 0.08 | 0 | 1.23 | 59.51 | 0.08 | 1.66 | 166.77 | 0.38 | 1.76 | 163.29 | 0 | 1.22 | 63.97 | | |
| bier127 | 118282 | 2.76 | 0.42 | 2.06 | 66.29 | 1.96 | 3.18 | 195.07 | 0.72 | 2.41 | 186.19 | 0.91 | 1.93 | 83.68 | | |
| kroA200 | 28568 | 5.71 | 3.49 | 6.7 | 318.44 | 4.38 | 7.04 | 648.64 | 3.47 | 5.9 | 558.65 | 3.45 | 6.02 | 273.34 | | |

When $n = L$, the next iteration ($n = L+1$) does not use a stochastic dynamics, so it is the *2-opt* (or *3-opt*) mentioned before.

Regarding solution quality, we can observe in Tables 1 and 2 that the formulation that gets the best results is FA₁, achieving the best average results in almost every test instance, showing its ability to escape from local minima. This fact can also be observed by comparing the optimal results of these methods.

It must be noted that results obtained by KNIES were achieved by means of a trial-error process, since this algorithm has to fine-tune a high number of parameters.

In columns labeled 't' in these tables, we can check that the time consumption is not a drawback in this case, since there are instances in which FA₂ is, at least, as fast as MREM and FA₁ does not represent a high increase of time consumption. This kind of 'acceleration' comes from the fact that Functional Annealing gets good solutions in the first iterations. Then, since F_n is very close to F in the last iterations, these good solutions are actually good approximations of the final solution, and the algorithm hardly iterates.

8 Conclusions

In this work we have studied an optimization technique, Functional Annealing, based on stochastic searches that can help to avoid certain local minima of the objective function. This technique also allows us to introduce some knowledge about the problem in its resolution.

We have proposed the theoretical results on which the algorithm is based. These results prove the convergence of Functional Annealing to a local minimum of the objective function.

We have used the proposed techniques to solve the well-known Traveling Salesman Problem. With these methods, we eliminate long paths in the tour, at the same time that crosses are avoided. Not eliminating those long paths is the main cause of the local minima in which an optimization algorithm may get trapped.

The proposed algorithms outperform, in most cases, results obtained by MREM, without a great increase of time consumption.

References

1. Reinelt, G.: The Travelling Salesman. In: Computational Solutions for TSP Applications, Springer, Heidelberg (1994)
2. Bland, R., Shallcross, D.F.: Large traveling salesman problem arising from experiments in x-ray crystallography: a preliminary report on computation. Technical Report No. 730, School of OR/IE, Cornell University, Ithaca, New York (1987)
3. Potvin, J.: Genetic algorithms for the traveling salesman problem. *Annals of Operations Research* 63, 339–370 (1996)
4. Aarts, E., Korst, J., Laarhoven, P.: A quantitative analysis of the simulated annealing algorithm: A case study for the traveling salesman problem. *J. Stats. Phys.* 50, 189–206 (1988)

5. Fiechter, C.: A parallel tabu search algorithm for large scale traveling salesman problems. Technical Report 90/1, Department of Mathematics, Ecole Polytechnique Federale de Lausanne, Switzerland (1990)
6. Potvin, J.: The traveling salesman problem: A neural network perspective. *INFORMS Journal on Computing* 5, 328–348 (1993)
7. Hopfield, J., Tank, D.: Neural computation of decisions in optimization problems. *Biological Cybernetics* 52, 141–152 (1985)
8. Wilson, V., Pawley, G.: On the stability of the TSP problem algorithm of Hopfield and Tank. *Biological Cybernetics* 58, 63–70 (1988)
9. Kohonen, T.: *Self-organizing Maps*. Springer, Heidelberg (1995)
10. Aras, N., Oomen, B.J., Altinel, I.: The Kohonen network incorporating explicit statistics and its application to the Travelling Salesman Problem. *Neural Networks* 12, 1273–1284 (1999)
11. Mérida-Casermeiro, E.: Red Neuronal recurrente multivaluada para el reconocimiento de patrones y la optimización combinatoria. PhD thesis, Universidad de Málaga, Spain (2000)
12. Mérida-Casermeiro, E., Galán-Marín, G., Muñoz Pérez, J.: An efficient multivalued Hopfield network for the travelling salesman problem. *Neural Processing Letters* 14, 203–216 (2001)
13. Mérida-Casermeiro, E., Muñoz Pérez, J., Domínguez-Merino, E.: An n-parallel multivalued network: Applications to the Travelling Salesman Problem. In: Mira, J.M., Álvarez, J.R. (eds.) *IWANN 2003*. LNCS, vol. 2686, pp. 406–413. Springer, Heidelberg (2003)
14. Mérida-Casermeiro, E., López-Rodríguez, D.: Graph partitioning via recurrent multivalued neural networks. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) *IWANN 2005*. LNCS, vol. 3512, pp. 1149–1156. Springer, Heidelberg (2005)
15. López-Rodríguez, D., Mérida-Casermeiro, E., Ortiz-de Lazcano-Lobato, J.O., López-Rubio, E.: Image compression by vector quantization with recurrent discrete networks. In: Kollias, S., Stafylopatis, A., Duch, W., Oja, E. (eds.) *ICANN 2006*. LNCS, vol. 4132, pp. 595–605. Springer, Heidelberg (2006)
16. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the Traveling Salesman Problem. *Operations Research* 21, 498–516 (1973)
17. Johnson, D.S., McGeoch, L.A.: The Traveling Salesman Problem: A Case Study in Local Optimization. In: *Local Search in Combinatorial Optimization*, John Wiley, Chichester (1997)
18. Hoos, H.H., Stuetzle, T.: Traveling Salesman Problems. In: *Stochastic Local Search*, Morgan Kaufman (2004)
19. Reinelt, G.: TSPLIB - a Travelling Salesman Problem library. *ORSA Journal of Computing* 3, 376–384 (1991)
20. Bixby, B., Reinelt, G.: Travelling Salesman Problem library (1999), <http://www.crpc.rice.edu/softlib/tsplib.html>

A Stochastic Local Search Approach to Vertex Cover

Silvia Richter^{1,3}, Malte Helmert², and Charles Gretton¹

¹ NICTA, 300 Adelaide St, Brisbane QLD 4000, Australia
{silvia.richter,charles.gretton}@nicta.com.au

² Albert-Ludwigs-Universität Freiburg, Georges-Köhler-Allee 052,
79110 Freiburg, Germany
helmert@informatik.uni-freiburg.de

³ Institute for Integrated and Intelligent Systems, Griffith University,
170 Kessels Road, Nathan QLD 4111, Australia

Abstract. We introduce a novel stochastic local search algorithm for the vertex cover problem. Compared to current exhaustive search techniques, our algorithm achieves excellent performance on a suite of problems drawn from the field of biology. We also evaluate our performance on the commonly used DIMACS benchmarks for the related clique problem, finding that our approach is competitive with the current best stochastic local search algorithm for finding cliques. On three very large problem instances, our algorithm establishes new records in solution quality.

1 Introduction

Finding a *minimum vertex cover* of a graph is a well-known NP-hard problem [1]. Given an undirected graph $G = (V, E)$, a vertex cover is defined as a subset of the vertices $C \subseteq V$, such that every edge of G has an endpoint in C , i.e. for all $(u, v) \in E : u \in C$ or $v \in C$. The task then is to find a vertex cover of minimum size, or, for the corresponding NP-complete decision problem *k-vertex cover*, to decide whether a vertex cover of size k exists.

Applications of the vertex cover problem arise in network security, scheduling and VLSI design [2]. For example, finding a minimum vertex cover in a network corresponds to locating an optimal set of nodes on which to strategically place controllers such that they can monitor the data going through every link in the network. Algorithms for minimum vertex cover can also be used to solve the closely related problem of finding a *maximum clique*, which has a range of applications in biology, such as identifying related protein sequences [3].

Over the past decade, numerous algorithms have been proposed for solving the vertex cover problem, including evolutionary algorithms [4], ant colony system approaches [5] and complete search [6]. A recent approach of the latter kind that has proven useful in biological applications is the work of Abu-Khzam et al. [3].

In this work, we introduce a stochastic local search algorithm for vertex cover, dubbed COVER (*Cover Edges Randomly*), and show that it achieves excellent

performance on a large variety of benchmarks. For the protein sequencing problems used by Abu-Khazam et al. [3], COVER is several orders of magnitude faster at finding the optimal solution than their approach. On a suite of “hard” benchmarks with hidden optimal solutions [7], COVER performs very well and establishes a new record on the largest instance. Furthermore, we compare the performance of COVER against state-of-the-art solvers for the related independent set and clique problems, showing that we achieve competitive results on a commonly used benchmark set. We also explore the importance of knowing the target solution size k for our algorithm.

The remainder of this article is organized as follows. We first give an overview of related work, including algorithms for the clique and independent set problems. We then describe the general idea of stochastic local search, and the specific details of our algorithm. In Sec. 5, we describe the empirical evaluation of our algorithm conducted on a wide range of benchmarks, after which we conclude.

2 Background and Related Work

Various complete algorithms for k -vertex cover have been derived from the theory of fixed-parameter tractability (FPT) [6,8,9,3]. The characterizing feature of FPT algorithms is that their run-time is bounded by $f(k) \cdot p(n)$, where the dependence $f(k)$ on the *parameter* k may be arbitrary, but the dependence $p(n)$ on the *graph size* n is polynomial. FPT algorithms generally work by first reducing the problem at hand in what is called a *kernelization* phase (transforming the problem to an equivalent problem with smaller parameter k), and then performing a bounded tree search on the remaining problem kernel. Recently, FPT algorithms have been implemented in a parallel fashion [9,3].

Besides FPT methods, there are a number of heuristic approaches to the vertex cover problem. Evans describes an evolutionary approach, and also reviews previous evolutionary algorithms for the problem [4]. Ant colony systems have been employed by Shyu et al. [10] and Gilmour and Dras [5]. Looking beyond work on vertex cover, a wide range of heuristic algorithms have been proposed for related problems, which we review in the following.

2.1 Maximum Clique and Independent Set

A clique of a graph $G = (V, E)$ is a subset of the vertices $K \subseteq V$ such that all vertices in K are pairwise connected. An independent set of a graph $G = (V, E)$ is a subset of the vertices $S \subseteq V$ such that no two vertices in S are connected. A *maximum clique* (*maximum independent set*) is a clique (independent set) of maximum cardinality for a given graph.

Cliques and independent sets are closely related to vertex covers. In particular, a vertex set S is an independent set of G iff $V \setminus S$ is a vertex cover of G , and a vertex set K is a clique of G iff K is an independent set of the complementary graph \overline{G} , in which two vertices are connected iff they are unconnected in G . Hence, the problems of computing maximum cliques or maximum independent

sets can be reduced to the computation of minimum vertex covers. In particular, all these problems are NP-hard, and the associated decision problems are NP-complete.

Many of the algorithms that have been proposed for computing maximum cliques or maximum independent sets in the past have been evaluated on the set of benchmark problems from the Second DIMACS Implementation Challenge in 1992–1993 [11]. The instances in this benchmark set are taken from a variety of application domains and also include examples that are specifically engineered to be “hard”. They can be considered the standard benchmark for algorithms that compute cliques or independent sets.

For the maximum independent set, a recently proposed heuristic approach is the *Widest Acyclic Orientation* algorithm by Barbosa and Campos, which is competitive with the algorithms used in the original DIMACS challenge [12]. Even better results are obtained with the recent QSH algorithm by Busygin, Batenko and Pardalos [13], which outperforms the maximum clique algorithms used during the DIMACS challenge. However, QSH fares badly on two classes of the DIMACS benchmarks.

For maximum clique, the recently published DLS-MC algorithm by Pullan and Hoos seems to deliver the best results, clearly dominating previously published algorithms on the DIMACS benchmark set [14]. DLS-MC stands for *Dynamic Local Search – Maximum Clique* and works by iteratively growing a candidate solution (initially one vertex) and conducting a plateau search where no further improvement is possible. When neither improvement nor new plateau steps are possible, search restarts from a single vertex. The DLS-MC algorithm has since evolved into the *Phased Local Search* algorithm [15], which eliminates the need for tuning a parameter while producing similar results.

3 Stochastic Local Search

Stochastic Local Search (SLS) methods are a popular means of solving notoriously hard combinatorial problems. They can be very effective while usually being conceptually simple [16]. For example, SLS algorithms are state of the art for solving Boolean Satisfiability problems [17]. In the following, we give a short description of SLS using the terminology of the textbook by Hoos and Stützle [16].

An SLS algorithm operates by searching in a space of *candidate solutions* for a problem π , where a candidate solution may not satisfy all of the constraints required by a solution.

Starting from an initial candidate solution, an SLS algorithm iteratively performs a small step to a *neighbouring* candidate solution by perturbing its current candidate solution. These steps, as well as the initialization of the search, may be randomized. The perturbation steps are only based on *local* information and on some *memory state* m of the algorithm (for example a taboo list). Pseudo code for a general SLS algorithm for a decision problem is shown in Alg. 1.

In line 1 of Alg. 1, a candidate solution s and an initial *memory state* m of the algorithm are computed. In line 2, a termination criterion is used to

Algorithm 1. SLS(π)

```

1: initialize ( $s, m$ )
2: while not terminate( $\pi, s, m$ ) do
3:   ( $s, m$ ) = step( $\pi, s, m$ )
4: end while
5: if  $s \in S^*(\pi)$  then
6:   return  $s$ 
7: else
8:   return failure
9: end if

```

determine whether the search should be terminated. In line [3](#) the step function replaces the current candidate solution s by a new candidate solution from the neighbourhood of s , while replacing the memory state m with a corresponding new memory state. If after terminating the search the candidate solution s is in the set $S^*(\pi)$ of solutions to π , then s is returned; otherwise, the algorithm fails.

In the following section, we describe the COVER algorithm as a specific instance of an SLS algorithm by elucidating the exact choices we made for the nature of candidate solutions as well as the initialization, termination and step functions.

4 The COVER Algorithm

COVER is an SLS algorithm for k -vertex cover, i. e. it takes as input a graph $G = (V, E)$ and a parameter k , and searches for a vertex cover of size k of G . Its candidate solutions are subsets of the vertices V of size k (which are not necessarily vertex covers). The step to a neighbouring candidate solution consists of exchanging two vertices: a vertex u that is in the current candidate solution C is taken out of C , and a vertex v which is not currently in C is put into C .

The initial candidate solution is constructed greedily. In detail, COVER builds C by iteratively adding vertices that have a maximum number of incident edges which are not *covered* by C , i. e. they have no endpoint in C , until the cardinality of C is k . When several vertices satisfy the criterion for inclusion in C , COVER selects one of them randomly, with uniform probabilities. Favouring vertices of high degree is a common heuristic for vertex cover algorithms. In fact, we find that some benchmark problems are even solved by this initialization step alone, e. g. the *p-hat* class of the DIMACS benchmark set.

The termination criterion COVER uses is straightforward: at each step, it tests whether its current candidate solution is a vertex cover of G . The algorithm terminates when either a vertex cover is found, or when a maximum number of steps, denoted by MAX_ITERATIONS, has been reached.

The most influential part of an SLS algorithm is the definition of its step function. COVER uses several heuristic criteria to choose which two vertices to exchange in C , but also utilizes a substantial element of randomness, thus striking a balance between guided search and the diversity that is necessary to escape

local optima. This balance is achieved with a simple division of responsibilities: the vertex to be taken out of C is chosen mainly according to heuristics, while the vertex to be put into C is chosen almost randomly.

When choosing possible candidates for inclusion in C , COVER selects uniformly at random an edge e that is not covered (following the strategy popularized by Selman and Kautz for SAT [18]). The vertex added to C is then chosen from one of the endpoints of e , ensuring that e will be covered in the successive candidate solution. When choosing which one of the two endpoints of e to include and which vertex to take out of the current candidate solution, COVER uses a heuristic based on an *edge weighting* scheme: with each edge of G , we associate a positive real number. Intuitively, these weights indicate for each edge how “difficult” it is to cover it – i. e. how difficult it is to find a candidate solution that contains one of the endpoints of that edge.

In the beginning, all edge weights are initialized to a small constant (0.05). In each of the following iterations, COVER adds 1 to the weights of all edges that are not covered. We then derive *vertex* weights from the weights of edges incident to the vertex. We say that a vertex v *potentially covers* an incident edge (v, u) if u is not in the current candidate solution C . Let the weight of a vertex v , $weight(v)$, be the weighted sum of all edges that vertex v potentially covers. An exchange of two vertices a and b , where a is taken out of C and b is put into C then results in a *gain* defined as $weight(b) - weight(a) + \delta$, where δ is the weight of the edge between a and b if they are connected, and zero otherwise. COVER tries to maximize this gain, thereby covering the more “difficult” edges of higher weight with greater priority. Using weights to direct the search of stochastic local search algorithms is popular practice, and a similar approach has led to very good results for the Boolean Satisfiability problem [19].

COVER also employs a taboo list of size 2, keeping track of the vertices last inserted into C and last removed from C . This prevents it from immediately reversing a decision made in the last iteration. Moreover, COVER remembers for each vertex the iteration count at which it was last removed from C . When inserting a vertex into C , COVER favours vertices that have not been in the cover recently. In particular, this “time stamp” criterion is used to break ties between all insertion candidates that result in maximum gain and are not taboo. When removing vertices from the candidate solution, COVER chooses randomly with uniform probability between all removal candidates.

Pseudo code for the algorithm is given in Alg. 2.

5 Empirical Performance Results

We evaluate the performance of COVER on an extensive set of benchmarks from three different sources. In Sec. 5.1, we use graphs representing biological real-world problems, which were kindly provided to us by Abu-Khzam et al. [3]. In Sec. 5.2, we report results on the BHOSLIB benchmark suite [7], a set of graphs with “hidden optimal solutions” that are specifically designed to be hard to solve. In Sec. 5.3, we evaluate our performance on the (complement) graphs of

Algorithm 2. COVER($G, k, \text{MAX_ITERATIONS}$)

```

1: initialize  $C$  greedily with  $|C| = k$ 
2: initialize weights
3:  $iteration\_number = 1$ 
4: while exists uncovered edge and  $iteration\_number < \text{MAX\_ITERATIONS}$  do
5:   choose uncovered edge  $e = (u_1, u_2)$  randomly
6:   choose vertices  $u \in \{u_1, u_2\}$  and  $v \in C$  according to max gain criterion
7:    $C = C \setminus \{v\}$ 
8:    $C = C \cup \{u\}$ 
9:    $taboo\_list = \{v, u\}$ 
10:   $u.time\_stamp = iteration\_number$ 
11:  update weights
12:  increase  $iteration\_number$  by 1
13: end while

```

the Second DIMACS Implementation Challenge for the maximum clique problem [11]. For each benchmark set, we compare against the best results we could find in literature. Unfortunately, we have to compare against a different system for each benchmark set, as we could not obtain the respective programs to run them on the other benchmark sets.

Our experiments were run on a machine with a 2.13 GHz CPU with 2 GB RAM. For comparing different algorithms on the DIMACS benchmark suite, three machine benchmarks are available from the DIMACS web site. When run on our machine, we obtained run-times of 0.51 seconds for r300.5, 3.01 seconds for r400.5, and 11.31 seconds for r500.5.

Due to the random elements of COVER, we ran the algorithm 100 times, with different random seeds, on each instance of each experiment described in this section. In all cases, the MAX_ITERATIONS parameter was set to 100,000,000. For each instance, we report the following information:

- *solution quality*: This is denoted as a triple a - b - c , where a is the number of runs (out of 100) in which the algorithm found a vertex cover with the minimal (or lowest known) cardinality k^* ; b is the number of runs in which a vertex cover of size k^* was not found, but a vertex cover of size $k^* + 1$ was found; and c is the number of runs where only vertex covers of cardinality $k^* + 2$ or worse could be found.
- *median run-time* and *1st quartile run-time*: We ordered the outcomes of the 100 runs by the cardinality of the solution found (the lower, the better) and, in case of vertex covers of equal cardinality, by search time. *Median run-time* is the run-time for the median element in this sequence (i. e., the 50th-best result), and *1st quartile run-time* is the run-time for the element at the 1st quartile (i. e., the 25th-best result). Thus, median run-time is indicative of a “typical run” of the algorithm, and 1st quartile run-time is indicative of the typical performance one might obtain by running the algorithm repeatedly, with four restarts. If the median (or 1st quartile) run did not achieve an optimal solution, the run-time result is reported in parentheses.

5.1 Biological Data

The graphs used in this section originate from phylogeny and correspond to protein sequencing data [3]. The task here is to find maximum sets of closely correlated protein sequences, which can be directly cast as a (weighted) maximum clique problem. Our input graphs are obtained from the biological problems as follows: first a weighted graph is constructed with vertices corresponding to protein sequences, and weighted edges between vertices corresponding to the extent of correlation between two sequences. Then, for a chosen threshold all edges with weights below the threshold are removed. The complement of this graph is the input for a vertex cover algorithm. The problems we use here were obtained from Abu-Khizam et al., who use them in their paper on parallel FPT algorithms for vertex cover [3].

Our run-time results, as well as the results obtained by the algorithm of Abu-Khizam et al. which we will denote by P-FPT (*parallel FPT*) in the following, are shown in Tab. 1. COVER found optimal solutions in all runs on all graphs except for the *globin-15* instance, where 98 of the 100 runs found optimal solutions, with the other two finding solutions of size $k^* + 1$ and $k^* + 3$, respectively.

Table 1. Results on biological problems. $|V|$ and $|E|$ denote the number of vertices and edges of the input graph, k^* the minimum vertex cover size, as determined by Abu-Khizam et al. The performance metrics for COVER (solution quality, 1st quartile run-time, median run-time) are explained in detail at the start of this section. The last column denotes the run-time of the P-FPT algorithm. All run-times are in seconds.

| Instance | Graph | | | COVER | | | P-FPT |
|----------|-------|---------|-------|---------------|-----------------|----------------|---------|
| | $ V $ | $ E $ | k^* | Quality Hist. | Runtime 1st Qu. | Runtime Median | Runtime |
| globin3 | 972 | 3898 | 165 | 100-0-0 | 0.01 | 0.01 | 23 |
| globin7 | 972 | 38557 | 350 | 100-0-0 | 0.01 | 0.01 | 47 |
| globin9 | 972 | 62525 | 378 | 100-0-0 | 0.01 | 0.01 | 227 |
| globin15 | 972 | 149473 | 427 | 98-1-1 | 0.01 | 0.01 | 14 |
| sh2-3 | 839 | 5860 | 246 | 100-0-0 | 0.01 | 0.01 | 22 |
| sh2-4 | 839 | 13799 | 337 | 100-0-0 | 0.01 | 0.01 | 2593 |
| sh2-5 | 839 | 26612 | 399 | 100-0-0 | 0.01 | 0.01 | 7 |
| sh2-10 | 839 | 129697 | 547 | 100-0-0 | 0.01 | 0.01 | 332 |
| sh3-10 | 2466 | 1508850 | 2044 | 100-0-0 | 0.47 | 0.80 | 8400 |

The difference in run-time between the two approaches is compelling. A problem that took P-FPT more than two hours to solve is solved by COVER in less than one second. Of course, our local search algorithm may fail to find an optimal solution, while P-FPT searches exhaustively, guaranteeing optimality. However, as the results show, COVER reliably finds optimal vertex covers.

The approach used by Abu-Khizam et al. has reportedly delivered valuable results in collaboration projects with biologists: they report that, based on the cliques they derived from microarray data, “neurobiologists have identified what appear to be both network structures and gene roles in intra-cellular transport that were previously unrecognized”. The vast difference in run-time between the two approaches prompts the question of what could be gained by using a local search algorithm like COVER in practice.

It is noteworthy that for many application domains of vertex cover, a near-optimal solution may be sufficient. In the protein sequencing domain, for example, choosing a certain correlation threshold has a somewhat arbitrary influence on the size of the greatest clique, which has more impact on the resulting solutions than the fact that the algorithm might be slightly suboptimal. Our algorithm is likely to provide solutions that are very close to the optimal, even for problems where the optimal solution is difficult to find. Hence, for solving large real-world problems, an incomplete algorithm like ours might prove to be more fruitful than a complete, but prohibitively slow algorithm.

5.2 The BHOSLIB Problems

The BHOSLIB problems (“Benchmarks with Hidden Optimal Solutions”) [7] result from translating binary Boolean Satisfiability problems that were generated randomly according to the model *RB* [20]. The satisfiability versions of these benchmarks are guaranteed to be satisfiable, and the model parameters were set to such values that the instances are in the phase transition area of model *RB*. They have been proven to be hard both theoretically and in practice [20]. The full BHOSLIB set of instances we use here is available on the Internet [7]. Some of these instances were also used in the 2004 SAT competition [21].

The problem instances are grouped by size into 8 groups, with 5 graphs per group, where all graphs of a group have the same number of vertices and edges. In addition to the 40 instances that form the actual benchmark suite, there is a single “challenge problem”, a very large graph with 4,000 vertices and 572,774 edges. The minimum vertex cover for this instance has size 3,900.

The first two instances from each of the groups 3–8 (the *frb40–frb59* graphs) were used in the 2004 SAT competition. From the 6th group (*frb53*) onwards, none of the 55 solvers in the 2004 SAT competition was able to solve either of the two instances within a time limit of 10 minutes. For the very large instance, the best solution found up to now was a vertex cover of size 3,904, using a run-time of 3,743 seconds on a Pentium IV 3.4GHz/512MB machine [7]. This solution was obtained by translating the problem to a propositional logic formula extended with cardinality atoms, and using a dedicated solver [22].

The results obtained by COVER on these benchmarks are displayed in Tab. 2. The increasing difficulty of the instances is apparent both in the increasing run-times, as the graph sizes grow, and in the fact that COVER does not find optimal solutions consistently, i.e. in all runs. However, COVER does find optimal solutions for each graph at least once, and the sizes of the vertex covers it finds never exceed the minimum by more than 1.

For comparison, Gilmour and Dras recently developed a series of ant colony system algorithms for the vertex cover problem, evaluated on the BHOSLIB benchmarks [5]. They do not report run-times or solution results for individual graphs, but only the *average* vertex cover size found over all BHOSLIB graphs. (We obtained the detailed results shown in Tab. 2 from personal communications.)

The best result they achieve, using the CKACS algorithm, is an average vertex cover size of 975.875, while 967.25 is the optimal value. This means that the

Table 2. Results on the BHOSLIB benchmark suite

| Instance | Graph | | | COVER | | | CKACS | |
|------------|-------|--------|-------|---------------|------------------------|---------|---------------|--------|
| | $ V $ | $ E $ | k^* | Quality Hist. | Runtime 1st Qu. Median | | Quality Hist. | Avg. |
| frb30-15-1 | 450 | 17827 | 420 | 100-0-0 | 0.06 | 0.08 | 0-1-9 | 424.0 |
| frb30-15-2 | 450 | 17874 | 420 | 100-0-0 | 0.07 | 0.10 | 0-0-10 | 424.5 |
| frb30-15-3 | 450 | 17809 | 420 | 100-0-0 | 0.21 | 0.40 | 0-0-10 | 424.6 |
| frb30-15-4 | 450 | 17831 | 420 | 100-0-0 | 0.05 | 0.08 | 0-0-10 | 424.0 |
| frb30-15-5 | 450 | 17794 | 420 | 100-0-0 | 0.12 | 0.17 | 0-0-10 | 423.6 |
| frb35-17-1 | 595 | 27856 | 560 | 100-0-0 | 0.45 | 0.90 | 0-0-10 | 565.5 |
| frb35-17-2 | 595 | 27847 | 560 | 100-0-0 | 0.40 | 0.84 | 0-0-10 | 566.5 |
| frb35-17-3 | 595 | 27931 | 560 | 100-0-0 | 0.15 | 0.27 | 0-0-10 | 564.4 |
| frb35-17-4 | 595 | 27842 | 560 | 100-0-0 | 0.62 | 1.12 | 0-0-10 | 565.5 |
| frb35-17-5 | 595 | 28143 | 560 | 100-0-0 | 0.34 | 0.49 | 0-0-10 | 564.1 |
| frb40-19-1 | 760 | 41314 | 720 | 100-0-0 | 0.33 | 0.62 | 0-0-10 | 725.6 |
| frb40-19-2 | 760 | 41263 | 720 | 100-0-0 | 4.52 | 10.21 | 0-0-10 | 726.8 |
| frb40-19-3 | 760 | 41095 | 720 | 100-0-0 | 1.37 | 3.17 | 0-0-10 | 727.6 |
| frb40-19-4 | 760 | 41605 | 720 | 100-0-0 | 3.37 | 8.81 | 0-0-10 | 726.1 |
| frb40-19-5 | 760 | 41619 | 720 | 96-4-0 | 21.80 | 63.47 | 0-0-10 | 725.3 |
| frb45-21-1 | 945 | 59186 | 900 | 100-0-0 | 3.54 | 8.48 | 0-0-10 | 908.2 |
| frb45-21-2 | 945 | 58624 | 900 | 100-0-0 | 11.67 | 28.46 | 0-0-10 | 908.5 |
| frb45-21-3 | 945 | 58245 | 900 | 99-1-0 | 28.91 | 70.13 | 0-0-10 | 908.3 |
| frb45-21-4 | 945 | 58549 | 900 | 100-0-0 | 4.90 | 12.28 | 0-0-10 | 908.4 |
| frb45-21-5 | 945 | 58579 | 900 | 99-1-0 | 22.14 | 66.53 | 0-0-10 | 909.1 |
| frb50-23-1 | 1150 | 80072 | 1100 | 89-11-0 | 58.32 | 171.92 | 0-0-10 | 1110.4 |
| frb50-23-2 | 1150 | 80851 | 1100 | 30-70-0 | 543.56 | (1.72) | 0-0-10 | 1109.7 |
| frb50-23-3 | 1150 | 81068 | 1100 | 24-76-0 | (0.70) | (2.61) | 0-0-10 | 1108.3 |
| frb50-23-4 | 1150 | 80258 | 1100 | 100-0-0 | 8.45 | 16.94 | 0-0-10 | 1109.6 |
| frb50-23-5 | 1150 | 80035 | 1100 | 98-2-0 | 24.43 | 88.94 | 0-0-10 | 1110.3 |
| frb53-24-1 | 1272 | 94227 | 1219 | 9-91-0 | (5.17) | (11.31) | 0-0-10 | 1229.9 |
| frb53-24-2 | 1272 | 94289 | 1219 | 34-66-0 | 403.98 | (4.24) | 0-0-10 | 1229.3 |
| frb53-24-3 | 1272 | 94127 | 1219 | 91-9-0 | 65.21 | 157.80 | 0-0-10 | 1231.6 |
| frb53-24-4 | 1272 | 94308 | 1219 | 24-76-0 | (1.26) | (10.74) | 0-0-10 | 1230.5 |
| frb53-24-5 | 1272 | 94226 | 1219 | 84-16-0 | 109.36 | 253.05 | 0-0-10 | 1231.8 |
| frb56-25-1 | 1400 | 109676 | 1344 | 15-85-0 | (8.48) | (20.73) | 0-0-10 | 1356.8 |
| frb56-25-2 | 1400 | 109401 | 1344 | 12-88-0 | (10.06) | (30.33) | 0-0-10 | 1355.7 |
| frb56-25-3 | 1400 | 109379 | 1344 | 76-24-0 | 130.11 | 435.30 | 0-0-10 | 1355.6 |
| frb56-25-4 | 1400 | 110038 | 1344 | 84-16-0 | 85.60 | 291.11 | 0-0-10 | 1354.8 |
| frb56-25-5 | 1400 | 109601 | 1344 | 98-2-0 | 30.45 | 89.58 | 0-0-10 | 1354.6 |
| frb59-26-1 | 1534 | 126555 | 1475 | 11-89-0 | (14.18) | (30.76) | 0-0-10 | 1486.8 |
| frb59-26-2 | 1534 | 126163 | 1475 | 6-94-0 | (18.11) | (40.86) | 0-0-10 | 1486.4 |
| frb59-26-3 | 1534 | 126082 | 1475 | 12-88-0 | (23.08) | (65.04) | 0-0-10 | 1487.8 |
| frb59-26-4 | 1534 | 127011 | 1475 | 1-99-0 | (31.47) | (73.92) | 0-0-10 | 1487.3 |
| frb59-26-5 | 1534 | 125982 | 1475 | 89-11-0 | 90.18 | 292.60 | 0-0-10 | 1487.3 |

vertex covers found by CKACS, on average, have 8.625 more vertices than an optimal solution. In comparison, COVER achieves an average vertex cover size of 967.50 on the BHOSLIB suite, i.e. the vertex covers it finds are only off by 0.25 on average.

On the challenge problem, COVER does not find an optimal solution. Indeed, the designer of the BHOSLIB benchmark set conjectures that this problem will not be solved on a PC in less than a day within the next two decades [7]. However, the COVER algorithm finds a solution of size 3,903 within 71 seconds, surpassing the best solution known so far in terms of both quality and run-time.

5.3 The DIMACS Benchmark Suite

The DIMACS benchmark set is taken from the Second DIMACS Implementation Challenge (1992-1993) [11], a competition targeting the maximum clique, graph

colouring, and satisfiability problems. The maximum clique benchmarks from this competition have since been used in many publications as a reference point for new algorithms [4,12,13,14,15]. The benchmark set comprises 80 problems from a variety of applications. For example, the *C-fat* family is motivated by fault diagnosis, the *johnson* and *hamming* graphs by coding theory, the *keller* group is based on Keller’s conjecture on tilings using hypercubes, and the *MANN* graphs derive from the Steiner Triple Problem [11]. In addition, there are graphs generated randomly according to various models. For example, the *brock* family is generated by explicitly incorporating low-degree vertices into the cover, in order to defeat algorithms that search greedily with respect to vertex degrees [23]. The sizes of the graphs range from less than 30 vertices and ~ 200 edges to more than 3000 vertices and $\sim 5,000,000$ edges.

The results for COVER are shown in Tab. 3 and 4. For comparison, the tables also contain the results obtained by Pullan and Hoos with the DLS-MC algorithm [14]. DLS-MC was also run 100 times with the same limit on iterations as COVER. The times reported are the ones published by Pullan and Hoos [14], and refer to a 2.2GHz Pentium IV machine with 512 MB RAM, which executed the DIMACS machine benchmarks r300.5 (r400.5, r500.5) in 0.72 (4.47, 17.44) seconds. The run-times are thus roughly comparable, our machine being 30–35% faster according to this measure. The “avg.” column shows the *mean* run-time for COVER across all runs where optimal solutions were found, for graphs where both algorithms found optimal solutions. This allows a direct comparison with the corresponding column for DLS-MC, taken from the article by Pullan and Hoos and determined by the same method. Note that this only compares the run-time for the cases *where an optimal solution was found*, and thus ignores runs where the found vertex cover was sub-optimal. Unfortunately, a direct comparison of our median run-time criterion (which we consider more indicative of actual performance because it is also influenced by sub-optimal runs) is not possible with the published results on DLS-MC.

In 75 of the 80 benchmarks, COVER finds a vertex cover of the putative minimum size for that instance. Note that it is only for some graphs of the *brock* family that COVER never finds optimal results. For the *brock* graphs, the cardinality of the vertex covers found by COVER can become as large as $k^* + 5$ in the worst case. This is not surprising, as COVER favours vertices of high degree, which generally is a helpful heuristic for finding minimum vertex covers. The *brock* graphs, however, were explicitly designed to counteract this approach.

Of the 75 instances where COVER finds an optimal solution, in 69 cases it does so consistently, i. e. in all 100 runs. For the remaining instances, there are occasional sub-optimal runs, but COVER always finds vertex covers of cardinality $k^* + 2$ or less. For *MANN_a81*, the putatively hardest problem in this benchmark set, COVER finds an optimal solution in 4 runs, is off by 1 in 3 runs, and off by 2 in the remaining 93 runs.

Comparing against the results of the DLS-MC algorithm, we find that the two algorithms are largely competitive. In fact, many of the benchmarks seem to be too easy for a state-of-the-art solver nowadays. On the graph families *c-fat*,

Table 3. Results on the DIMACS benchmark suite (continued in Tab. 4)

| Instance | Graph | | | COVER | | | DLS-MC | | |
|----------------|-------|---------|------|----------------|---------|---------|---------------|----------------|--------|
| | V | E | k* | Quality Hist. | Runtime | | Quality Hist. | Runtime Avg. | |
| | | | | 1st Qu. | Median | Avg. | | | |
| brock200_1 | 200 | 5066 | 179 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.02 |
| brock200_2 | 200 | 10024 | 188 | 100-0-0 | 0.15 | 0.23 | 0.43 | 100-0-0 | 0.02 |
| brock200_3 | 200 | 7852 | 185 | 100-0-0 | 2.32 | 5.54 | 7.62 | 100-0-0 | 0.04 |
| brock200_4 | 200 | 6811 | 183 | 100-0-0 | 2.04 | 6.52 | 7.90 | 100-0-0 | 0.05 |
| brock400_1 | 400 | 20077 | 373 | 0-0-100 | (0.04) | (0.06) | n/a | 100-0-0 | n/a |
| brock400_2 | 400 | 20014 | 371 | 0-1-99 | (0.04) | (0.05) | n/a | 100-0-0 | n/a |
| brock400_3 | 400 | 20119 | 369 | 60-30-10 | 71.36 | 247.87 | 135.26 | 100-0-0 | 0.18 |
| brock400_4 | 400 | 20035 | 367 | 76-18-6 | 44.16 | 137.68 | 112.98 | 100-0-0 | 0.07 |
| brock800_1 | 800 | 112095 | 777 | 0-0-100 | (0.77) | (1.06) | n/a | 100-0-0 | n/a |
| brock800_2 | 800 | 111434 | 776 | 0-0-100 | (0.60) | (0.98) | n/a | 100-0-0 | n/a |
| brock800_3 | 800 | 112267 | 775 | 0-0-100 | (1.43) | (2.31) | n/a | 100-0-0 | n/a |
| brock800_4 | 800 | 111957 | 774 | 0-0-100 | (0.99) | (1.35) | n/a | 100-0-0 | n/a |
| C125.9 | 125 | 787 | 91 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| C250.9 | 250 | 3141 | 206 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| C500.9 | 500 | 12418 | 443 | 100-0-0 | 0.08 | 0.24 | 0.31 | 100-0-0 | 0.13 |
| C1000.9 | 1000 | 49421 | 932 | 100-0-0 | 1.32 | 3.27 | 5.82 | 100-0-0 | 4.44 |
| C2000.5 | 2000 | 999164 | 1984 | 100-0-0 | 0.82 | 1.84 | 3.78 | 100-0-0 | 0.97 |
| C2000.9 | 2000 | 199468 | 1922 | 84-16-0 | 124.03 | 323.11 | 369.33 | 93-7-0 | 193.22 |
| C4000.5 | 4000 | 3997732 | 3982 | 100-0-0 | 423.08 | 621.38 | 689.74 | 100-0-0 | 181.23 |
| c-fat200-1 | 200 | 18366 | 188 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| c-fat200-2 | 200 | 16665 | 176 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| c-fat200-5 | 200 | 11427 | 142 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| c-fat500-1 | 500 | 120291 | 486 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| c-fat500-2 | 500 | 115611 | 474 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| c-fat500-5 | 500 | 101559 | 436 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 4.44 |
| c-fat500-10 | 500 | 78123 | 374 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| DSJC500.5 | 500 | 62126 | 487 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| DSJC1000.5 | 1000 | 249674 | 985 | 100-0-0 | 0.28 | 0.95 | 2.17 | 100-0-0 | 0.80 |
| gen200_p0.9_44 | 200 | 1990 | 156 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| gen200_p0.9_55 | 200 | 1990 | 145 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| gen400_p0.9_55 | 400 | 7980 | 345 | 100-0-0 | 0.04 | 0.06 | 0.08 | 100-0-0 | 0.03 |
| gen400_p0.9_65 | 400 | 7980 | 335 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| gen400_p0.9_75 | 400 | 7980 | 325 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| hamming6-2 | 64 | 192 | 32 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| hamming6-4 | 64 | 1312 | 60 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| hamming8-2 | 256 | 1024 | 128 | 0-0-100 | (0.01) | (0.01) | n/a | 100-0-0 | n/a |
| hamming8-4 | 256 | 11776 | 240 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| hamming10-2 | 1024 | 5120 | 512 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| hamming10-4 | 1024 | 89600 | 984 | 100-0-0 | 0.01 | 0.01 | 0.11 | 100-0-0 | 0.01 |
| johnson8-2-4 | 28 | 168 | 24 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| johnson8-4-4 | 70 | 560 | 56 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| johnson16-2-4 | 120 | 1680 | 112 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| johnson32-2-4 | 496 | 14880 | 480 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| keller4 | 171 | 5100 | 160 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| keller5 | 776 | 74710 | 749 | 100-0-0 | 0.01 | 0.03 | 0.07 | 100-0-0 | 0.02 |
| keller6 | 3361 | 1026582 | 3302 | 100-0-0 | 12.35 | 15.18 | 15.63 | 100-0-0 | 170.48 |
| MANN_a9 | 45 | 72 | 29 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| MANN_a27 | 378 | 702 | 252 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.05 |
| MANN_a45 | 1035 | 1980 | 690 | 41-59-0 | 246.92 | (0.28) | n/a | 0-100-0 | n/a |
| MANN_a81 | 3321 | 6480 | 2221 | 4-3-93 | (3.36) | (30.89) | n/a | 0-0-100 | n/a |
| p_hat300-1 | 300 | 33917 | 292 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| p_hat300-2 | 300 | 22922 | 275 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| p_hat300-3 | 300 | 11460 | 264 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| p_hat500-1 | 500 | 93181 | 491 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| p_hat500-2 | 500 | 61804 | 464 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| p_hat500-3 | 500 | 30950 | 450 | 100-0-0 | 0.01 | 0.02 | 0.02 | 100-0-0 | 0.01 |
| p_hat700-1 | 700 | 183651 | 689 | 100-0-0 | 0.01 | 0.01 | 0.04 | 100-0-0 | 0.02 |
| p_hat700-2 | 700 | 122922 | 656 | 100-0-0 | 0.01 | 0.02 | 0.01 | 100-0-0 | 0.01 |
| p_hat700-3 | 700 | 61640 | 638 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| p_hat1000-1 | 1000 | 377247 | 990 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| p_hat1000-2 | 1000 | 254701 | 954 | 100-0-0 | 0.01 | 0.04 | 0.04 | 100-0-0 | 0.01 |

Table 4. Results on the DIMACS benchmark suite (continued from Tab. 3)

| Instance | Graph | | | Quality Hist. | COVER Runtime | | | DLS-MC | |
|--------------|-------|--------|-------|---------------|---------------|--------|-------|---------------|--------------|
| | V | E | k^* | | 1st Qu. | Median | Avg. | Quality Hist. | Runtime Avg. |
| p_hat1000-3 | 1000 | 127754 | 932 | 100-0-0 | 0.06 | 0.10 | 0.11 | 100-0-0 | 0.01 |
| p_hat1500-1 | 1500 | 839327 | 1488 | 100-0-0 | 13.80 | 18.25 | 21.27 | 100-0-0 | 2.71 |
| p_hat1500-2 | 1500 | 555290 | 1435 | 100-0-0 | 0.09 | 0.12 | 0.12 | 100-0-0 | 0.01 |
| p_hat1500-3 | 1500 | 277006 | 1406 | 100-0-0 | 0.07 | 0.10 | 0.11 | 100-0-0 | 0.01 |
| san200_0.7_1 | 200 | 5970 | 170 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| san200_0.7_2 | 200 | 5970 | 182 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.07 |
| san200_0.9_1 | 200 | 1990 | 130 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| san200_0.9_2 | 200 | 1990 | 140 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| san200_0.9_3 | 200 | 1990 | 156 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| san400_0.5_1 | 400 | 39900 | 387 | 100-0-0 | 0.05 | 0.14 | 0.12 | 100-0-0 | 0.16 |
| san400_0.7_1 | 400 | 23940 | 360 | 100-0-0 | 0.05 | 0.06 | 0.06 | 100-0-0 | 0.11 |
| san400_0.7_2 | 400 | 23940 | 370 | 100-0-0 | 0.06 | 0.07 | 0.08 | 100-0-0 | 0.21 |
| san400_0.7_3 | 400 | 23940 | 378 | 100-0-0 | 0.08 | 0.12 | 0.13 | 100-0-0 | 0.42 |
| san400_0.9_1 | 400 | 7980 | 300 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| san1000 | 1000 | 249000 | 985 | 100-0-0 | 0.98 | 3.88 | 3.91 | 100-0-0 | 8.36 |
| sanr200_0.7 | 200 | 6032 | 182 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| sanr200_0.9 | 200 | 2037 | 158 | 100-0-0 | 0.01 | 0.01 | 0.01 | 100-0-0 | 0.01 |
| sanr400_0.5 | 400 | 39816 | 387 | 100-0-0 | 0.01 | 0.02 | 0.06 | 100-0-0 | 0.04 |
| sanr400_0.7 | 400 | 23931 | 379 | 100-0-0 | 0.01 | 0.01 | 0.03 | 100-0-0 | 0.02 |

DSJC, *gen*, *hamming*, *johnson*, *p_hat*, *san* and *sanr*, both DLS-MC and COVER consistently find optimal solutions within extremely short run-times. However, on *MANN_a45* and *MANN_a81*, COVER significantly outperforms the DLS-MC algorithm. On these graphs, DLS-MC does not find an optimal solution. On the hard *MANN_a81* instance, DLS-MC indeed only finds solutions that are of distance 2 or more from the optimum, while COVER finds optimal solutions for both graphs. In fact, to our knowledge COVER is the first algorithm to find covers of this quality for the two *MANN* graphs.

On the other hand, on the *brock* family DLS-MC shows far better results than COVER. This can be explained by the fact that DLS-MC uses a parameter called *penalty-delay*, which Pullan and Hoos hand-tuned for each graph to achieve the best possible performance. While for almost all other graphs this parameter was set to a value between 1 and 5, it was set to 15 and 45 for the larger *brock* graphs, encouraging DLS-MC to quite drastically change its usual behaviour in these cases [14]. We conclude that, despite the fact that COVER is designed for vertex cover problems and DLS-MC is designed for clique problems, COVER is competitive with DLS-MC on clique benchmarks. COVER furthermore has the advantage of requiring no parameters, while achieving excellent results but for one special class of artificial graphs.

5.4 Search Without Parameter

To further understand the run-time complexity of COVER, we conduct a set of experiments aimed at determining the importance of knowing k , the target cover size. Most state-of-the-art solvers, including the ones we compared against in this paper, are, like COVER, solving the k -vertex cover problem (or k -clique, respectively). In practice, however, we do not usually know the optimal value

Table 5. Run-time distribution for various parameters

| Graph | Run-time(s) | k^* | $k^* + 1$ | $k^* + 2$ | $k^* + 3$ | $> k^* + 3$ |
|---------------|-------------|--------|-----------|-----------|-----------|-------------|
| Globin7 | 0.54 | 48.15% | 48.15% | 3.70% | | |
| Sh2-5 | 0.70 | 37.14% | 37.14% | 25.71% | | |
| johnson32-2-4 | 0.92 | 28.26% | 28.26% | 26.09% | 11.96% | 5.43% |
| brock200_1 | 1.52 | 21.71% | 17.11% | 17.11% | 16.45% | 27.63% |
| p_hat700-1 | 2.16 | 43.06% | 24.54% | 11.57% | 10.65% | 10.19% |
| keller5 | 5.12 | 40.43% | 7.23% | 5.08% | 5.08% | 42.19% |
| frb30-15-1 | 7.15 | 41.68% | 24.20% | 7.27% | 3.64% | 23.22% |
| hamming10-4 | 10.73 | 47.62% | 25.63% | 2.42% | 2.42% | 21.90% |
| san400_0.5_1 | 15.30 | 34.77% | 22.35% | 15.82% | 19.28% | 7.78% |
| DSJC1000.5 | 88.10 | 97.63% | 1.16% | 0.30% | 0.30% | 0.62% |
| brock200_3 | 166.01 | 99.18% | 0.19% | 0.16% | 0.16% | 0.32% |
| san1000 | 645.26 | 28.25% | 20.13% | 18.73% | 18.72% | 14.16% |
| frb50-23-4 | 1344.88 | 85.85% | 9.78% | 2.42% | 1.42% | 0.53% |
| frb59-26-5 | 17611.90 | 84.57% | 13.84% | 0.89% | 0.35% | 0.35% |

for k . Instead, we want to find a minimum (or close to minimum) vertex cover of unknown size.

The question thus arises whether COVER can be used efficiently for finding a minimum vertex cover by iteratively searching for various decreasing values of k . Specifically, we are interested in determining how much run-time is spent searching for several values of k as opposed to just searching with a known optimal value k^* . We expect that it is much easier to find solutions that are suboptimal than ones that are optimal, and that indeed only the last few runs where k is close to k^* substantially influence run-time.

To test this hypothesis, we extend COVER to an iterative version COVER-I, which runs without a parameter k , as follows. First, COVER-I greedily computes a vertex cover for the input graph. This is done much in the same way as COVER computes an initial candidate solution. Instead of stopping when the size of the candidate solution reaches a prespecified parameter, however, COVER-I keeps adding vertices until the candidate solution is indeed a vertex cover. The size k of this vertex cover is thus an upper bound for the optimal value k^* . COVER-I then iteratively calls COVER as a subroutine, decreasing k each time COVER succeeds in finding a solution within the usual limit of 100,000,000 iterations. When COVER fails to find a solution, COVER-I stops and returns the last solution found.

For our experiment, we select a representative set of graphs containing instances from all three benchmark suites in varying sizes. The results are displayed in Tab. 5. The *run-time* column shows total run-time for COVER-I for the given graphs, summed up for 25 different random seeds, to give an impression of the relative difficulty of these instances. The column k^* shows the percentage of total run-time spent on the final iteration (producing the optimal solution), with columns $k^* + 1$, $k^* + 2$ etc. referring to the previous iterations.

The results largely confirm our expectations. For small graphs, where the search times for the optimal value k^* are already short, run-time is sometimes spread out fairly evenly across iterations; but for the larger graphs, the amount of time spent in the ultimate iteration dominates the total run-time of COVER-I.

6 Conclusion and Outlook

We have presented a stochastic local search algorithm for the vertex cover problem, COVER, and evaluated its performance on a wide variety of benchmarks. COVER is surprisingly effective while being conceptually simple and not requiring any instance-dependent parameters. For biological real-world problems, COVER finds optimal solutions in just a fraction of the time needed by a complete search, which leads us to believe that COVER is a valuable approach for practical problems. On the hard BHOSLIB benchmark set, COVER vastly improves on existing results and sets a new record for the 20-year challenge problem.

Compared to the state-of-the-art solver DLS-MC for the maximum clique problem, COVER shows competitive results on the DIMACS suite. We emphasize the fact that unlike DLS-MC and many algorithms proposed in the literature previously, COVER has not been tuned in any way to the benchmark sets we evaluated it on. The excellent performance of COVER is further underlined by the fact that it sets a new record in solution quality on two large benchmark instances of the DIMACS set. However, COVER did not perform well on the *brock* family of graphs from the DIMACS test set. An obvious direction of future work is therefore to develop further techniques to more reliably escape local minima during search.

Acknowledgements

NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. We thank Abdul Sattar for his input on the direction of this project and Duc Nghia Pham for helpful discussions. We also thank Stephen Gilmour, Wayne Pullan and Michael Langston for providing problem graphs, results, and helpful suggestions.

References

1. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company (1979)
2. Gomes, F.C., Meneses, C.N., Pardalos, P.M., Viana, G.V.R.: Experimental analysis of approximation algorithms for the vertex cover and set covering problems. *Computers and Operations Research* 33(12), 3520–3534 (2006)
3. Abu-Khzam, F.N., Langston, M.A., Shanbhag, P., Symons, C.T.: Scalable parallel algorithms for FPT problems. *Algorithmica* 45, 269–284 (2006)
4. Evans, I.K.: Evolutionary algorithms for vertex cover. In: Porto, V.W., Waagen, D. (eds.) *Evolutionary Programming VII*. LNCS, vol. 1447, pp. 377–386. Springer, Heidelberg (1998)
5. Gilmour, S., Dras, M.: Kernelization as heuristic structure for the vertex cover problem. In: Hess, F., Pauli, S., Pohst, M. (eds.) *Algorithmic Number Theory*. LNCS, vol. 4076, Springer, Heidelberg (2006)

6. Downey, R.G., Fellows, M.R., Stege, U.: Parameterized complexity: A framework for systematically confronting computational intractability. In: *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*. DIMACS Series, vol. 49, pp. 49–99 (1999)
7. Xu, K.: BHOSLIB: Benchmarks with hidden optimum solutions for graph problems (maximum clique, maximum independent set, minimum vertex cover and vertex coloring) – hiding exact solutions in random graphs. Web site, <http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm>
8. Niedermeier, R., Rossmanith, P.: Upper bounds for vertex cover further improved. In: *Proceedings of the 16th Symposium on Theoretical Aspects in Computer Science (STACS'99)*, pp. 561–570 (1999)
9. Cheetham, J., Dehne, F., Rau-Chaplin, A., Stege, U., Taillon, P.J.: Solving large FPT problems on coarse grained parallel machines. *Journal of Computer and System Sciences* 67, 691–706 (2003)
10. Shyu, S.J., Yin, P.Y., Lin, B.M.T.: An ant colony optimization algorithm for the minimum weight vertex cover problem. *Annals of Operations Research* 131(1–4), 283–304 (2004)
11. Johnson, D.S., Trick, M.A. (eds.): *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*. DIMACS Series, vol. 26. American Mathematical Society (1996)
12. Barbosa, V.C., Campos, L.C.D.: A novel evolutionary formulation of the maximum independent set problem. *Journal of Combinatorial Optimization* 8, 419–437 (2004)
13. Busygin, S., Butenko, S., Pardalos, P.M.: A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. *Journal of Combinatorial Optimization* 6, 287–297 (2002)
14. Pullan, W., Hoos, H.H.: Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research* 25, 159–185 (2006)
15. Pullan, W.: Phased local search for the maximum clique problem. *Journal of Combinatorial Optimization* 12, 303–323 (2006)
16. Hoos, H.H., Stützle, T.: *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, San Francisco (2004)
17. Kullmann, O.: The SAT 2005 solver competition on random instances. *Journal on Satisfiability, Boolean Modeling and Computation* 2, 61–102 (2005)
18. Selman, B., Kautz, H.A.: Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp. 290–295 (1993)
19. Thornton, J., Pham, D.N., Bain, S., Ferreira Jr., V.: Additive versus multiplicative clause weighting for SAT. In: *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, pp. 191–196 (2004)
20. Xu, K., Boussemart, F., Hemery, F., Lecoutre, C.: A simple model to generate hard satisfiable instances. In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pp. 337–342 (2005)
21. Le Berre, D., Simon, L.: The SAT'04 competition. Web site, <http://www.lri.fr/~simon/contest04/results/>
22. Liu, L., Truszczynski, M.: Local-search techniques for propositional logic extended with cardinality constraints. In: Rossi, F. (ed.) *CP 2003*. LNCS, vol. 2833, pp. 495–509. Springer, Heidelberg (2003)
23. Brockington, M., Culberson, J.C.: Camouflaging independent sets in quasi-random graphs, pp. 75–88 [11]

A Connectionist Architecture for Learning to Play a Simulated Brio Labyrinth Game

Larbi Abdenebaoui¹, Elsa A. Kirchner¹, Yohannes Kassahun¹,
and Frank Kirchner^{1,2}

¹ Robotics Group, University of Bremen
Robert-Hooke-Str. 5, D-28359, Bremen, Germany
² German Research Center for Artificial Intelligence (DFKI)
Robert-Hooke-Str. 5, D-28359, Bremen, Germany

1 Introduction

The Brio labyrinth, shown in Figure 1(a), is a popular game consisting of a board with holes and walls. In addition, the game has knobs which are used to tip the board in two planar directions for controlling the angle of the board. The aim of the game is to maneuver a steel ball along a marked path from a starting position to a final position on the board by tipping it so that the ball moves without falling into any of the holes. The path is partially bordered by the walls. Some of the walls form corners, where the ball can be controlled easily.

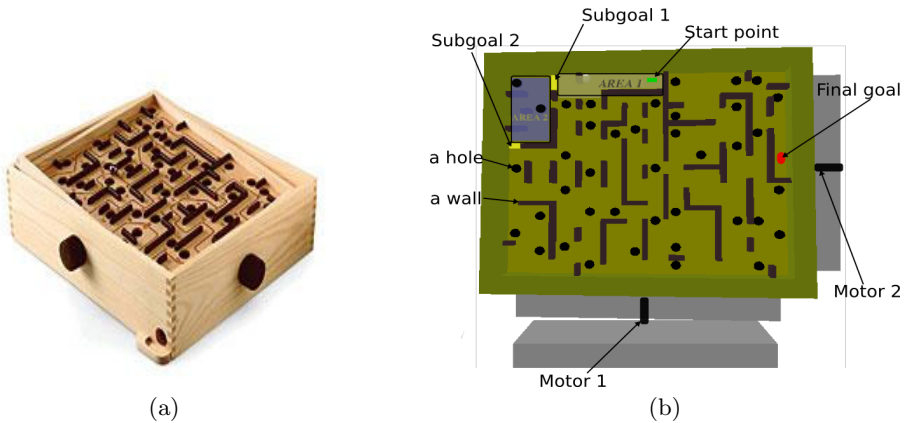


Fig. 1. (a) The Brio labyrinth (b) The simulation with the first two subareas labeled

To enable an artificial agent to play the game, an ODE [3] based simulation of the game has been realized (see Figure 1(b)). The dimensions of the board and the number of holes and walls of the simulated game correspond to those of the real one. The walls are modeled as rigid bodies represented by boxes connected to the main board with fixed joints. The ODE-based simulation allows a realistic reproduction of the physical properties of the game. Collision-handling routine

was called at each step of the simulation using the Coulomb friction model. The knobs are simulated with motors.

2 Learning Architecture

Steering the ball through the whole Brio labyrinth is a very complex task. A human player solves it by breaking it down into several smaller problems. To learn the proper movements a hierarchical learning architecture based on QCON has been developed. The QCON is a connectionist Q-learning model proposed by Lin [1] where each action has a separate network. As observed in human players and following the divide and conqueror paradigm, the labyrinth was subdivided into small regions, where a QCON is assigned to each region (Figure 1b/2a).

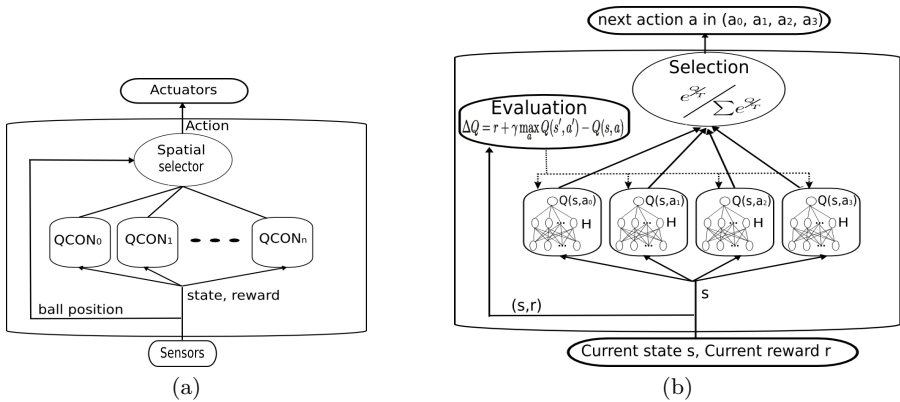


Fig. 2. (a) Illustration of the whole architecture. The current state and reward are inputs into the architecture. The output of the whole architecture is the output of an active QCON. (b) QCON architecture: Each action-value is represented by a feedforward network with one hidden layer that is trained using back-propagation algorithm and Q-learning with the parameters given in Table 1. We have four possible actions a_0, a_1, a_2, a_3 (see Table 1).

First the QCONs are trained separately on their respective subareas. In order to connect two subsequent areas, the subgoal of the first area is used as a starting region for the next one. In the play phase, based on the current position of the ball, a spatial selector module selects the appropriate learned QCON to be active and sends the output of the QCON to the actuators. This solution is inspired by "place cells" [2] found in the hippocampal brain region of rats. Place cells are found to be selectively active when the rat is situated in different locations while performing navigational tasks in a labyrinth environment [4]. The chosen approach has the following advantages: (1) It is easier to achieve a solution with an architecture composed of a committee of QCONs than a monolithic one. (2) The solution scales up easily as the complexity of the game increases. The

complexity of the architecture (the number of QCONs and the number of the hidden neurons in each QCON) is directly proportional to the complexity of the game (number of holes and walls).

3 Experiments

The parameters of the experimental setup are shown in Table 1. They are found empirically after performing various experiments on different subareas.

Table 1. Parameters of the experimental setup

| Factor | Description |
|-------------------|--|
| State | state $s=(x, y, V_x, V_y, P_1, P_2)$ (x,y) The ball position on the board (V_x, V_y) The ball velocity on the board (P_1, P_2) Motors position values |
| Action | For every motor there are two possible rotations: turn clockwise or turn anti-clockwise relative to the current position in steps of 0.1 deg; there are 4 possible actions $a_0 = (-0.1, 0.1), a_1 = (-0.1, -0.1), a_2 = (0.1, 0.1), a_3(-0.1, -0.1)$ |
| Reward | -0.5 if in hole; 1 if in subgoal; 0 otherwise |
| Learning | Discount factor $\gamma=0.8$; Learning rate $\alpha=0.2$ Number of hidden units in a QCON net $H=10$ |
| Actions selection | Stochastic: $\frac{e^{Q/T}}{\sum e^{Q/T}}$ Simulated annealing $T:1 \rightarrow 0.0002$ |
| Study | Average over 10 experiments in a single area; Play after after each 10 trials with greedy policy Maximum number of steps per play 600 |

For each study, where a QCON was trained in a given subarea, we performed 10 experiments. An experiment consisted of 300 trials, and after each tenth trial the agent played with the learned greedy policy. A trial begins with a random position and terminates when the ball falls in a hole, or when it attains the subgoal, or when the number of steps is greater than 600. We subdivided the labyrinth manually based on a predefined number of holes on a single subarea. This number is limited to two holes (see Figure 2b).

4 Results

The plots in Figure 3 show the results on the first two subareas. Two subtasks are solved by the agent. The first one is to avoid the holes which needed on average about 100 trials for both areas, and the second one is to attain a subgoal in a given subarea as fast as possible. The second subtask needed about 250 trials for both areas. We have found that the number of trials to learn the two subtasks

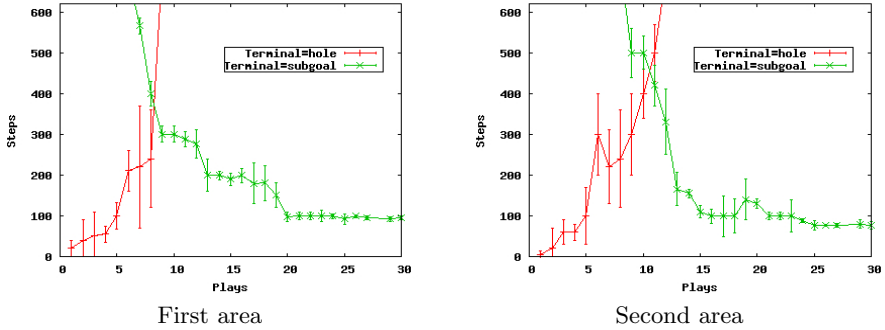


Fig. 3. Plots of the number of steps versus plays. One step corresponds to one action, and every play was performed after 10 trials using the learned greedy policy. Red curve (hole): number of steps needed before the ball falls in a hole. Green curve (subgoal): number of steps needed to reach the defined subgoal.

was similar for the other remaining subareas. Once trained, a QCON network does not need further adaptation when playing the game continually from the start point to the final goal in the whole labyrinth.

5 Summary and Outlook

We have presented a connectionist architecture for learning to play a simulated Brio labyrinth game that uses the divide and conquer paradigm inspired by the way a human player plays the game. We have shown that the architecture scales up easily as the number of subareas increases. In the future, we want to develop a way of automatically dividing the board into subareas. We also want to transfer and test the architecture on the real labyrinth and thereby improve its performance.

References

1. Lin, L.-J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning* 8(3-4), 293–321 (1992)
2. O’Keefe, J., Dostrovsky, J.: The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Research* 34(1), 171–175 (1971)
3. Smith, R.: Open dynamics engine (2005), <http://www.ode.org>
4. Wilson, M.A., McNaughton, B.L.: Dynamics of the hippocampal ensemble code for space. *Science* 261(5124), 1055–1058 (1993)

Divergence versus Convergence of Intelligent Systems: Contrasting Artificial Intelligence with Cognitive Psychology

Stefan Artmann

Friedrich-Schiller-University, Institute of Philosophy, 07737 Jena, Germany
stefan.artmann@uni-jena.de

1 Introduction

Artificial Intelligence (AI) and Cognitive Psychology (CP) are two sciences of intelligent systems that share many features. If we want, nevertheless, to contrast AI with CP, we must investigate differences between the strategies they follow in exploring intelligence. To do so, I transform the Turing Test into a more adequate intelligence test based on a necessary condition for intelligence, namely that intensions of second-order intentional predicates are observable in a system (sect. 2). I then contrast CP and AI by their criteria for progress in research on this necessary condition for intelligence (sect. 3).

2 Artificial Intelligence Face to Face with Human Intelligence

The conduct of a **Turing Test** (TT) minimally requires the following components: first, two devices that not only can encode written linguistic information and send it, but also can receive encoding signals and decode them into written linguistic information; second, a reliable bidirectional communication channel through which those devices are connected; third, a human that acts as an information source and destination at one of the devices; fourth, a digital computer that acts as an information source and destination at the other device; fifth, a sensory barrier that hinders the human to perceive that a digital computer is at the other end of the communication channel. The communication between the human and the computer obeys the following procedure: first, the human inputs a question in a predefined language into one of the devices; second, the question is encoded and transmitted through the communication channel; third, the computer's device receives the signals and decodes them into linguistic information; fourth, the computer inputs an answer into its device; fifth, the encoding of the answer is sent through the communication channel and decoded by the human's device. This exchange is to be repeated until, after a given time interval, the communication channel is blocked. Then the human must answer the following metaquestion about the conversation: 'Did you communicate with another human or with a computer?' The computer passes TT if the human answers that the interlocutor has been another human.

TT provides AI and CP with a sufficient condition for intelligence. Yet any science of intelligence needs a set of necessary conditions each of which implies an

observable criterion that, if satisfied, makes the presence of intelligence in a system more likely. To ascribe intelligence to a system, its internal organization has also to be taken into account. Externally observable behaviour shows only that a computer can use linguistic signs in a way that does not disappoint human expectations of how the computer relates communicated signs to denoted objects (extensions). What is needed is at least a test of whether the computer designates, by means of its internal information processing, intensions of linguistic signs. Thus, TT has to be transformed from an extensional into an intensional test, which I call **Simon and Newell Test** (SNT). A human “[...] can determine the computer’s exact intensions by examining the portions of its program and memory that incorporate its perceptual tests for discriminating among observed objects, processes, or relations [...]” [1] In SNT the discrimination tests of the computer refer, not to sensory data, but like in TT to signals that already have been decoded into linguistic information. Letting the computer undergo SNT requires that the computer must answer the human’s questions also by truthfully giving information about its own internal processing of these questions. The main difference between TT and SNT is that the computer not only inputs an answer into its device but also true information about how it has generated this answer. For humans, to give a reasonable answer to the metaquestion about the conversation: ‘Did you communicate with another human or with a computer?’ means therefore that they have to find out the procedures by which the computer tests whether linguistic signs received from its environment denote certain extensions. If humans think that these procedures are similar to those followed by themselves, they must answer that they have talked to another human. Then the computer passed SNT. SNT thus requires a formal schema for representing metainformation about information transmission, processing, and storage both in the messages the interlocutors exchange with each other and in the theory about their communication.

John McCarthy has proposed a criterion for the satisfaction of a necessary condition that a system must fulfil in order to be regarded as intelligent, namely the condition that a system thinks. This criterion concerns the semantic compressibility of metainformation about a machine: “Perhaps, a machine should be regarded as thinking if and only if its behavior can most concisely be described by statements, including those of the form ‘It now believes Pythagoras’ theorem.’” [2] To apply McCarthy’s criterion in SNT, we must answer the question of what schema is used for representing intentional predicates, which express a system’s epistemic qualification of the information it processes, as metainformation. Let s be a state of a system and p an information expressed by a sentence in a predefined language. Then the intentional predicate $B(s,p)$ means that, if the system is in state s , it believes p . For TT and SNT, it is necessary that at least the human can intensionally represent intentions of intentions. This requires second-order definitions of beliefs by second-order predicates $((W,M,B))$ asserting that a first-order predicate B is a useful notion of belief for the machine M in the world W . [3] If a system did not have second-order intentional predicates at its command, it could not process meta-epistemic qualifications of first-order intentional predicates.

An intelligent system must be able to intensionally represent other systems in its environment as possibly having intentions, so it must possess intensions of second-order intentional predicates to represent the intentions of other systems. I shall call an

SNT that ends with the following metaquestion **McCarthy Test (MT)**: ‘Did you communicate with another system that has intensions of second-order intentional predicates?’ In MT the human must decide whether the computer does ascribe intentions to the human. If so, the human has to consider the machine, not as another human, but as satisfying a necessary condition for being intelligent. In this case, the human comes to the result that the communication with the computer has been symmetrical in the following sense: both interlocutors must have used second-order intentional predicates for intensionally representing, e.g., beliefs about each other’s beliefs. MT thus implies a criterion for the satisfaction of a necessary condition that the computer must fulfil in order to be regarded as intelligent: there exists, after some questions of the human and answers of the computer, an invariant symmetry between the human and the computer in that they ascribe intensions of second-order intentional predicates to each other in a formal representational schema.

3 Artificial Intelligence Back to Back with Human Intelligence

I use the terms ‘convergence’ and ‘divergence’ to differentiate between two sciences of intelligence, and take them to mean ‘process of mimicking the behaviour and information processing of a paradigm more and more closely’ and ‘process of becoming in behaviour and information processing more and more dissimilar to a paradigm’, respectively. The progress fostered by research into intelligence may consist in the convergence of models of intelligence towards human intelligence: any artifact that implements such a model must then be compared, in its externally observable behaviour and its internally detectable information processing, to the gold standard of intelligence that is set by the biological paradigm of *homo sapiens*. This idea of progress is implied if a science of intelligence uses TT, since TT is nothing but a concrete operationalization of the convergence of intelligent artifacts towards human intelligence. SNT does not leave TT’s orientation to convergence behind, quite the contrary: the metaquestion that is put to the human after the conversation with the machine has ended addresses the overall similarity of the machine to the human. This implies that success in SNT requires of a computer to internally process information in a human-like fashion. SNT follows, thus, a methodology that is even more mimetic than TT – yet thanks to its integration of internal information processing into the communication between human and machine SNT also allows AI to follow another research strategy.

The scientific and technological progress as driven by another type of research into intelligence may consist in the divergence of human and non-human intelligence: here an artifact whose intelligence is tested has, neither in its externally observable behaviour nor in its internally detectable information processing, to measure up to *homo sapiens*. If, in this sense, all intelligent beings are created equal, the science that searches for the most divergent forms of intelligence is in need of formally expressible necessary conditions for general intelligence. One way of discovering such conditions is by thought experiments like MT. A science of divergent forms of intelligence needs manifold transformations of SNT that each take a necessary condition of intelligence as the subject of the question asked at the end of a test.

CP may be construed as aiming at an understanding of human intelligence so that CP must foster a convergence of intelligent artifacts towards the standard of human intelligence, and AI may be regarded as being indifferent towards the alternative of convergence and divergence. Yet to compare CP and AI on an equal footing it is more distinct to consider the divergence of intelligent artifacts from human intelligence as the idea of technological progress in basic AI research. Now we picture two disciplines, AI and CP, positioned at opposite ends of the spectrum of possible sciences of intelligence with regard to their acceptance of human behaviour and information processing as the paradigm of general intelligence. A science like CP is, thanks to its interest in a clearly demarcated class of empirical objects, quite near to traditional natural sciences. A science like AI is more inclined towards engineering and formal science, since it tries to systematically expand the class of actual objects whose imitation might have been the original stimulus for its research, into the class of all possible objects that satisfy some formal definition of intelligence. Sciences like AI (and Artificial Life) constitute a particular type of engineering that may be called modal engineering. **Modal engineers** construct artifacts that are equivalents of, or improvements on, fundamental functional properties of organisms, though they try to diverge as much as possible from the biochemical constitution and the anatomical organization of the latter. By so doing, modal engineering tries to define the minimal structures of intelligence, life, etc. From the perspective of common sense and empirical science, the realizations of these structures continue to be possible objects in the sense that they remain being imitations of actual paradigms.

Focussing on MT, an important task of AI is the modal engineering of second-order intentions. AI investigates possible systems that have intensions of second-order intentions by building artifacts that might diverge in their internal information processing from human beings to a very high degree. AI systematically tests the principal limits of how to construct such intensions by engineering empirically testable machines, and contributes thereby to a formal theory of necessary conditions of general intelligence. One method of doing AI research in this vein, the method of transforming TT into other experiments, uses metainformation that is communicated in well-defined test situations about the internal information processing of interlocutors to explore the range of possible systems satisfying a formal definition of intelligence. In MT, AI might try to stretch the communicational symmetry of both interlocutors as regards their use of second-order intentional predicates to its extreme, i.e., as far as it still lets the interlocutors recognize each other as intelligent beings.

References

1. Simon, H.A., Eisenstadt, S.A.: A Chinese Room that Understands. In: Preston, J., Bishop, M. (eds.) *Views into the Chinese Room*, pp. 95–108. Clarendon Press, Oxford (2002)
2. McCarthy, J.: The Inversion of Functions Defined by Turing Machines. In: Shannon, C.E., McCarthy, J. (eds.) *Automata Studies*, pp. 177–181. Princeton UP, Princeton/NJ (1956)
3. McCarthy, J.: Ascribing Mental Qualities to Machines. In: Ringle, M. (ed.) *Philosophical Perspectives in Artificial Intelligence*, pp. 161–195. Harvester Press, Brighton (1979)

Deep Inference for Automated Proof Tutoring?*

Christoph Benzmüller^{1,2}, Dominik Dietrich¹, Marvin Schiller¹, and Serge Autexier^{1,3}

¹ Dept. of Computer Science, Saarland University, 66041 Saarbrücken, Germany

² Computer Laboratory, The University of Cambridge, Cambridge, CB3 0FD, UK

³ German Research Centre for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany

1 Introduction

Ω MEGA [7], a mathematical assistant environment comprising an interactive proof assistant, a proof planner, a structured knowledge base, a graphical user interface, access to external reasoners, etc., is being developed since the early 90's at Saarland University. Similar to HOL4, Isabelle/HOL, Coq, or Mizar, the overall goal of the project is to develop a system platform for formal methods (not only) in mathematics and computer science. In Ω MEGA, user and system interact in order to produce verifiable and trusted proofs. By continuously improving (not only) automation and interaction support in the system we want to ease the usually very tedious formalization and proving task for the user.

A very recent application direction of Ω MEGA, studied in the DIALOG project [3], is e-learning in mathematics. The hypothesis is that our system can fruitfully support the tutoring of mathematical proofs.

In 2001 our group opted for a major reimplementaion of the Ω MEGA system. A major motivation was to replace the natural deduction (ND) calculus at the logical core of the system by another, ideally more suitable, logical base formalism. What exactly did we have in mind?

1. An earlier study (cf. [5]) of the influence of the Ω MEGA's ND core to its abstract level reasoning mechanism, such as proof planning, revealed a negative influence mainly due to unnatural, structural peculiarities of ND. We conjectured that a less 'low-level' logical core should remedy this problem.
2. Abstract level proofs in Ω MEGA may be unsound. They have to be expanded at logical core level before they can be verified and finally trusted. Expansions to ND level, however, often dramatically increase the proof size and they typically lead to very unnatural and human incomprehensible proofs. Supporting the tutoring of proofs based on these unnatural, low-level proofs thus seemed a suboptimal starting point for our DIALOG project.

For our reimplementaion we therefore picked Serge Autexier's CORE calculus [1] as a promising alternative to ND. CORE has then been transformed into an 'assertion level' inference mechanism [11] which constitutes the logical kernel of the new Ω MEGA^{CORE} system. CORE and our assertion level inference mechanism are (higher-order) variants of the deep inference approach [8], that is, they support deductions deeply inside a given

* Funded by the DFG SFB378 (Project OMEGA) and by EPSRC under grant EP/D070511/1.

¹ <http://alessio.guglielmi.name/res/cos/index.html>

| | |
|--|--|
| <p><i>s1: Let $(x, y) \in (R \circ S)^{-1}$.</i></p> <p><i>t1: Good. correct</i></p> <p><i>s2: It follows that $(y, x) \in (R \circ S)$.</i></p> <p><i>t2: That's correct. correct</i></p> <p><i>s3: Therefore $\exists z \in M : (y, z) \in R \wedge (z, x) \in S$.</i></p> <p><i>t3: Yes, that's right. correct</i></p> <p><i>s4: This implies: $\exists z \in M : (z, y) \in R^{-1} \wedge (x, z) \in S^{-1}$</i></p> <p><i>t4: Your deduction is correct. You are on a good way. correct</i></p> <p><i>s5: It follows: $(x, y) \in S^{-1} \circ R^{-1}$.</i></p> <p><i>t5: Yes. Therefore it is shown that every element from $(R \circ S)^{-1}$ already lies in $S^{-1} \circ R^{-1}$. correct</i></p> | <p><i>s6: Now let $(x, y) \in S^{-1} \circ R^{-1}$.</i></p> <p><i>t6: O.k. correct</i></p> <p><i>s7: $\Rightarrow \exists z \in M : (x, z) \in S^{-1} \wedge (z, y) \in R^{-1}$.</i></p> <p><i>t7: Yes. correct</i></p> <p><i>s8: $\Rightarrow \exists z \in M : (z, x) \in S \wedge (y, z) \in R$.</i></p> <p><i>t8: This deduction is also correct. correct</i></p> <p><i>s9: $\Rightarrow (y, x) \in R \circ S$.</i></p> <p><i>t9: This deduction is again correct. correct</i></p> <p><i>s10: $\Rightarrow (x, y) \in (S \circ R)^{-1}$.</i></p> <p><i>t10: Congratulations! With this you have shown both inclusions. Your solution is now complete. correct</i></p> |
|--|--|

Fig. 1. Example dialog; $s_..$ are student turns and $t_..$ are tutor turns

formula without requiring preceding structural decompositions as needed in ND (or sequent calculus). In $\Omega\text{MEGA}^{\text{CORE}}$ we thus have a smaller ‘distance’ between abstract level proofs and their expansions to the verifiable assertion level. Most importantly, we now support reasoning directly at the assertion level, while such a layer did only exist in the old ΩMEGA for *a posteriori* proof presentation purposes.

In this short paper we report on our ongoing application and evaluation of the $\Omega\text{MEGA}^{\text{CORE}}$ -system for proof tutoring in the DIALOG project. For this, we apply our novel proof assessment module [6] developed in the DIALOG project to 17 proof dialogs which we have obtained in a previous experiment [4]. We study the ‘quality’ of the automatically reconstructed proofs and analyse the coverage of our proof assessment module.

2 Evaluation

We have applied our proof assessment module to 17 tutorial dialogs taken from the Wizard-of-Oz experiment reported in [4]. These dialogs consist in alternating utterances by a student and a tutor. The student attempts to solve an exercise from the domain of binary relations, namely to show that $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$ holds for two relations R and S in a set M , where \circ denotes relation composition and $^{-1}$ denotes inversion. The tutor responds to the subsequent proof step utterances of the student. Each step is annotated with a judgment regarding its correctness by the tutor. One example dialog from our evaluation set is shown in Figure 1.

The idea of the proof assessment module is to support automated proof tutoring, that is, to automatically judge about the correctness (and also the granularity and the relevance – not discussed in this paper) of each single student proof step (cf. [2]). For this, the assessment module is initialized with the relevant axioms for this domain, which are automatically transformed into inference rules at the assertion level (cf. [8]) by $\Omega\text{MEGA}^{\text{CORE}}$ and made available for proving.

$$\begin{array}{c}
\frac{}{s5: (x,y) \in s^{-1} \circ r^{-1} \vdash \text{---}} \text{Close} \\
\frac{s4: (z,y) \in r^{-1} \wedge (x,z) \in s^{-1} \vdash \text{---}}{(y,z) \in r \wedge (x,z) \in s^{-1} \vdash \text{---}} \text{Def.}^{-1} \\
\frac{s3: (y,z) \in r \wedge (z,x) \in s \vdash \text{---}}{(x,y) \in (r \circ s) \vdash \text{---}} \text{Def.}^{\circ} \\
\frac{s1: (x,y) \in (r \circ s)^{-1} \vdash (x,y) \in s^{-1} \circ r^{-1}}{\vdash (r \circ s)^{-1} \subseteq s^{-1} \circ r^{-1}} \text{Def.}^{\subseteq} \\
\hline
t10: \vdash (r \circ s)^{-1} = s^{-1} \circ r^{-1}
\end{array}
\qquad
\begin{array}{c}
\frac{}{s10: (x,y) \in (r \circ s)^{-1} \vdash \text{---}} \text{Close} \\
\frac{s9: (y,x) \in (r \circ s) \vdash \text{---}}{(z,x) \in s \wedge (y,z) \in r \vdash \text{---}} \text{Def.}^{\circ} \\
\frac{s8: (z,x) \in s \wedge (y,z) \in r \vdash \text{---}}{(x,z) \in s^{-1} \wedge (y,z) \in r \vdash \text{---}} \text{Def.}^{-1} \\
\frac{s7: (x,z) \in s^{-1} \wedge (z,y) \in r^{-1} \vdash \text{---}}{(x,y) \in s^{-1} \circ r^{-1} \vdash (x,y) \in (r \circ s)^{-1}} \text{Def.}^{\circ} \\
\frac{s6: (x,y) \in s^{-1} \circ r^{-1} \vdash (x,y) \in (r \circ s)^{-1}}{\vdash s^{-1} \circ r^{-1} \subseteq (r \circ s)^{-1}} \text{Def.}^{\subseteq} \\
\hline
\text{Def.}^{\text{=}}
\end{array}$$

Fig. 2. Annotated $\Omega\text{MEGA}^{\text{CORE}}$ assertion level proof for the example dialog

Then, for each of the 17 dialogs, the assessment was performed stepwise by our assessment module. The assessment module maintains an assertion level proof object that represents the current state of the proof under construction, which can include several proof alternatives in the case of underspecified, that is, insufficiently precise, proof step utterances by the student causing ambiguities (cf. [216]). For each proof step uttered by the student, the module uses a *depth-limited breadth-first search* (with pruning of superfluous branches) to expand the given proof state to all possible successor states up to that depth. From these, those successor states that match the given utterance wrt. to some filter function (analyzing whether a successor state is a possible reading of the student proof step) are selected. We thus obtain, modulo our filter function, assertion level counterparts to all possible interpretations of correct student proof steps. If for a given utterance, no matching successor state can be reached, the utterance is considered as incorrect.

We compared the results of the automated proof step analysis with the original correctness judgments by the tutors. All steps in the example dialog are correctly classified as valid by our assessment module (used with proof depth four), taking approximately 13.2 seconds on a standard PC.

Figure 2 shows one complete assertion level proof (in sequent notation and annotated by the corresponding student proof steps) that was constructed by the assessment module for the dialog in Figure 1. The number of assertion level steps required (13, excluding the automatic *Close* steps) is still comparable to the number of proof steps as uttered by the student in the original dialog (10), which provides evidence that the $\Omega\text{MEGA}^{\text{CORE}}$ assertion level proof is at a suitable level of granularity. Had we used natural deduction as in the old ΩMEGA system, we would have obtained many intermediate steps of rather technical nature making breadth-first proof search for our task infeasible, compare:

$$\begin{array}{c}
\frac{}{A := (z,y) \in r^{-1} \wedge (x,z) \in s^{-1}} \\
\frac{(y,z) \in r^{-1}}{(y,z) \in r} \text{Def.}^{-1} \\
\frac{}{(z,y) \in r^{-1} \wedge (x,z) \in s^{-1}} \text{Def.}^{-1} \\
\hline
\text{Core}
\end{array}
\qquad
\begin{array}{c}
\frac{}{A} \wedge E \\
\frac{(y,z) \in r^{-1}}{(y,z) \in r} \text{Def.}^{-1} \\
\frac{}{(x,z) \in s^{-1}} \wedge I \\
\frac{}{(y,z) \in r \wedge (x,z) \in s^{-1}} \wedge I
\end{array}$$

Core

Natural Deduction

The 17 dialogs in the evaluation contain a total of 147 proof steps. All the steps within a dialog are passed to the assessment module sequentially until a step that is labeled as correct cannot be verified, in which case we move on to the next dialog. This way, we correctly classify 141 out of the 147 steps (95.9%) as correct or wrong. Among the remaining six steps are three where the verification fails, and further three remain untouched.

3 Concluding Remarks

Our initial question whether moving from Ω MEGA's previous ND based logical core to assertion level reasoning in Ω MEGA^{CORE} was a reasonable decision, can (preliminarily) be answered with 'Yes':

In Ω MEGA^{CORE} we obtain more adequate formal counterparts of the human proofs as was possible before. Most importantly, we directly search for these proofs at the assertion level which enables us to employ a simple depth-limited breadth-first search algorithm in our proof step assessment module. Interestingly, already a depth limit of just four assertion level steps enables our approach to correctly classify 95.9% of the proof steps in our corpus.

Related to our work are the EPGY Theorem Proving Environment [9], using Otter to justify or reject proof steps proposed to the environment, and the computational framework by Claus Zinn [10] for the analysis of textbook proofs. Our approach differs in the following ways: (i) We address the problem of underspecification and multiple interpretations of student proof step utterances, (ii) we construct one, or several, global, coherent proof object(s) for each dialog instead of just looking from step to step, (iii) we are not just interested in the correctness of proof steps but also in their granularity and relevance; for this adequate formal proofs are even more important.

References

1. Autexier, S.: The core calculus. In: Nieuwenhuis, R. (ed.) Automated Deduction – CADE-20. LNCS (LNAI), vol. 3632, Springer, Heidelberg (2005)
2. Benzmüller, C., Vo, Q.B.: Mathematical domain reasoning tasks in natural language tutorial dialog on proofs. In: Proc. AAI-05, AAAI Press/The MIT Press (2005)
3. Benzmüller, C., et al.: Natural language dialog with a tutor system for mathematical proofs. In: Ullrich, C., Siekmann, J.H., Lu, R. (eds.) Cognitive Systems. LNCS (LNAI), vol. 4429, Springer, Heidelberg (2007)
4. Benzmüller, C., et al.: Diawoz-II - a tool for wizard-of-oz experiments in mathematics. In: Freksa, C., Kohlhase, M., Schill, K. (eds.) KI 2006. LNCS (LNAI), vol. 4314, Springer, Heidelberg (2007)
5. Benzmüller, C., et al.: Proof planning: A fresh start? In: Proc. of IJCAR 2001 Workshop: Future Directions in Automated Reasoning, Siena, Italy (2001)
6. Dietrich, D., Buckley, M.: Verification of Proof Steps for Tutoring Mathematical Proofs. In: Proc. AIED 2007 (to appear)
7. Siekmann, J., et al.: Computer supported mathematics with omega. J. Applied Logic 4(4), 533–559 (2006)

8. Autexier, S., Dietrich, D.: Synthesizing Proof Planning Methods and Oants Agents from Mathematical Knowledge. In: Borwein, J.M., Farmer, W.M. (eds.) MKM 2006. LNCS (LNAI), vol. 4108, Springer, Heidelberg (2006)
9. McMath, D., Rozenfeld, M., Sommer, R.: A computer environment for writing ordinary mathematical proofs. In: Nieuwenhuis, R., Voronkov, A. (eds.) LPAR 2001. LNCS (LNAI), vol. 2250, pp. 507–516. Springer, Heidelberg (2001)
10. Zinn, C.: A computational framework for understanding mathematical discourse. *Logic J. of the IGPL* 11, 457–484 (2003)
11. Dietrich, D.: The tasklayer of the omega system. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany (2006)

Exploiting Past Experience – Case-Based Decision Support for Soccer Agents

– Extended Abstract –

Ralf Berger and Gregor Lämmel

Humboldt University Berlin, Department of Computer Science
berger@informatik.hu-berlin.de, laemmel@vsp.tu-berlin.de
<http://www.robocup.de/AT-Humboldt>

Abstract. Selecting and initiating an appropriate (possibly cooperative) behavior in a given context is one of the most important and difficult tasks for soccer playing robots or software agents. Of course, this applies to other complex robot environments as well.

In this paper we present a methodology for using Case Based Reasoning techniques for this challenging problem. We will show a complete workflow from case-acquisition up to case-base maintenance. Our system uses several techniques for optimizing the case base and the retrieval step in order to be efficient enough to use it in a realtime environment.

The framework we propose could successfully be tested within the robot soccer domain where it was able to select and initiate complex game plays by using experience from previous situations. Due to space constraints we can give just a very brief overview about the most important aspects of our system here.

1 Introduction

Selecting and initiating an appropriate (long term and possibly cooperative) behavior in a given context is one of the most important tasks for autonomous robots in complex and dynamic environments. Various methods have been developed in order to determine the best action or the best behavior (in terms of action sequences).

Case Based Reasoning (CBR) is a method of problem solving and learning based on the principle of conclusion by analogy. In simple terms it means using old experiences to understand and solve new problems - a reasoner remembers a past problem/situation similar to the current one and uses this to solve the new problem.

The goal of the system we are about to introduce is to support the decision making processes of our soccer playing robots / agents with the help of experience gained by already played games. We found this particular useful on a higher behavior selection level, where the designer wants to influence the decision making processes in an easy and symbolical way. Our test scenario is the *wall pass*¹

¹ Player one passes the ball to player two, who immediately passes it back to player one. The idea is to use the ball as a distraction for the opponent team to allow the passer to move into a position of advantage in order to receive the ball again.

in RoboCup [1] – a typical coordination problem where a cooperative behavior has to be initialized and controlled only by individual cognition and reasoning. Although the *wall pass* is stated the simplest combination play in soccer, it is hard to achieve intentionally for robots, especially without using communication.

Case Based Reasoning has been used in RoboCup for a long time and for very different purposes. [2] gives a very broad overview about what has been achieved so far in this field. An approach that is quite similar to ours in some aspects comes from Ros [3]. The major differences are that our system addresses the whole CBR-workflow and could already provide first results in a competitive environment.

As we already stated, we can only pick some interesting points from our work here, namely the topics of case acquisition, retrieval, case base optimization and maintenance.

2 Building Up a Case Base

The selection of a suitable case format is of vital importance for the Case Based Reasoning system. Basically the case format dictates by which features the similarity between a case from the case-base and a query situation is determined. Several spatial features are possible, like positions, velocities as well as game-based features as current score or remaining playing time. In the underlying domain, a snapshot of a situation seems to be sufficient to determine, whether a *wall pass* is possible or not. We think the possibility and utility of performing a *wall pass* most of all depends on the spatial relation between the involved players (attackers and defenders). Information about velocities are neglected here, especially since the perception of other's velocities is quite unreliable. Finally we define the similarity between cases / situations to be only dependant on the similarity of positional features. Thus, a case contains, besides the information about the class (*wall pass* possible/ impossible), the information about the players' positions on the field as quantified Euclidean coordinates.

For every CBR system the question comes up how to acquire cases for a first case base. Our test domain (RoboCup Simulation League) provides the exceptional opportunity to access a huge repository of logfiles of already played games. We exploit this pool of experience by building up our initial case-base from these matches. Our goal is, that after the primary case acquisition, the agents start to advance and extend their case-bases by own experience. First, we have experimented with a fully automated case extraction. But because of the minor number of already played *wall passes* the automatic case acquisition was not suitable to built up the initial case base².

To get an initial case-base under these circumstances, we decided to extract all 'potential' *wall pass* situations automatically and to classify the situation manually afterwards. The thusly built-up case-base contains so far 1010 cases (560 cases belong to the class ***wall pass possible***, 450 cases belong to the class ***wall pass impossible***).

² However, after more teams are able to perform such game-plays intentionally, the automatic case acquisition could be again an option to acquire additional cases.

3 Retrieval

There are two key issues for the retrieval mechanism. Firstly, it should find the most similar case out of the case-base for a given query situation. Secondly, the retrieval needs to be fast, especially in the underlying domain. In spatial domains it is common to use a distance function to describe the relation between similarity and distance. In many approaches the similarity decreases with the distance according to a Gaussian function (e.g. [4]). However, in this approach the similarity is rather defined by degree of match of the quantified spatial features, then of the (spatial) distance. That means, a feature based retrieval mechanism was required. A Case Retrieval Net (CRN) [5] excellently meets those requirements. It provides an efficient and flexible way of retrieving a relatively small number of relevant cases out of a possibly huge case base. Especially, the avoidance of exhaustive memory search speeds up the retrieval. We not only show that the CRN work with a high grade of accuracy, but we also show that this kind of retrieval mechanism is very fast³.

4 Case Base Optimization and Maintenance

It is obvious that the runtime performance of the retrieval mechanism does not only depend on the retrieval mechanisms itself, but also on the amount of information that a case in the case base describes. There are two opposed needs, on the one hand we want to reduce the information that a case describes (i.e. deletion of irrelevant players out of the cases). On the other hand we must keep the cases unambiguous, otherwise a case based retrieval is not applicable. In the underlying domain it is easy to see, that only a few players have an influence on the success of a *wall pass*. If a player can take influence depends on its spatial relation to the directly involved players. We developed a simple procedure to extract the relevant players. We could show that mostly 3-4 players are sufficient to describe a case sufficiently.

Using the procedure in [2] it's obvious that the initial case base contains some redundant cases. Thus we have to find a way to find and delete redundant cases. Competence models provide a way to identify redundant parts of knowledge-bases [6]. Smyth and McKenna introduced a competence model that select redundant cases based on their individual competence contribution [7]. We show that this model outperforms other (competence) models in the number of deletable cases without a decrease of competence.

Maintenance in the context of CBR usually denotes the enduring adaptation, refinement and optimization of the system (mainly the case-base), as well as remedying deficiencies, in order to ensure or improve the usability and applicability of the CBR-system. It is substantial for the success of CBR in soccer programs, since the overall behavior of the teams or individual skills are subject to continuous changes. However, maintenance was often ignored for a long time

³ For a given query situation, the retrieval of the most similar case takes not even 1 ms, although the case base is built-up of much more than 1000 cases.

§, also in RoboCup. We have developed a framework and a tool-chain for offline maintenance of our case bases. It handles the automatic acquisition of new cases from own experience, the monitoring of consistency and redundancy within the case-base, analysis and optimization based on use- and success-histories to name only the most important things.

5 Conclusion

We have developed a methodology for using Case Based Reasoning for high-level decision making in the robot soccer domain. Our work encompasses the complete workflow of finding an appropriate case-format, the case-acquisition, defining similarity measures and an efficient retrieval, as well as providing optimization and maintenance tools. We have successfully tested the system in the RoboCup Soccer Simulation League with a first cooperative game-play (*wall pass*), where it showed very promising results. Right now we are working on applying it to other cooperative combination plays, e.g. free-kicks.

References

1. The RoboCup Federation: Official Website URL: <http://www.robocup.org>
2. Burkhard, H.D., Berger, R.: Cases in Robotic Soccer. In: 7th International Conference on Case-Based Reasoning (ICCBR'07) (2007)
3. Ros, R., Arcos, J.L.: Acquiring a robust case base for the robot soccer domain. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007) (2007)
4. Ros, R., Veloso, M., de Mantaras, R.L., Sierra, C., Arcos, J.: Retrieving and reusing game plays for robot soccer. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, Springer, Heidelberg (2006)
5. Burkhard, H.D.: Extending some Concepts of CBR – Foundations of Case Retrieval Nets. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 17–50. Springer, Heidelberg (1998)
6. Smyth, B., Kean, M.T.: Remembering to forget: A competence preserving deletion policy for case-based reasoning systems. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 399–382. Morgan Kaufmann, San Francisco (1995)
7. Smyth, B., McKenna, E.: Building compact competent case-bases. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) Case-Based Reasoning Research and Development. LNCS (LNAI), vol. 1650, p. 329. Springer, Heidelberg (1999)
8. Smyth, B.: Case-base maintenance. In: IEA/AIE '98: Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, London, UK, Springer, Heidelberg (1998)

Externalizing the Multiple Sequence Alignment Problem with Affine Gap Costs*

Stefan Edelkamp and Peter Kissmann

Computer Science Department
University of Dortmund

Abstract. Multiple sequence alignment (MSA) is a problem in computational biology with the goal to discover similarities between DNA or protein sequences. One problem in larger instances is that the search exhausts main memory. This paper applies disk-based heuristic search to solve MSA benchmarks. We extend iterative-deepening dynamic programming, a hybrid of dynamic programming and IDA*, for which optimal alignments with respect to similarity metrics and affine gap cost are computed. We achieve considerable savings of main memory with an acceptable time overhead. By scaling buffer sizes, the space-time trade-off can be adapted to existing resources.

1 Introduction

When designing a cost function for MSA, computational efficiency and biological meaning have to be taken into account. Altschul [1] argues that at least a character-pair scoring matrix and affine gap costs have to be included in the most widely-used function *sum-of-pairs*. Affine gap costs induce a linear function $a + bx$, where x is the size of the gap, a is the cost for gap opening and b is the cost for extending the gap. Use of an affine gap cost in multiple sequence alignment is a challenge because identifying the opening of a gap is challenging. In terms of [2] the gap costs we consider are *quasi-natural*. It is the cost model used in practice by biologists and their alignment programs [3].

There is a host of algorithms that has been applied to solve the MSA problem. In contrast to its name, dynamic programming is a static traversal scheme, traversing the problem graph in a fixed order. The storage requirements are considerable, all reachable nodes are visited. Given k sequences of maximal length n this accumulates to $O(n^k)$ nodes and $O(2^k \cdot n^k)$ edge visits.

Hirschberg [2] proposes a strategy that stores only the search frontier and reconstructs the solution path in divide-and-conquer manner. The reduction of the search frontier has inspired most of the upcoming algorithms. Frontier search [4] combines A* with Hirschberg's approach to omit already expanded states from the search. Sparse-memory graph search [8] stores some of the already expanded states to speed-up the computation. Compared to frontier search it describes an

* The work is supported by DFG in the projects ED-74/3 and ED-74/4.

alternative scheme of dealing with back leaks. Sweep-A* [9] is the MSA adaption of breadth-first heuristic search [10].

Iterative-deepening dynamic programming [6], IDDP for short, is a hybrid of dynamic programming and IDA*. A difference to the above approaches is that not the nodes but the edges are expanded, with extra start edge s_e and target edge t_e . IDDP carries out a series of searches with successively larger thresholds. The estimate for the path from s_e to t_e via the current edge e is given by $f(e) = g(e) + h(e)$. The traversal order is along increasing levels, such that all states that do not lie on a shortest path can be removed. Additionally, path compression algorithms can be applied to store less states in main memory. By the same argument as in IDA*, IDDP will find an optimal solution. IDDP shares similarities with iterative-deepening bounded dynamic programming as introduced by [4]. The use of a global upper bound U additionally saves memory as it prunes generated edges whose f -value is outside the current threshold.

2 External IDDP

Externalization considers maintaining data structures on (one or several) hard disks by the application program. In External IDDP [3], all file access is buffered. For the expanded layer, we need one read buffer for the nodes with edges that have not yet been expanded, and one write buffer for the nodes with edges that have been expanded. Additionally, we need n write buffers for placing successors in one of the next n layers. An advantage is that the memory requirements can be adapted to match the existing hardware.

Nodes are sorted with respect to their coordinates and read, if an outgoing edge is expanded. For affine gap costs we further have to check, whether a gap is continued or opened. Moreover, we do no longer support the internal compression of the list of already expanded nodes through deletion of states, as the buffers already reduce the memory for each layer considerably, and further backup data is flushed to the disk. This implies that each coordinate between two nodes v and v' can differ by either 0 or 1, such that the coordinate difference between v and v' can be encoded by the integer $diff(v, v') = (v'_1 - v_1)2^{k-1} + \dots + (v'_k - v_k)2^0$. Using the reversed encoding given v' and $diff(v, v')$ we can reconstruct v .

External algorithms are measured in the number of file accesses (I/Os) and the number of times they scan or sort N data items $scan(N)$ and $sort(N)$, respectively. We denote the number of expanded nodes by $|V_{Exp}| = |\{n' \in V \mid e = (n, n') \wedge f(e) \leq f^*\}|$ and the number of expanded edges by $|E_{Exp}| = |\{e \in E \mid f(e) \leq f^*\}|$ where f^* is the optimal solution path cost. Let $L = h(s_e)$ be the initial threshold and U be the global upper bound. The optimality of IDDP is inherited from IDA* and dynamic programming, provided that U is correct. It is also simple to see that the last iteration is actually the largest, since each iteration contains at least one more edge than the previous one.

As the length of any alignment is bounded by nk the number of iterations is polynomial in n , k and the maximal edge cost. The last iteration applies $O(sort(|E_{Exp}|) + scan(|V_{Exp}|))$ I/Os. To perform delayed duplicate detection we

Table 1. Comparing IDDP with Ex-IDDP (time in *hh:mm:ss* and space in kilobytes)

| | k | Cost | $h(s_e)$ | Time | RAM | Time | RAM | DISK |
|-------|-----|---------|----------|----------|------------|-----------|---------|-----------|
| 1lcf | 6 | 134,097 | 133,540 | 5:30:22 | 219,824 | 6:40:23 | 106,772 | 437,888 |
| 1rthA | 5 | 70,387 | 70,243 | 0:00:20 | 11,216 | 0:00:43 | 13,684 | 1,365 |
| 1taq | 5 | 119,552 | 119,160 | 12:02:26 | 678,020 | 60:03:41 | 129,356 | 3,012,294 |
| 1ac5 | 4 | 39,675 | 39,515 | 0:03:13 | 44,352 | 0:03:34 | 40,276 | 4,787 |
| 1bgl | 4 | 80,552 | 80,406 | 0:03:45 | 42,380 | 0:03:31 | 48,888 | 8,860 |
| 1dlc | 4 | 49,276 | 49,141 | 0:02:48 | 29,724 | 0:03:29 | 36,992 | 2,044 |
| 1eft | 4 | 33,151 | 33,053 | 0:00:39 | 12,060 | 0:00:44 | 11,184 | 1,262 |
| 1gowA | 4 | 40,727 | 40,577 | 0:03:09 | 22,348 | 0:03:41 | 30,896 | 1,246 |
| 2ack | 5 | 69,608 | 69,060 | 3:39:14 | 419,944 | 4:59:43 | 264,240 | 363,705 |
| arp | 5 | 58,300 | 57,865 | 1:12:22 | 91,636 | 1:26:30 | 69,300 | 182,314 |
| glg | 5 | 66,606 | 66,408 | 0:06:22 | 44,644 | 0:07:01 | 51,940 | 3,332 |
| 1ajsA | 4 | 34,501 | 34,277 | 0:09:57 | 66,964 | 0:10:28 | 60,148 | 19,148 |
| 1cpt | 4 | 36,612 | 36,414 | 0:02:54 | 38,784 | 0:03:10 | 42,220 | 6,094 |
| 1lvl | 4 | 39,849 | 39,602 | 0:20:18 | 194,432 | 0:20:11 | 179,744 | 42,567 |
| 1ped | 3 | 16,333 | 16,170 | 0:00:05 | 11,244 | 0:00:22 | 255 | 3,311 |
| 2myr | 4 | 41,281 | 40,935 | 5:48:18 | 937,612 | 12:05:19 | 722,324 | 533,749 |
| gal4 | 5 | 57,286 | 56,632 | - | >1,048,576 | 182:55:51 | 580,008 | 7,354,524 |

sort the layers with respect to the nodes. The number of nodes in the next layer is bounded by the number of edges from the nodes in the previous layer. Therefore, the cumulated sorting efforts for removing duplicates in the individual layers are less than the sorting efforts for the entire set E_{Exp} . For reading a layer and for solution reconstruction at most $scan(|V_{Exp}|)$ I/Os are needed. Therefore, given that there are at most $U - L + 1$ iterations, the overall run time is bounded by $O((U - L) \cdot (sort(|E_{Exp}|) + scan(|V_{Exp}|)))$ I/Os. Factor $U - L$ can be avoided by using a strategy called refined threshold determination [7].

3 Experimental Results

We experimented on a 64-bit Opteron 2.2GHz Linux machine with 1 gigabyte memory and a total runtime limit of 480 hours. Table 1 displays the cost-optimal solutions obtained with internal and external IDDP on the hardest sequences of BALiBASE. We denote the number of sequences to be aligned, the initial and optimal cost, as well as the resource consumption of the exploration. We see that there are considerable RAM savings while the time increase remains moderate and that the ratio of disk space and RAM usage can be large. Up to the storage structures for the estimate, the RAM requirements remain constant.

On a 32-bit system IDDP consumed 140 kilobytes to solve 1lcf, while for External IDDP only 79 kilobytes were used. As the heuristic calls IDDP for a smaller set of sequences (3 in case of the triple heuristic), it is also possible to externalize its calculation. For solving 1lcf with External IDDP, the time increased from 6:40:23 for the internal heuristic to 18:18:54 for the external one.

Table 2. Comparison with other quasi-natural gap cost sequence alignment solvers

| | OMA | | 5-Group A* | | IDDP | External IDDP | |
|-------|--------|-----|------------|-----|--------|---------------|--------|
| TaboA | 10,674 | 973 | 10,674 | 199 | 10,665 | 8 | 15 |
| laho | 9,807 | 6 | 9,807 | 0 | 8,828 | 0 | 8,828 |
| 1hfh | 19,208 | 23 | 19,208 | 3 | 17,628 | 2 | 17,628 |
| lidy | 9,542 | 3 | 9,508 | 45 | 8,637 | 3 | 8,637 |
| 1pfc | 17,708 | 19 | 17,708 | 3 | 15,843 | 0 | 15,774 |
| 1plc | 14,205 | 4 | 14,195 | 0 | 12,745 | 0 | 12,745 |
| 2mhr | 16,687 | 4 | 16,687 | 0 | 14,765 | 0 | 14,765 |
| 451c | 13,364 | 200 | 13,364 | 74 | 12,171 | 1 | 12,171 |

Compared to the literature, the cost function used in [4] neither uses similarity measures nor affine gap costs. Niewiadomski [5] showed good results in solving BALiBASE alignment problems with internal parallel frontier search using similarity matrices with fixed gap cost. The peak RAM requirements for solving 1pamA were 55.8 gigabytes. K-group A* [11] extends Sweep-A* [9] and uses quasi-natural gap costs (but different metrics). Table 2 shows that (External) IDDP is competitive (cost in unit, time in seconds).

Besides parallelization, in the future we will look at variants that adapt to the time one wants to spend on computing the alignment.

References

1. Altschul, S.: Gap costs for multiple sequence alignment. *Journal of Theoretical Biology* 138, 297–309 (1989)
2. Hirschberg, D.S.: A linear space algorithm for computing common subsequences. *Communications of the ACM* 18(6), 341–343 (1975)
3. Kissmann, P.: Externalisierung des Sequenzalignierungsproblems. Diploma Thesis, University of Dortmund (January 2007)
4. Korf, R.E., Zhang, W., Thayer, I., Hohwald, H.: Frontier search. *Journal of the ACM* 52(5), 715–748 (2005)
5. Niewiadomski, R., Amaral, J.N., Holte, R.C.: Sequential and parallel algorithms for frontier A* with delayed duplicate detection. In: *AAAI* (2006)
6. Schroedl, S.: An improved search algorithm for optimal multiple sequence alignment. *Journal of Artificial Intelligence Research* 23, 587–623 (2005)
7. Wah, B.W., Shang, Y.: A comparison of a class of IDA* search algorithms. *International Journal of Tools with Artificial Intelligence* 3(4), 493–523 (1995)
8. Zhou, R., Hansen, E.: Sparse-memory graph search. In: *IJCAI*, pp. 1259–1268 (2003)
9. Zhou, R., Hansen, E.: Sweep A*: Space-efficient heuristic search in partially-ordered graphs. In: *ICTAI*, pp. 427–434 (2003)
10. Zhou, R., Hansen, E.: Breadth-first heuristic search. In: *ICAPS*, pp. 92–100 (2004)
11. Zhou, R., Hansen, E.: K-Group A* for multiple sequence alignment with quasi-natural gap costs. In: *ICTAI*, pp. 688–695 (2004)

Text Generation in the SmartWeb Multimodal Dialogue System

Ralf Engel and Daniel Sonntag

DFKI GmbH, Stuhlsatzenhausweg 3, 66123 Saarbrücken
{ralf.engel,daniel.sonntag}@dfki.de

Abstract. This paper presents the text generation module of SMARTWEB, a multimodal dialogue system. The generation module bases on NipsGen which combines SPIN, originally a parser developed for spoken language, and a tree-adjointing grammar framework for German. NipsGen allows to mix full generation with canned text.

Introduction. This paper presents how the generation of text is handled in the SMARTWEB system^[1] [1], a multimodal dialogue system to access semantic databases and Web Services using a smartphone. For this task, we use the NipsGen module which combines two already existing components: (1) the SPIN parser [2] which was originally designed for natural language understanding and (2) a tree-adjointing grammar (TAG) framework for German [3]. SPIN is basically a rewriting system for typed feature structures (TFSS) with a powerful rule language. Instead of TFSS representing the recognized words, the object which should be verbalized is used as input for the parser^[2]. The processing result is a derivation tree which describes how to construct the syntax tree using the TAG grammar. The rule set of the parser is divided into two parts, a domain specific rule set that transforms the input into an intermediate syntactic representation (ISR) and a domain independent rule set that transforms this intermediate representation into a TAG derivation tree. The TAG framework is finally used to assemble the trees, to propagate the associated features through the syntax tree and to inflect the words assigned to lexical leafs. We introduced the intermediate representation since a direct generation of the TAG derivation tree would lead to a rule set that is difficult to write and maintain. In order to avoid the construction of a syntactic representation for static text, the derivation tree might contain so-called TC nodes (text collection nodes). Possible child nodes of a TC node are nodes containing canned text (which may include formatting instructions like HTML tags), other TC nodes and nodes containing TAG derivation trees. The canned text remains unmodified by the TAG grammar^[3].

The SMARTWEB demonstrator includes a semantic database containing information about previous FIFA World Cups and access to Web Services including

¹ <http://www.smartweb-project.org>

² The name NipsGen stands for a reversed usage of the SPIN parser as generator.

³ The approach to compose canned text is also called a template based approach in contrast to a full generation approach; a discussion on that can be found in [4].

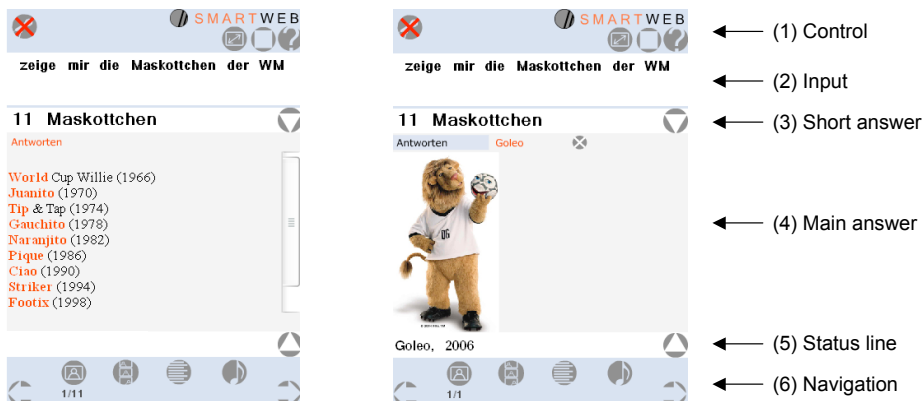


Fig. 1. The user interface of SMARTWEB. The left screenshot shows the names of the World Cup mascots, the right screenshot shows a picture of a selected mascot.

weather forecast, information about points of interest (POIs) and route planning. SMARTWEB's primary language is German, but an English version with reduced SPIN and NipsGen functionality is also available. In the context of the SMARTWEB system text generation means that objects represented in terms of the system-wide used ontology SWINTO⁴ which is RDF/S-based have to be converted into textual representations. The textual representations includes full utterances for the speech synthesizer as well as typography-enriched texts and tables which are presented on the mobile device. Figure 1 shows the graphical user interface of SMARTWEB. A more detailed description can be found in [5].

Processing Steps. Natural language generation (NLG) systems that work on the planning of coherent multisentential and longer texts exist since the late 80's, e.g., Penman⁵. Recent products, e.g., RealPro⁶, work with TFS-based, multi-level linguistic representations. We build NLG systems for Semantic Web data on mobile devices based on ontology instances. Content determination and micro-planning (e.g., reference to images in generated captions) are generated according to the requirements of the football domain, and the presentation environment. In our case the rendering on a small PDA requires more advanced summarization capabilities⁷. In addition, multimodal deixis in visual graphical presentations comes to the fore (cf. result presentation example in figure 1). The generation may have different but allegedly interchangeable surface realizations. We present our micro-planning system concerned with an expressive lexicalization—the choice of particular words and constructions used to communicate domain concepts and relations used to generate the short answer, the

⁴ http://smartweb.dfki.de/ontology_en.html

⁵ <http://www.isi.edu/natural-language/penman/penman.html>

⁶ <http://www.cogentex.com/technology/realpro/>

⁷ A glance on the display while walking and short synthesis because of modality busy setting: users can only draw attention to short information pieces.

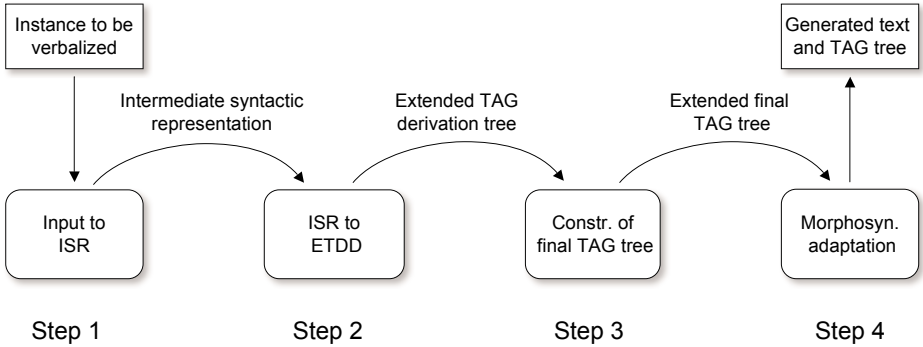


Fig. 2. The four processing steps within the NipsGen module

main answer, and the status line. The main processing consists of four processing steps (see also figure 2):

Step 1: Construction of the intermediate syntactic representation A direct transformation to the TAG derivation tree is not desirable as the derivation tree is hardly human-readable and the creation of it is an error-prone task. Therefore, an intermediate syntactic layer has been introduced which reads well and is quite easy to create. A set of domain-dependent SPIN rules transforms the input object into an intermediate syntactic representation. The nodes of the intermediate syntax tree are phrase nodes representing nominal phrases (NPs), verb phrases (VPs), adjective phrases (AdjP), adverbial phrases (AdvP), prepositional phrases (PPs), and a special node for canned text, called TC. The nodes are connected by syntactic relations like sub (subject), iObj (indirect object), dObj (direct object), pp (prepositional phrase), etc.

Step 2: Transformation to the extended TAG derivation tree A second set of domain-independent SPIN rules transforms the intermediate syntactic representation into an extended TAG derivation tree. We call it an extended TAG derivation tree as it is possible to include TC nodes which contain canned text. A TAG derivation tree is represented as a TFS whereby each TAG tree has its own type. Initial trees have the supertype `InitialTagTree`, adjunct trees the supertype `AdjunctTagTree`. Features of each tag tree encode the possible actions together with the positions, e.g., `s_221` stands for substitution operation at the tree position 221⁸.

Step 3: Construction of the derived TAG tree The syntax tree is built up in a top-down fashion using the derivation tree which is the result of step 2 and the elementary trees of the TAG grammar. After the syntax tree is built up, the

⁸ The tree position 221 means, that starting from the root node, the second child node is selected, then again the second child node and finally the first child node. The top node has the position 0.

features of the nodes are propagated using structure sharing and unification. The features are important to propagate syntactic information within the tree, e.g., the number of a sentence subject is propagated to its corresponding finite verb.

Step 4: Morphosyntactic surface realizations In a last step, the word stems are inflected. For example, **case**, **gender**, **number**, **weak/strong**, **tense**, and **person** are extracted from each lexical leaf and the corresponding inflected words are looked up in a full-form lexicon. To allow the inclusion of typographic formatting commands, the lexical leaves provide the features **formatBegin** and **formatEnd** which may contain, e.g., HTML tags, or markers of a user defined layout language.

Outlook. In the future, we plan to develop a tool that takes the TAG grammar as input and generates types, features, and rules which transform the intermediate syntactic representation into the TAG derivation tree semi-automatically (step 2 in the processing chain). We also plan to generate multiple alternative less constrained texts (something that is already supported by the SPIN parser) and test the texts against layout and presentation constraints, like the available space in the presentation area, to choose among one of these texts.

Acknowledgments. The research presented here is sponsored by the German Ministry of Research and Technology (BMBF) under grant 01IMD01A (SMARTWEB). The responsibility for this paper lies with the authors.

References

1. Reithinger, N., Bergweiler, S., Engel, R., Herzog, G., Pflieger, N., Romanelli, M., Sonntag, D.: A look under the hood - design and development of the first SmartWeb system demonstrator. In: Proc. ICMI 2005, Trento (2005)
2. Engel, R.: Robust and efficient semantic parsing of free word order languages in spoken dialogue systems. In: Proc. of Interspeech 2005, Lisbon (2005)
3. Becker, T.: Natural language generation with fully specified templates. In: Wahlster, W. (ed.) SmartKom: Foundations of Multi-modal Dialogue Systems, pp. 401–410. Springer, Heidelberg (2006)
4. Becker, T., Busemann, S.: "May I Speak Freely?" Between Templates and Free Choice in Natural Language Generation. In: Burgard, W., Christaller, T., Cremers, A.B. (eds.) KI-99: Advances in Artificial Intelligence. LNCS (LNAI), vol. 1701, Springer, Heidelberg (1999)
5. Sonntag, D.: Interaction design and implementation for multimodal mobile Semantic Web interfaces. In: Proceedings of 12th International Conference on Human-Computer Interaction (HCI 2007), Beijing, China (2007)

A Method to Optimize the Parameter Selection in Short Term Load Forecasting

Humberto F. Ferro^{1,2}, Raul S. Wazlawick^{1,3}, Cláudio M. de Oliveira^{1,3},
and Rogério C. Bastos^{1,3}

¹ Universidade Federal de Santa Catarina, UFSC-PPGCC, Florianópolis, SC, Brazil

² Centrais Elétricas de Santa Catarina, Florianópolis, SC, Brazil

³ Instituto IDESTI, Florianópolis, SC, Brazil

humbertoff@celesc.com.br, raul@inf.ufsc.br,
claudiom@inf.ufsc.br, rogerio@inf.ufsc.br

Keywords: parameter selection, load forecasting, feature extraction, industrial applications of AI, power systems.

1 Introduction

Load forecasting allows electric utilities to enhance energy purchasing and generation, load switching, contracts negotiation and infrastructure development [1].

The *consumption regions* have characteristic *consumption profiles* which determine a causal relationship between the load and a set of predictors. For short term load forecasting, in which the predictions range from few minutes to some days ahead, it is crucial to model this relationship. Because only a subset of all the available variables is relevant [3, 4], they should be examined before the model is specified. Figure 1 shows this procedure and presents the scope of this work: parameter selection and predictors identification.

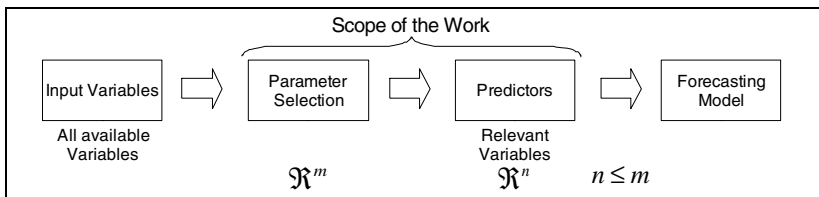


Fig. 1. Schematic Diagram of an Electrical Load Forecasting Model

The causal relationships are dynamic; i.e., predictors may change continuously their predictive relevance over time [4]. Hence, a forecasting model should be monitored continuously and rebuilt as necessary. This cycle might be repeated continually to prevent forecast errors from growing progressively with time [4].

By exploring empirical evidences that *similar profiles have similar sets of predictors*, a similarity measure among profiles is proposed. This way, new forecasting models might be constructed from the existing ones. The bigger the similarity between two profiles, the more their forecasters are alike. In the limit, the profiles will be identical and a single forecaster will fit both.

2 Load Forecasting Solutions

Hybrid models process the input data before an ANN could actually forecast. By omitting this step, known as *preprocessing*, the performance of the model is greatly decreased [2, 3]. Indeed, the overall performance of the model is far more likely to be dependent on the preprocessing than on the neural network architecture [4].

The preprocessing tries to find a set of predictors by combining input variables in several ways. The predictive relevance of these sets is tested by an evaluation model and, if the performance of such model is acceptable, a definitive model is built.

The concept of similarity among profiles can enhance preprocessing by using the knowledge that exists in forecasters already constructed. In practical terms, a criterion of similarity could help to set the slack parameters of an ANN and gives valuable insights about the optimal predictors of a new profile.

3 Forecasting Optimization Method

Figure 2 shows the proposed method as a block diagram. The *feature extraction* transforms a profile (represented by several time series of meteorological and electrical variables) in a manageable feature vector. The *knowledge base* stores information about profiles already processed; i.e., vectors, predictor sets and forecasting models. The *Control Center* recognizes the similarity among profiles and supplies some *a priori* information to boost the construction of the models. This approach sets the slack parameters of an ANN in a such way that the convergence is accelerated.

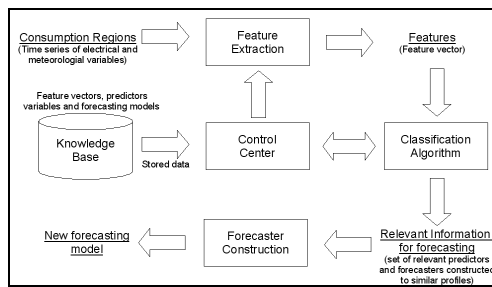


Fig. 2. Forecasting System Optimized by the Use of a Knowledge Base

Feature extraction

Let ξ_x an estimator specifically constructed to an unknown profile ρ_x . Due to the lack of knowledge about ρ_x , ξ_x employs all available variables as inputs and it is called a *dummy estimator* of ρ_x . Once just a subset of all variables has actual prediction power, this approach worsens the estimation. However, this is not an issue because the actual goal of ξ_x is to detect similarities among profiles, instead of estimate load.

If ξ_x is employed to estimate the load of another profile, say ρ_y , and the observed performance is fair, one may assume that ρ_x and ρ_y are similar; otherwise, they are not. In order to accept this similarity criterion, similar consumption profiles should have similar predictor sets, according to the following hypotheses:

\mathcal{H}_1 – The performance of a load estimator set applied to a consumption profile determines a feature vector of that profile;

\mathcal{H}_2 – Profiles with similar feature vectors have similar sets of predictors.

Figure 3 shows the performance (RMSE) of some dummy SVM estimators when they process similar (a) and different (b) profiles, producing performance curves.

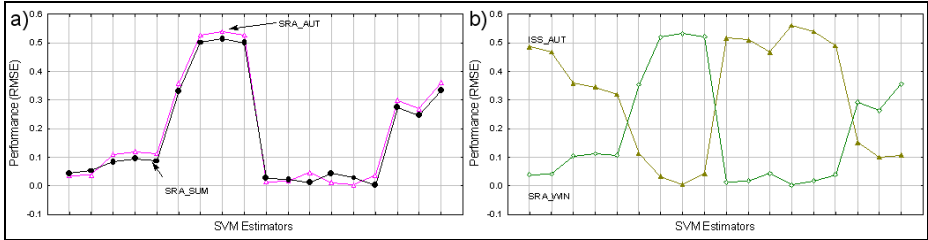


Fig. 3. Performance of several SVM estimators applied to different consumption regions

Each point of a performance curve determines a vector, which stands for a profile in a feature space. The plot shown in Figure 4a shows some vectors in the feature space, which was reduced to the Cartesian plane with MDS. The figure indicates that \mathcal{H}_1 is valid because the clusters match the intuitive definition of similarity among profiles.

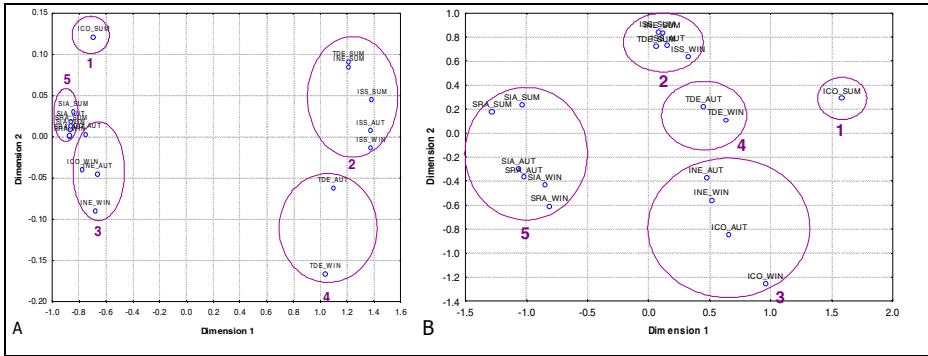


Fig. 4. Consumption profiles: (a) Feature vectors (b) Relevance vectors

A dummy estimator is employed to capture the relevance degree of a variable in the load forecasting. The worse the estimation performance when the variance of a variable is artificially nullified, the more relevant is it. Hence, such performance is a quantitative measure for the degree of relevance of that variable.

A vector (named *vector of relevance*) may be created by grouping the relevance degrees of all variables available in a profile. Such vector stands for the relevance of all variables in that profile and may be plotted in a space named *causal space*.

Figure 4b is a low resolution version of the causal space obtained by MDS. The clusters of both plots of Figure 4 are corresponding, meaning that the profiles tend to form the same clusters as their vectors of relevance, as highlighted by the numbered ellipses in both plots. This way, \mathcal{H}_2 is accepted.

4 Results and Conclusions

Twenty neural forecasters (ANNs) were created from scratch for the profiles of Figure 4a. Table 1 shows the minimum (MINTC), maximum (MAXTC) and average (AVGTC) times of convergence required to construct such forecasters.

Subsequently, the profiles were compared to each other through their feature vectors (Figure 4a). To a given profile ρ_a (column Profile in table 1), it was determined the profile ρ_b (Profile-S) that more resembles it by applying the Euclidean distance to the feature space. Also, the ANN that converged faster for ρ_b was retrained with the data of ρ_a , which produced a new ANN called *optimized ANN*. In relation to the original AVGTC of ρ_a , the optimized ANN converged faster in all cases, as indicated by the columns OTC (*optimized time of convergence*) and OTC% of Table 1.

Table 1. Elapsed times, given in seconds, to build basic forecast models

| | Profile | MINTC | MAXTC | AVGTC | Profile-S | OTC | Gain | OTC% |
|----|---------|--------|--------|--------|-----------|--------|-------|-------|
| 1 | ICO_WIN | 615.6 | 1272.8 | 1022.9 | ICO_AUT | 621.5 | 401.4 | 39.2% |
| 2 | ICO_AUT | 1056.5 | 1661.7 | 1350.5 | ICO_WIN | 951.1 | 399.4 | 29.6% |
| 3 | ICO_SUM | 1336.4 | 1696.5 | 1530.9 | ICO_AUT | 1509.0 | 21.9 | 1.4% |
| 4 | INE_WIN | 91.6 | 194.5 | 123.9 | INE_AUT | 88.3 | 35.6 | 28.7% |
| 5 | INE_AUT | 120.9 | 212.1 | 175.0 | INE_WIN | 57.4 | 117.7 | 67.2% |
| 6 | INE_SUM | 87.7 | 173.8 | 128.2 | TDE_SUM | 61.2 | 67.0 | 52.2% |
| 7 | ISS_WIN | 60.8 | 155.6 | 106.2 | ISS_AUT | 29.9 | 76.3 | 71.9% |
| 8 | ISS_AUT | 113.4 | 220.0 | 165.1 | ISS_WIN | 27.5 | 137.6 | 83.3% |
| 9 | ISS_SUM | 62.1 | 89.0 | 72.7 | ISS_AUT | 31.8 | 40.9 | 56.2% |
| 10 | SIA_WIN | 141.3 | 199.5 | 162.2 | SIA_AUT | 50.0 | 112.2 | 69.2% |
| 11 | SIA_AUT | 129.0 | 185.9 | 156.0 | SIA_WIN | 56.5 | 99.5 | 63.8% |
| 12 | SIA_SUM | 93.4 | 211.8 | 142.8 | SIA_AUT | 58.4 | 84.5 | 59.1% |
| 13 | SRA_WIN | 141.8 | 258.3 | 190.7 | SRA_AUT | 86.7 | 104.0 | 54.5% |
| 14 | SRA_AUT | 102.4 | 186.5 | 163.6 | SRA_WIN | 56.9 | 106.6 | 65.2% |
| 15 | SRA_SUM | 97.9 | 178.6 | 132.0 | SIA_WIN | 61.5 | 70.4 | 53.4% |
| 16 | TDE_WIN | 84.3 | 139.4 | 106.7 | TDE_AUT | 59.3 | 47.4 | 44.4% |
| 17 | TDE_AUT | 93.8 | 128.8 | 113.4 | TDE_WIN | 30.7 | 82.7 | 72.9% |
| 18 | TDE_SUM | 91.5 | 341.2 | 153.4 | INE_SUM | 47.7 | 105.7 | 68.9% |

Table 1 shows considerable improvements in the learning time of all forecasters. Hence, Figure 2 may be considered a realistic approach to optimize load forecasting.

References

1. Iyer, V., Che, C., Gedeon, T.: A Fuzzy-Neural Approach to Electricity Load and Spot Price Forecasting. Tencom (2003)
2. Guo, X., Chen, Z., Ge, H., Liang, Y.: Short-Term Load Forecasting Using Neural Network With Principal Components Analysis. In: 3rd International Conference on Machine Learning and Cybernetics, Shanghai, China (2004)
3. Tao, X., Renmu, H., Peng, W., Dongjie, X.: Input Dimension Reduction for Load Forecasting Based on Support Vector Machines. In: 2004 IEEE Conference on Electric Utility De-regulation, Restructuring and Power Technologies (2004)
4. Oliveira, C.M.: Modelo Adaptativo Para Previsão De Carga Ativa De Curto Prazo. PhD Thesis, Production Eng. Dept. – UFSC (2004)

Visual Robot Localization and Mapping Based on Attentional Landmarks

Simone Frintrop

Comp. Science III, University of Bonn, Germany
frintrop@iai.uni-bonn.de

Abstract. In this paper, we present a system for simultaneous localization and map building of a mobile robot, based on an attentional landmark detector. A biologically motivated attention system finds regions of interest which serve as visual landmarks for the robot. The regions are tracked and matched over consecutive frames to build stable landmarks and to estimate the 3D position of the landmarks in the environment. Furthermore, matching of current landmarks to database entries enables loop closing and global localization. Additionally, the system is equipped with an active camera control, which supports the system with a tracking, a re-detection, and an exploration behaviour.

1 Introduction

One of the most important tasks of a mobile robot is to localize itself within its environment. This task is especially difficult if the environment is not known in advance. Within the robotics community, this problem is well known as *SLAM* (*Simultaneous Localization and Mapping*). Currently, there has been special interest in *visual SLAM*, which uses cameras as main sensors since cameras are low-cost, low-power and lightweight sensors [1,6,7].

A key competence in visual SLAM is to choose useful visual landmarks which are easy to track, stable over several frames, and easily re-detectable when returning to a previously visited location. Here, we present a visual SLAM system based on an attentional landmark detector: the attention system VOCUS [2] detects regions of interest (ROIs) which are tracked and matched over consecutive frames. To improve the stability of the features, the ROIs are combined with Harris corners. When re-visiting a location after some time, knowledge about the appearance of expected landmarks is used to search in a top-down manner for expected features. Additionally, active camera control improves the quality and distribution of detected landmarks.

The novelty of the presented system in comparison to other approaches of visual SLAM – e.g., [1,6,7] – lies first, in the attentional feature detection in combination with Harris corners [4], second, in the top-down, target-directed feature computations to improve loop closing [3], and third, in the active camera control [5]. Here, we combine the results of these previous findings.

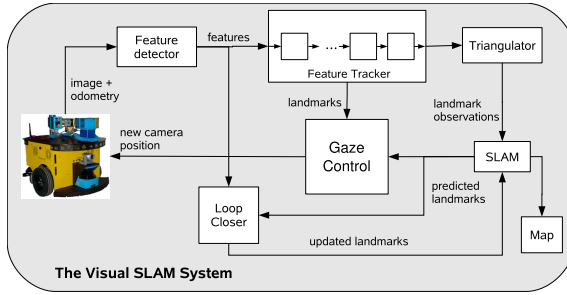


Fig. 1. The visual SLAM system builds a map based on image data and odometry

2 The Visual SLAM System

The visual SLAM architecture (Fig. 1) consists of a *robot* which provides camera images and odometry information, a *feature detector* to find ROIs in the images, a *feature tracker* to track ROIs over several frames and build landmarks, a *triangulator* to identify useful landmarks, a *SLAM module* to build a map of the environment, a *loop closer* to match current ROIs to the database, and a *gaze control module* to determine where to direct the camera to.

When a new frame from the camera is available, it is provided to the *feature detector*. This module finds ROIs based on the visual attention system VOCUS [2]. VOCUS computes a bottom-up saliency map, based on strong contrasts and uniqueness of the features intensity, orientation, and color. For each ROI, a feature vector is stored which is used for matching and top-down search. Since the shape of attentional ROIs differs sometimes in consecutive frames, the ROIs are combined with Harris corners to improve position stability [4]. A bottom-up saliency map and the corresponding ROIs are displayed in Fig. 2.

Next, the features are provided to the *feature tracker* which stores the last n frames, performs matching of ROIs and Harris corners in these frames and creates landmarks which are lists of features found in several frames. Matching of ROIs and Harris corners is based on proximity and similarity of the feature vector (ROIs) or a SIFT descriptor (Harris) [4]. The purpose of the buffer is to identify features which are stable over several frames and have enough parallax information for 3D initialization. These computations are performed by the *triangulator*. Selected landmarks are stored in a database and provided to the *SLAM module* which computes an estimate of the position of landmarks and integrates the position estimate into the *map* (details to SLAM module in [6]).

The task of the *loop closer* is to detect if a scene has been seen before. The SLAM module provides the loop closer with expected landmark positions and their feature descriptions. The attentional feature vector is used to search in a top-down manner for the expected landmarks. The result is a top-down saliency map which highlights regions which correspond to the target (cf. Fig. 2). The corresponding top-down ROIs are compared with the ROIs of the expected landmarks by comparing the similarity of their feature vectors. If two ROIs match,

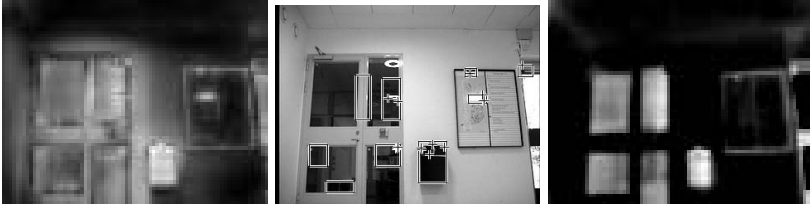


Fig. 2. Left: bottom-up saliency map. Middle: attentional ROIs (rectangles) and Harris corners (crosses). Right: top-down saliency map for target “wastebin” (black box).

this information is provided to the SLAM module to update the positions of the robot and the landmarks.

Finally, the *gaze control module* controls the camera actively with three behaviours: a *tracking* behaviour identifies the most promising landmarks and prevents them from moving out of the field of view. A *redetection* behaviour actively searches for expected landmarks to support loop-closing. Finally, an *exploration* behaviour investigates regions with no landmarks, leading to a more uniform distribution of landmarks. The process to decide which behaviour is activated is based on the amount of uncertainty about the current position and on the number of currently visible landmarks (details in [5]).

3 Experiments and Results

To illustrate the advantages of the presented visual SLAM system, we performed two experiments which show i) the advantages of the top-down attentional matching approach in loop closing situations, and ii) the advantages of active over passive camera control. In both experiments, the robot drove through a room in an office environment, through a corridor, and entered the room again. Here, it should detect that it closed a loop.

In the 1st experiment, we compared bottom-up matching of ROIs (VOCUS computes a bottom-up saliency map and the similarity of ROIs is compared based on a threshold) and top-down matching (VOCUS searched for the expected landmarks in the current frame and the resulting ROIs are compared afterwards) (Fig. 3 left). If only very few false matches are accepted, the bottom-up matching is better. But if more false matches are acceptable, we get a significantly higher number of correct matches (42% more). Note that this number of false matches is not the number of false matches reported to SLAM since several of the matched ROIs belong to the same landmark and we also use matching of Harris corners afterwards to reduce the number strongly (details in [4]). In the current example, only one false landmark match remained.

In the 2nd experiment, we compared passive with active camera control. The resulting maps are displayed in Fig. 3 middle/right. With active control, we achieve a better distribution of landmarks and more matches, e.g. loop closing takes places earlier and more reliably (details in [5]).

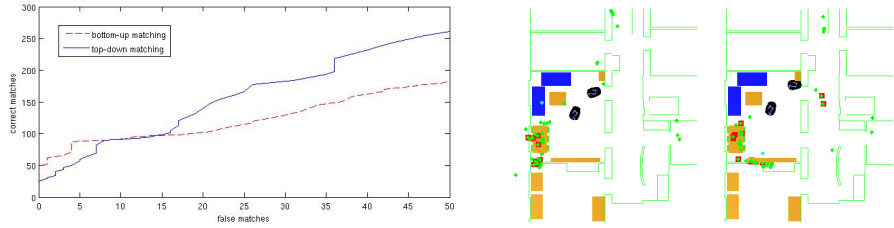


Fig. 3. Left: Experiment 1: Correct matches for bottom-up and top-down matching depending on the error rate: For a low number of false detections, bottom-up matching results in more correct matches. If more false matches are acceptable, top-down matching provides more correct matches. **Middle/Right: Experiment 2:** Two maps consisting of visual landmarks (green/cyan dots), created with passive (middle) and with active (right) camera control. Two robots in one image correspond to the robot at the beginning and at the end of the buffer, i.e., the robot further ahead on the path is the real robot, the one behind is the virtual robot position 30 frames later. Landmarks matched to database entries are displayed as large, red dots. Active control enables a better distribution of landmarks and more matches.

4 Conclusion

We have presented a visual SLAM system based on an attentional landmark detector. The attentional regions are especially useful landmarks for tracking and redetection; the loop closing is improved by using top-down guidance. Active camera control helps to achieve better, more stable landmarks, a better distribution of landmarks, and a faster and more reliable loop closing.

In future work, we plan to combine the method with other visual loop-closing techniques, for example by considering not only one expected landmark for matching, but all in the current field of view.


References

1. Davison, A.J.: Real-time simultaneous localisation and mapping with a single camera. In: Proc. of the ICCV (october 2003)
2. Frintrop, S.: VOCUS: A Visual Attention System for Object Detection and Goal-directed Search. PhD thesis, Bonn, Germany. LNAI, Springer (2006)
3. Frintrop, S., Cremers, A.B.: Top-down attention supports visual loop closing. In: Proc. of ECMR (to appear, 2007)
4. Frintrop, S., Jensfelt, P., Christensen, H.: Pay attention when selecting features. In: Proc. of the 18th Int'l Conf. on Pattern Recognition (ICPR 2006) (2006)
5. Frintrop, S., Jensfelt, P., Christensen, H.: Attentional robot localization and mapping. In: ICVS Workshop WCAA (2007)
6. Jensfelt, P., Kragic, D., Folkesson, J., Björkman, M.: A framework for vision based bearing only 3D SLAM. In: Proc. of ICRA'06, Orlando, FL (May 2006)
7. Newman, P., Ho, K.: SLAM- loop closing with visually salient features. In: Proc. of the International Conference on Robotics and Automation (ICRA 2005) (2005)

Bridging the Sense-Reasoning Gap Using DyKnow: A Knowledge Processing Middleware Framework

Fredrik Heintz, Piotr Rudol, and Patrick Doherty

Department of Computer and Information Science
Linköpings universitet, Sweden
{frehe, pioru, patdo}@ida.liu.se

Abstract. To achieve complex missions an autonomous unmanned aerial vehicle (UAV) operating in dynamic environments must have and maintain situational awareness. This can be achieved by continually gathering information from many sources, selecting the relevant information for current tasks, and deriving models about the environment and the UAV itself. It is often the case models suitable for traditional control, are not sufficient for deliberation. The need for more abstract models creates a sense-reasoning gap. This paper presents DyKnow, a knowledge processing middleware framework, and shows how it supports bridging the gap in a concrete UAV traffic monitoring application. In the example, sequences of color and thermal images are used to construct and maintain qualitative object structures. They model the parts of the environment necessary to recognize traffic behavior of tracked vehicles in real-time. The system has been implemented and tested in simulation and on data collected during flight tests. 

1 Introduction

Unmanned aerial vehicles (UAVs) are becoming commonplace in both civil and military applications, especially for missions which are considered dull, dirty and dangerous. One important application domain for UAVs is surveillance. Such missions may involve flying over unknown areas to build terrain models, to quickly get an overview of a disaster area including helping the rescue services to find injured people and deliver medical supplies, or to help law enforcement agencies to monitor areas or people for on-going or potential criminal activity. To achieve these complex missions an autonomous UAV must continuously gather information from many different sources, including sensors, databases, other UAVs, and human operators. It then selects relevant information for the current task, and derives higher-level knowledge about the environment and the UAV itself in order to understand what is happening and to make appropriate decisions. In other words, the UAV must create and maintain its own situational awareness in a timely manner.

To create situation awareness a UAV needs to build models of the environment and use them to reason contextually about the past, current, and future state of the world.

¹ This work is supported in part by the National Aeronautics Research Program NFFP04 S4203 and the Strategic Research Center MOVIII, funded by the Swedish Foundation for Strategic Research, SSF.

These models should be constructed from information gathered from distributed sources and aggregated in a timely manner in order to capture the latest developments. Since there are many models that could be built and since a UAV has limited resources it is important that the appropriate models are constructed contextually for the particular task at hand. When the task changes this should be reflected in the models as well.

What is an appropriate model will depend on what properties of the world are relevant, what reasoning is needed to make decisions to achieve a task, and the context within which that reasoning is made. One functionality is the construction of models from data aggregated from different sensors that can be used to reason about the environment and the UAV in real-time. There are numerous approaches to building quantitative models based on sensor data. These models are suitable for traditional tracking and control applications but do not provide appropriate abstractions when reasoning about complex situations such as traffic. On the other hand, there are many qualitative modeling approaches using formal symbols which are well suited to do high level reasoning about the environment. How to connect these different approaches and to close the gap between sensing and reasoning is still an open research question.

This paper presents an implemented traffic monitoring application that uses the knowledge processing middleware framework DyKnow [12] to bridge the sense-reasoning gap. It is done by creating tailored models at different levels of abstraction as described by declarative policies. The models are interconnected in order to describe dependencies and to keep them updated. The models created can be used to reason qualitatively about the world as it develops using for example temporal logic and a complex event formalism called chronicle recognition.

2 Traffic Monitoring

Imagine a human operator trying to maintain situational awareness about a traffic situation in an urban area using UAVs which look for accidents, reckless driving, or other relevant activities. One approach would be for one or more UAVs to relay videos and other data to an operator for human inspection. Another more scalable approach would be for the UAVs to monitor traffic situations which arise and only report back the high level events observed. This would reduce the amount of information generated and help an operator focus attention on salient events. This paper describes such a traffic monitoring application where cars are tracked by a UAV platform and streams of observations are fused with a model of the road system in order to draw conclusions about the behavior of cars.

The input consists of images taken by color and thermal cameras on a UAV which are fused and geolocated into a single world position. This stream of positions is then correlated with geographical information system (GIS) data in order to know where in a road system an object is located. Based on this information, high level behaviors such as turning at intersections and overtaking are recognized in real-time as they develop using a chronicle recognition system.

An overview of the components of the traffic monitoring application is shown in Fig. 1. The three sensors used, the two cameras and the helicopter state estimation (which is fused from inertial and GPS data), are shown to the left. These provide the

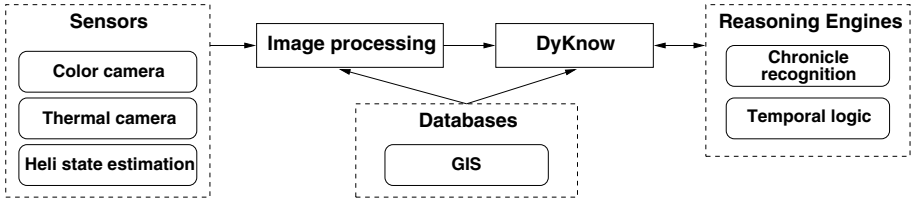


Fig. 1. Overview of the components of the traffic monitoring application

low level data about the environment and the UAV. The next component is an image processing system which tracks objects seen by the cameras. When an object is being tracked, images from the two cameras are fused to provide an estimation of the position in the world of the tracked object. Each time a new frame is analysed a new position estimate is produced. From this stream of position estimations DyKnow derives further abstractions used to recognize high level events and closes the sense-reasoning gap.

To describe a high-level event a formal representation called a *chronicle* is used [3]. A chronicle defines a class of events using a simple temporal network (STN) [4] where the nodes are primitive events and the edges are temporal constraints between event occurrences. The chronicle recognition engine takes a stream of primitive events and detects all chronicle instances. An instance is detected if the stream contains a set of event occurrences which satisfy all the constraints in a chronicle model. The chronicles used in this application contain primitive events which capture the structure of the road network, qualitative information about cars such as which road segment they are on, and qualitative spatial relations between cars such as beside and behind. Creating a stream of events based on sensor data which accurately represents the environment of the UAV, is a concrete instance of the sense-reasoning gap.

To bridge the gap, DyKnow takes a stream of position observations provided by the image processing system and derives an event stream representation of cars and qualitative spatial relations between cars. DyKnow also derives an event stream representation of the road network from the information stored in the GIS. One issue that must be handled is how to anchor car symbols used in the chronicles to objects being tracked [5]. Since the image processing system may lose track of cars or start tracking other non-car objects, DyKnow has to dynamically estimate and continually monitor the type and identity of objects being tracked. To do this, the normative behavior of different objects and the conditions for assuming that two objects have the same identity are described using temporal logic. When a tracked object is found which satisfies the normative behavior of e.g. a car, a new car representation is created and the tracked object is *linked* to the new car representation. From this moment the car representation will be updated each time the tracked object is updated. Since links only represent hypotheses, i.e. they are always subject to becoming invalid given additional observations, the UAV continually has to verify the validity of the links. This is done by monitoring that the normative behavior of the assumed object type is not violated. For example, an object assumed to be a car must not violate the normative constraints on cars, e.g. leaving the road. If it does violate the constraint, then the corresponding link is removed, in other words the object is no longer assumed to be a car.

To evaluate a temporal logical formula, DyKnow has to derive a model representing the value over time of the variables used in the formula. These values must be synchronized in time, so that the evaluation mechanism receives a state for each time-point containing the value of each of the variables at that time-point. This is done by defining a policy for each of the formulas which DyKnow uses to derive the required model. Since these models are derived from sensor data, it is another concrete example of how DyKnow can be used to bridge the sense-reasoning gap. The evaluation is done using progression which means that the evaluation is performed in real-time as soon as a new state is available. This means that the truth value of a formula will be derived as soon as it is possible.

The application has been tested both on simulated cars driving in a road system and on real data captured during flight tests.

3 Conclusions

A traffic monitoring application which is an instance of a general approach to creating high-level situation awareness has been presented. The implemented system takes as input sequences of color and thermal images. They are used to construct and maintain qualitative object structures and recognize the traffic behavior of the tracked vehicles in real-time. The system is tested both in simulation and on data collected during flight tests. We believe that this type of system where streams of data are generated at many levels of abstraction using both top-down and bottom-up reasoning handles many of the problematic issues related to closing the sense-reasoning gap in robotic systems. One reason for this is that the information derived at each level is available for inspection and use at all times. This means that the subsystems have access to the appropriate abstraction while it is being continually updated with new information and used to derive even more abstract structures. High-level information, such as the type of vehicle, can then be used to constrain and refine the processing of lower level information. The result is a very powerful and flexible system capable of achieving and maintaining high-level situation awareness.

References

1. Heintz, F., Doherty, P.: DyKnow: An approach to middleware for knowledge processing. *Journal of Intelligent and Fuzzy Systems* 15(1), 3–13 (2004)
2. Heintz, F., Doherty, P.: A knowledge processing middleware framework and its relation to the JDL data fusion model. *Journal of Intelligent and Fuzzy Systems* 17(4), 335–351 (2006)
3. Ghallab, M.: On chronicles: Representation, on-line recognition and learning. In: *Proceedings of the Fifth Intl Conf on Principles of Knowledge Representation and Reasoning* (1996)
4. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. *AIJ* 49 (1991)
5. Coradeschi, S., Saffiotti, A.: An introduction to the anchoring problem. *Robotics and Autonomous Systems* 43(2-3), 85–96 (2003)

Emotion Based Control Architecture for Robotics Applications

Jochen Hirth, Tim Braun, and Karsten Berns

Robotics Research Lab
Department of Computer Science
University of Kaiserslautern Germany
{j_hirth, braun, berns}@infomatik.uni-kl.de

Abstract. Assistance and service systems are one of the main research topics in robotics today. A major problem for creating these systems is that they have to work and navigate in the real world. Because this world is too complex to model, these robots need to make intelligent decisions and create an intelligent behavior without knowing everything about the current situation. For these aspects, the importance of emotion increases, because the emotional influence helps human beings as well as animals to make their decisions. To enable a robot to use emotions, a concept for an emotion based control architecture was designed. The basis of this architecture is a behavior based approach. This paper presents the developed architecture. Furthermore two application possibilities are presented, where parts of the architecture were already tested and implemented.

Keywords: control architecture, behavior based control, intelligent robotic systems.

1 Introduction

The realization of emotions should be a central aspect of any intelligent machine. Rational and intelligent behavior is needed in nearly every autonomous robot system. These robots have to make decisions depending on their sensor data. Neuroscience, psychology and cognitive science suggest that emotion plays an important role in rational and intelligent behavior [1]. Because of this it is very important to use the emotional component in a robot system that should work and decide autonomously.

Worldwide, several research projects focus on the development of emotional control architectures for robot systems, like e.g. [2] or [3]. In [4] a survey of artificial cognitive systems is presented. Different models, theories, and paradigms of cognition addressing cognitive approaches, emergent systems approaches, encompassing connectionist, dynamical, and enactive systems. Furthermore several cognitive architectures drawn from these paradigms are presented.

2 Emotional Architecture

Depending on psychological theories [5] [6] an emotion based robot control architecture was designed. In the following section the concept of this architecture will be described in detail. The architecture consists of 3 main parts: behavior, emotion, and cognition. All possible movements of the robot from simple reflexes up to high level motor skills are located in the behavior group. These behaviors are activated in different ways, e.g. directly depending on sensor data, depending on the emotional state of the machine or deliberately by the cognition part.

Behavior. Every single behavior is build out of the behavior nodes, described in [7]. Depending on these behavior-nodes all different kinds of behaviors can be realized. Low level behaviors that moves the different motors to one direction like e.g. reflexes or high level behaviors that represent complex motor skills. These behaviors are activated by different parts of the architecture. The more high level behaviors are mostly activated by the emotion and especially by the cognitive part. Whereas the low level behaviors are also activated directly by sensor input. These low level reflexes directly activated by the sensor perception build the reactive layer of our architecture. In the information flow of this reactive system is displayed.

Emotion. The emotion group consists of 2 parts. The emotional state which is just for the representation of the actual internal emotional state of the robot and drives. The drives represent low level goals of the robots behavior like e.g. survival or energy consumption etc. A drive gets active if the discontent reaches a certain threshold. The drive than calculates parameters that change the emotional state and that select the behaviors of the robot. The aim of the drive is to reach a saturated state by the selection of the behaviors. If the saturation of the drive is getting higher the activity of the drive is getting lower. According to projects like e.g. [2] an emotional space for the representation of the emotional state was developed. The 3 axis of this emotional space are arousal (A), valence (V), and stance (S). That means every emotion is described by these 3 parameters. In most emotional spaces for every emotion a certain area in the emotional space is reserved. If the actual emotional space is in this area the corresponding emotion is activated. That means every emotion that should be used has to be defined. The problem is that psychologists say that there are a lot of emotions and they do not even know all of them. But most of them agree that all emotions are build out of the 6 basic emotions: anger, disgust, fear, happiness, sadness, and surprise.

Cognition. The cognition should generate a plan to reach a certain goal by combining several behaviors of the system. Therefore it can use sensor information as well as emotional information and behavior information. As mentioned above, especially the emotional state is a main factor to create an intelligent decision. According to a human the cognitive part is also able to suppress the emotional state for a while or to influence deliberately the expression of the emotional state to reach a certain goal. The cognitive layer then creates a chain of

behaviors. Running this chain should lead to the goal. If something unexpected happens during this run the cognitive layer has to reschedule this chain. For this reschedule decision the emotional state is of enormous importance if the system should work in a real world environment.

3 Possible Applications of the Emotional Architecture

Humanoid Robot Head ROMAN. The emotional architecture was already used for the humanoid robot head ROMAN (see Fig. 1). Because of the actual emotional state the corresponding facial expressions are generated more details and experiments can be found in 8.

The architecture was also used to realize a drive-based behavior of the robot. That means within the architecture different drives like e.g. exploration and communication are defined. These drives determine the goals of the robots behavior and the emotional state of the robot (see 9).

One of the next steps in this project will be the usage of the cognitive layer of the proposed architecture. The robot should use its expressions to reach a certain goal within an interaction.

Mobile Robot RAVON. Another application possibility for the emotional architecture described in this paper arises in the path planning component of the mobile outdoor robot RAVON (Fig. 2). Here, the introduction of an 'emotional state' into the robots' navigational layer allows the system to solve the 'action-selection' type problem of choosing a path from the set of currently possible trajectories in a psychologically plausible way. Using the emotional state as an abstracted indication of the overall robot situation (considering navigational capabilities, battery state and/or available mission time), the path planner can weight the different factors that influence the path finding decision appropriately and select a solution that is globally optimal.

With the basic emotionally influenced cost model for path planning in place, the project currently focuses on adding the drives component described in this paper in order to adjust the motivational state of the robot according to success



Fig. 1. The robot head ROMAN



Fig. 2. The mobile robot RAVON

and failure in exploration and exploitation of the topological map. It is planned to combine drives modeling self-preservation, curiosity and fatigue for this.

4 Summary and Outlook

The concept of an emotion based control architecture for autonomous robots is presented. This architecture consists of 3 main parts: Behavior, Emotion, and Cognition. The great advantage of the introduced architecture is that it can be used in completely different systems, as described in section 3. In the future the proposed architecture had to be improved with the help of psychologists, sociologist, and biologists. In addition the parts that had not been implemented and tested till now have to be realized on robots. And finally the whole system has to be tested on different robots.

References

1. Picard, R.: Affective computing. Technical Report 321, MIT Media Laboratory, Perceptual Computing Section (November 1995)
2. Breazeal, C.: Sociable Machines: Expressive Social Exchange Between Humans and Robots. PhD thesis, Massachusetts Institute Of Technoligy (May (2000)
3. Zhang, H., Liu, S., Yang, S.X.: A hybrid robot navigation approach based on partial planning and emotion-based behavior coordination. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, October 9-15 2006, pp. 1183–1188 (2006)
4. Vernon, D., Matta, G., Sandini, G.: A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agent. *IEEE Trans. Evolutionary Computation*, special issue on Autonomous Mental Development, 2006 (in press)
5. Bösel, R.: Biopsychologie der Emotionen. Walter de Gruyter (1986)
6. Martin, L., Clore, G.: Theories of Mood and Cognition. Lawrence Erlbaum Associates, Inc., Mahwah (2001)
7. Albiez, J., Luksch, T., Berns, K., Dillmann, R.: An activation-based behavior control architecture for walking machines. *The International Journal on Robotics Research*, Sage Publications 22, 203–211 (2003)
8. Berns, K., Hirth, J.: Control of facial expressions of the humanoid robot head roman. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 9-15 2006, Beijing, China (2006)
9. Hirth, J., Schmitz, N., Berns, K.: Emotional architecture for the humanoid robot head roman. In: IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy, April 11-13, 2007, IEEE Computer Society Press, Los Alamitos (2007)

Inductive Synthesis of Recursive Functional Programs

A Comparison of Three Systems

Martin Hofmann, Andreas Hirschberger, Emanuel Kitzelmann,
and Ute Schmid

University of Bamberg, Faculty of Information Systems
and Applied Computer Science

{martin.hofmann, andreas.hirschberger}@stud.uni-bamberg.de,
{emanuel.kitzelmann, ute.schmid}@wiai.uni-bamberg.de

1 Introduction

One of the most challenging subfields, and a still little researched niche of machine learning, is the inductive synthesis of recursive programs from incomplete specifications, such as examples for the desired input/output behavior [1,2,3,4]. The special appeal of an *inductive* approach to automated program construction is that the user only provides some examples of the desired program behaviour, such as $[A, B, C] \rightarrow [C, B, A]$, as input to the synthesis system and a general program (here for reversing a list) is created. Potential applications for automatic program induction are to enable end-users to create their own simple programs, to assist professional programmers or even to automatically invent new and efficient algorithms.

Existing inductive program synthesis systems use either a search-based or an analytical approach. While the most promising are allocated in the subfield of program synthesis, also concept learners with extended codomain and especially ILP-based systems exhibit success worthy to mention.

Since no broadly accepted fundamentals and approaches prevail in the field of inductive program synthesis. We have systematically evaluated three systems to inductively synthesise functional recursive programs [5] which are based on three fundamentally different induction methods.

2 The Systems

ADATE [5] (Automatic Design of Algorithms Through Evolution) is a system for automatic programming utilising evolutionary computation for program generation. It's specifications written in a sub-language of ML, which are then embedded in Adate's own ML source code. This is used as an initial program which is evolved by applying evolutionary operators during the search.

¹ By “functional” we refer to the mapping of each input to a unique output by a synthesised program and not to a specific programming paradigm.

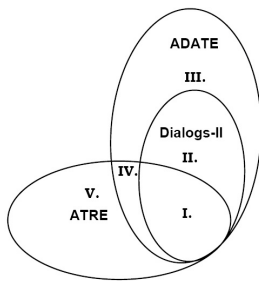
ATRE [6,7] is a classification learner, capable of simultaneously learning mutually dependent, recursive multi-class target predicates. ATRE performs a general-to-specific parallel beam search in the space of definite clauses ordered by generalised implication [7]. The search space can be described as a forest of several search trees, where multiple trees are processed by a sequential covering algorithm simultaneously, following a learn-one-rule strategy.

DIALOGS-II (Dialogue-based Inductive and Abductive LOGic program Synthesiser) [8] is a schema-guided, interactive, inductive and abductive recursion synthesiser that takes the initiative and queries a (possibly computational naive) specifier for evidence in her/his conceptual language.

Although our focus is on the synthesis of recursive *functions*, our nomenclature is, for the sake of convenience, based on the ILP glossary. Therefore, we understand *background knowledge* as any additional, user provided, problem-specific information, as e.g. predefined predicates or helper-functions and *predicate invention* [2] as the automated generation of new predicates or subfunctions to solve the problem, neither mentioned in the background knowledge, nor in the training examples.

3 Empirical Setup and Results

Problem Classes. Although all three synthesis systems provide to some extent the same means to learn recursive programs, they still remain quite inhomogeneous concerning their underlying concepts. To properly evaluate them, we were forced to some common denominator for their problem space. The Venn diagram (Fig. 1) shall illustrate the extend of the capabilities of the three systems and explains the identified classes used in the evaluation process.



Single recursive call without predicate invention (I.): solvable with a single recursive call in the body of the predicate definition; no predicate or variable invention is required.

Single recursive call with predicate invention (II.): at least the invention of an auxiliary predicate is required.

Multiple recursive call (III. + IV.): at least a second recursive call is necessary (either of another recursive predicate or of the target predicate itself)

Miscellaneous (V. + III.): emphasises the individual strengths of a certain system.

Fig. 1. Venn Diagram of System Capabilities²

² Classes III. and VI. were combined, since DIALOGS-II is not capable of multiple recursive calls and an ATRE specification for such a problem would result in an extensive enumeration of input/output pairs.

Table 1. Overview of the evaluated problems**(1.) Single Recursive Call without Predicate Invention**

evenpos(X, Y) holds iff list Y contains all elements of list X at an even position in unchanged order.

insert(X, Y, Z) holds iff X is a list with its elements in a not decreasing order, and Z is X with Y inserted on the right place.

inlast(X, Y, Z) holds iff Z is the list X with Y inserted at the end.

last(X, Y) holds iff Y is the last element of the list X .

length(X, Y) holds iff Y is the length of the list X .

member(X, Y) holds iff X is a list containing the element Y .

switch(X, Y) holds iff list Y can be obtained from list X were all elements on an odd position changed place with their right neighbour.

unpack(X, Y) holds iff Y is a list of lists, each containing one element of X in unchanged order.

(2.) Single Recursive Call with Predicate Invention

i-sort(X, Y) holds iff the list Y is a permutation of list X with elements in a non decreasing order.

multlast(X, Y) holds iff the list Y contains nothing but the last element of list X as many times as the number of elements in X .

reverse(X, Y) holds iff the list Y is the reverse of list X .

shift(X, Y) holds iff list Y could be derived from list X by shifting the first element to the end.

swap(X, Y) holds iff list Y could be derived from list X by swapping the first and the last element.

(3.) Multiple Recursive Call with(out) Predicate Invention

lasts(X, Y) holds iff X is a list of lists, and Y contains the last elements of each list in X in the correct order.

flatten(X, Y) holds iff Y is the flattened version of the list of lists X .

(4.) Miscellaneous Problems

mergelists(X, Y, Z) holds iff the list Z could be derived from the lists X and Y such that $Z = [x_1, y_1, x_2, y_2, \dots]$ where each x_n and y_n is the n^{th} of the list X and Y , respectively.

odd(X)/*even*(X) holds iff X is an odd, respectively even number, and each predicate is defined in terms of *zero*(X) and the other.

Description of Problems. We especially concentrated on problems over lists, because with their simple structure they allowed us to tailor problems with a minimum of necessary background knowledge and also remained more or less unchanged for all systems, to make the whole evaluation more transparent and facilitate the comparison of results between the different systems. Table 1 gives a short description of our examined problems. It contains typical, already researched problems [9], as well as some of our own problems, tailored to these classes.

3.1 Results of the Test Setting

Table 2 shows the results of the test runs with different systems on problems from classes described above. It is not surprising that program synthesis is up till

Table 2. Problem runtimes *in seconds* on different systems

| | (1.)■ | | | | | | | | (2.)■ | | | | | (3.)■ | (4.)■ | | |
|------------|-----------------|-----------------|-----------------|---------------|-----------------|-----------------|------------------|-----------------|------------------|-----------------|---------------|----------------|-------------------|------------------|----------------|---------------------|---------------------|
| | <i>member/2</i> | <i>unpack/2</i> | <i>length/2</i> | <i>last/2</i> | <i>inlast/3</i> | <i>switch/2</i> | <i>evenpos/2</i> | <i>insert/3</i> | <i>reverse/2</i> | <i>i-sort/2</i> | <i>swap/2</i> | <i>shift/2</i> | <i>multlast/2</i> | <i>flatten/2</i> | <i>lasts/2</i> | <i>odd/1 even/1</i> | <i>mergelists/3</i> |
| ADATE | 2.0 | 1.5 | 1.2 | 0.2 | 2.7 | 2.8 | 1.6 | 16 | 78 | >70 | 232 | 15 | 4.3 | 110 | 822 | — | 80 |
| ATRE | 91.6 | × | 17.9 | 6.4 | × | 1983 | 156⊥ | — | — | — | — | — | — | — | — | 0.05 | — |
| DIALOGS-II | 0.03⊥ | 0.05 | 0.04 | 0.03⊥ | 0.03 | 0.19 | × | 0.06 | 0.07 | 0.09⊥ | 0.15 | 0.11 | 0.13⊥ | × | × | — | — |

(— not tested × failed ⊥ wrong)

now characterised by a very strong trade-off between the restriction of the search space and the time needed for synthesis. ADATE operates in a quite unrestricted search space, capable of finding powerful solutions for complex problems, whereas DIALOGS-II successively confines the search space, but with disadvantageous loss of expressional power. ATRE is an indicative example that extending a concept learner with recursive abilities is not sufficient for satisfactory program synthesis.

4 Conclusion

The goal of future research should be to combine the DIALOGS-II's search bias with the unrestricted search space of ADATE and the expressional power of functional languages. This could for example be done by using the input/output examples during ADATE's search not only for validation but also as an heuristic. Nevertheless, ATRE can still serve as a salutary example, since the accessory adoption of ATRE's k -beam search strategy could make it possible to learn mutually dependent recursive target functions, provided ADATE's search time could be reduced significantly. We currently use these insights in our system IGOR [4]. This new approach formalises functional program synthesis in the term-rewriting framework and represents functional programs as constructor term rewriting systems containing recursive rules.

References

1. Biermann, A.W., Guiho, G., Kodratoff, Y. (eds.): Automatic Program Construction Techniques. Macmillan, New York (1984)
2. Flener, P., Yilmaz, S.: Inductive synthesis of recursive logic programs: Achievements and prospects. *J. Log. Program.* 41(2-3), 141–195 (1999)
3. Flener, P., Partridge, D.: Inductive programming. *Automated Software Engineering* 8(2), 131–137 (2001)
4. Kitzelmann, E., Schmid, U.: Inductive synthesis of functional programs: An explanation based generalization approach. *Journal of Machine Learning Research* 7(February), 429–454 (2006)

5. Olsson, J.R.: Inductive functional programming using incremental program transformation. *Artificial Intelligence* 74(1), 55–83 (1995)
6. Esposito, F., Malerba, D., Lisi, F.A.: Induction of recursive theories in the normal ILP setting: Issues and solutions. In: Cussens, J., Frisch, A.M. (eds.) *ILP 2000*. LNCS (LNAI), vol. 1866, pp. 93–111. Springer, Heidelberg (2000)
7. D. Malerba, A. Varalro, M.B.: Learning recursive theories with the separate-and-parallel conquer strategy. In: *Proceedings of the Workshop on Advances in Inductive Rule Learning in conjunction with ECML/PKDD*, pp. 179–193 (2004)
8. Flener, P.: Inductive logic program synthesis with Dialogs. In: Muggleton, S. (ed.) *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pp. 28–51. Stockholm University, Royal Institute of Technology (1996)
9. Muggleton, S., Feng, C.: Efficient induction of logic programs. In: *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pp. 368–381. Ohmsma, Tokyo, Japan (1990)

Training on the Job — Collecting Experience with Hierarchical Hybrid Automata*

Alexandra Kirsch and Michael Beetz

Technische Universität München

Abstract. We propose a novel approach to experience collection for autonomous service robots performing complex activities. This approach enables robots to collect data for multiple learning problems at a time, abstract it and transform it into information specific to the learning tasks and thereby speeding up the learning process. The approach is based on the concept of hierarchical hybrid automata, which are used as expressive representational mechanisms that allow for the specification of these experience-related capabilities independent of the program itself.

1 Motivation

Our vision is to build general purpose service robots that do not have to be pre-programmed for every task variation and every new environment. Such robots should be capable of improving and adapting themselves doing training on the job. If their jobs are complex activities such as household chore they have to do these optimizations in little training time and with sparse experience. In such settings being effective in the collection of experiences is an essential precondition for successful robot learning. To achieve training time efficiency robots must decompose their overall learning task into sets of modular and specific learning tasks that address detected flaws in their behavior. Learning task specific experiences for these subtasks that capture the interaction of the robot’s behavior and the context conditions have to be collected concurrently. Finally, collected experiences must be stored, managed and transformed into abstract descriptions that speed up the learning process.

An instance of this vision is a household robot (see figure [1](#)) equipped with default plans for setting the table, cleaning, etc. It has basic skills for manipulating and navigating as well as default plans — general purpose plans that are developed to be applicable in a variety of environments and to be stable and robust in most cases — for its high-level tasks. However, the plans can fail or operate suboptimally if the robot’s lower-level routines are not adapted to the specific environment. For example, when objects are to be taken out of cupboards or drawers the robot might encounter difficulties in a new kitchen, e.g. when it hasn’t positioned itself appropriately for the manipulation task.

* The research reported in this paper is partly funded by the cluster of excellence CoTeSys (www.cotesys.org) and the DFG under the project “Semi-automatic Acquisition of Visuo-motoric plans” (SPP 1125).

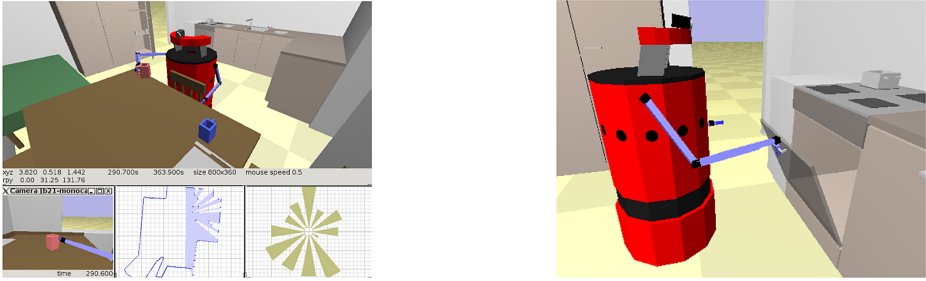


Fig. 1. Kitchen scenario with a simulated B21 robot equipped with a camera, laser and sonar sensors

The robot can recognize such failures and locate its source and execution context inside the program. So when it detects that the manipulation task fails strikingly often, it puts it on the agenda of problems that have to be learned. The experience for all of those unsolved learning problems are then gathered while it performs its household chores. One of the difficulties here is that experience sometimes is only desired in a certain context. For example, when the robot detects that the navigation routine performs different while it is cleaning the floor (presumably because the floor is wet), it is only interested in experience about the navigation routine while cleaning the floor. Other navigation tasks are of no interest at all in this situation.

As learning takes place at various levels of abstractions, it does not suffice to simply log all sensor data and control signals. The learning systems also need information about the program state, such as local variable values. For example, to learn informative causal models of gripping actions the learning mechanism might need information about the scene — its clutteredness and the existence of reaching corridors. Or it might need information about which hand was used and why, or the position where the robot intended to grasp the cup. This data is learning task specific and typically not contained in the low-level sensor and control data.

To account for the robot's behavior being a result of its interaction with the environment, meaningful learning experience can only be obtained with a context-specific experience acquisition. We also need a compact and powerful behavior abstraction by segmenting continuous behavior into states that are meaningful with respect to the learning task. Furthermore, the abstraction of experience for learning should be possible on different levels.

In this paper we propose hierarchical hybrid automata (HHA) as a basic mechanism to fulfill these criteria. We propose HHAs to collect experience and learn from it that allow for

- the modular specification of experience collection independent of primary activity. This is necessary, because the interesting points in the control program can be distributed within the code and because our control program is modified by plan transformations.

- anchoring experience collection into the control program such that program execution automatically effectuates state transitions in the automaton.
- the specification of the problem specific data to be recorded, such as properties of objects being grasped or context conditions such as clutter.
- abstraction of behavior into learning problem specific information, for example the duration of the activation of a state or the number of failures encountered.

Our approach is implemented in RPL (Reactive Plan Language) [1], a programming language for reactive plans implemented as a set of LISP macros. It provides high-level programming constructs like loops and conditionals as well as process control (parallel execution, mutual exclusion, etc.). The constructs for acquiring experience are embedded in a language called RoLL (Robot Learning Language), which is based on RPL.

2 Approach

For an illustration of our approach see Figure 2.

The first step is to define the required experience by specifying execution episodes and data associated with it. We model this information as an experience automaton, which is a hierarchical hybrid automaton where the episodes are modeled as discrete states with continuous behavior.

We anchor the experience automaton to the control program in such a way that it starts accepting an interpretation stream when the program task matching the automaton starts in order to interpret the sensor and execution data in

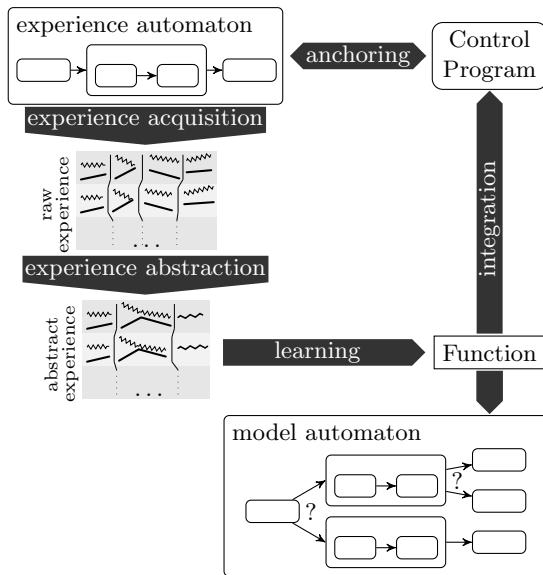


Fig. 2. Overall procedure for learning with RoLL

the context of the experience. Because the behavior that generates the episodes emerges from different parts of the program, it is not practicable to add the automaton code at the specific places in the program. Rather, we run a process parallel to the primary activity that is anchored in the control program and in this way can detect state transitions and observe the desired data.

The automaton returns experiences, i.e. data traces structured along the definition of the specified automaton. These experiences contain all information for understanding the interaction of the program and the environment needed for the respective learning task. In Figure 2 the experience data is structured according to the automaton structure and possible data sources are the robot's internal variables (indicated as bars) and external variables (indicated as zigzagging lines). After the acquisition, the experiences are stored in a database, which facilitates the retrieval, but also purification of the data with data mining tools.

Before starting the learning process, the experience must usually be converted to more abstract features. This can be regarded as a transformation of the hierarchical hybrid automaton of the experience acquisition process to a more abstract form. Figure 2 illustrates this by showing the possible abstraction steps, changing the automaton structure (the data of the two automata in the middle are combined into one) and combining data (as in the data of the third automaton, where robot and environment data is merged).

Finally, the whole learning process can be regarded in the light of HHA. We can build models of the robot's control program with probabilistic hierarchical hybrid automata (PHHA). PHHA are HHA with the difference that state transitions are not activated by conditions, but rely on a probability distribution. After learning from experience, the program model can be replenished with the probability distributions of state transitions.

With the techniques presented here, we can for example observe the activity of lifting an object whose weight is greater than 2 kg. The respective automaton would be active as soon as a gripping action starts and the gripped object fulfills the specification. During the execution of the action, several interesting pieces of data can be observed, e.g. the required time, failures, or if both grippers are used or only one.

References

1. McDermott, D.: A reactive plan language. Technical report, Yale University, Computer Science Dept. (1993)
2. Kirsch, A., Beetz, M.: Combining learning and programming for high-performance robot controllers. In: *Autonome Mobile Systeme* (2005)
3. Kirsch, A., Schweitzer, M., Beetz, M.: Making robot learning controllable: A case study in robot navigation. In: *Proceedings of the ICAPS Workshop on Plan Execution: A Reality Check* (2005)
4. Beetz, M., Kirsch, A., Müller, A.: RPL-LEARN: Extending an autonomous robot control language to perform experience-based learning. In: *3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS)* (2004)

Selecting Users for Sharing Augmented Personal Memories

Alexander Kröner, Nathalie Basselin, Michael Schneider¹, and Junichiro Mori²

¹ German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany

`firstname.lastname@dfki.de`

<http://www.dfki.de/sharedlife/>

² University of Tokyo

7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

`jmori@mi.ci.i.u-tokyo.ac.jp`

Abstract. Dense records of user actions allow an intelligent environment to support its user with an augmented personal memory. In this article, we report on task-oriented user studies concerning mechanisms for sharing such memories, and show how the structure of a social network can be exploited in order to extend the resulting sharing approach.

1 Sharing Augmented Personal Memories

Human decision making usually takes not only the decision maker’s personal experiences into account, but also experiences made by other persons. This behavior can be supported by a personal assistant with access to a digital memory. For instance, Gemmell et al. describe how a memory of documents allows its owner to create stories about past occurrences [1]. This idea can be extended in order to support users during the planning of future actions. For instance, by combining blogs with a trust-based recommender, a user may be supported on the basis of trusted blogs (cf. [2]). In this article, we report on research conducted in SharedLife (BMBF grant 01 IW F03), which is focusing on user support for sharing and exploiting *augmented personal memories*. This ongoing work is based on experience records, which are automatically created for single users by an instrumented environment. Here, we think of an “experience” as a user action, involved objects, contextual information (e.g., location, time), and personal annotations (e.g., ratings, written comments).

The system is tested in a grocery shopping and cooking environment (see Figure 1 and 3), which supports its users in sharing information such as recipes, personal cooking instructions, and experiences with ingredients. A user controls most sharing processes by means of a mobile device. In order to learn about potential users’ preferences regarding these mechanisms, we conducted a focus group study with 23 students (around 23 years old) from a school of engineering. The study consisted of mockups of user interfaces for more or less automated sharing of experiences, examples of sharing occasions, questionnaires, and a discussion round. Its participants understood the need for support in handling a



Fig. 1. Building an augmented memory in a shopping (left-hand side) and cooking (right-hand side) environment. In the middle: events tracked by means of RFID.

large number of requests and appreciated automated methods for reducing the need of manual intervention. Some participants expressed concerns regarding the efforts needed to configure such services, especially in a “cold start” situation where a new system needs to be personalized. Others worried about losing control of their augmented memories if the set of constraints affecting the sharing process becomes complex. In order to deal with these issues they favored mechanisms such as access rights for user groups, and voted strongly for the possibility to make exceptions from such general approaches to protect selected data (e.g., to hide events which could be misinterpreted by a recipient).

We used the participants’ comments to guide the implementation of a peer-to-peer sharing mechanism. Incoming requests are recorded in the user’s augmented memory, so that the user (and the system as well) may handle them at any time. Here, one option for the user is to inspect requested experiences, exclude selected ones, and then reply manually. This approach is in compliance with comments from our study concerning the option to make exceptions from regular response behavior, either to protect certain data without ignoring a request completely, or to eliminate irrelevant information¹. This manual treatment is complemented by rules, which can be created to trigger automated sharing behavior (e.g., reply, dismiss, notify user) in response to similar requests in the future. The precondition of such a rule consists in features such as questioner, location, action type, action patient, which are extracted from experiences in the user’s memory matching the request. The user may verify these features’ impact on a potential response and edit them if desired.

2 Selecting Sharing Partners

These mechanisms allow our users to exchange experiences—but they don’t assist them in the challenging task of finding “sharing partners” actually able and willing to answer an information need. A contextual inquiry regarding general properties of our scenario with 4 persons indicated the special value of food profiles for this problem: countries of origin, religion, health issues and dietary constraints are attributes upon which “relationships by common property” can

¹ As pointed out by Consolvo et al., if people are willing to share information, then they are usually interested in providing information which is useful for the questioner [4].

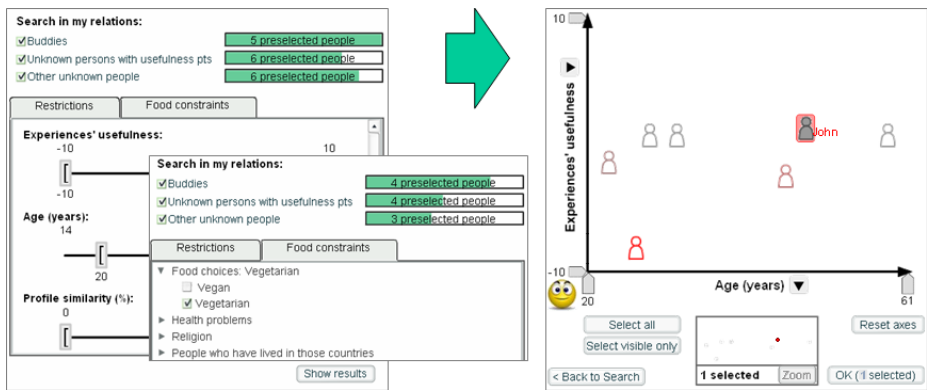


Fig. 2. A user interface for discovery and selection of sharing partners. The screen on the left-hand side enables the specification of initial selection constraints, while the screen on the right-hand side allows exploring and refining the selection.

be specified. Matsuo et al. assume that people sharing common properties are implicitly related and form sub-communities [5]. Relations based on common environments or artefacts (stores, regions, products, or dishes) also belong to this category of relationships. Other relationships result from the community members' communication. In our case, such "relationships by communication" [5] imply that the community evolves through the exchange of experiences: with time, individuals will build relationships based on their evaluation of the received experiences. To some extent, these relationships may be extracted from a user's augmented memory, where we store records of each sharing process.

In order to design a user interface for selecting sharing partners, we conducted two iterations of design and evaluation. We implemented four prototypes, all offering the same possibilities of selection criteria based on information from food profiles (e.g., "is vegetarian"), some other relationships by common properties (e.g., age or physical proximity) and relationships by communication (e.g. number of experiences exchanged). In each design, potential partners were grouped into three categories: the user's buddies (i.e. people known from the user in the real world), "familiar strangers" (people the user has exchanged information with in the past, but who are actually unknown to her), and other candidates (e.g., other people nearby). For the prototypes' design we chose techniques known from community-based applications (buddy lists, social networks) and from information visualization (e.g., data clouds, fish eye view effect).

Participants had to perform the same tasks (constraining a candidate set of users) with all of the prototypes in different orders. They performed best with an interface composed from regular widgets (e.g., checkboxes, sliders). However, while well-suited for the selection task, that interface does not support the user in exploring the candidate set, which is reflected by a participant's comment: "not enough information about the persons is available". Therefore, we combined this prototype with a two-dimensional plot in which sharing candidates are distributed according to two customizable dimensions (see right-hand side of

Fig. 2). A second evaluation with 6 participants attested that the combination was better than the previous prototypes.

Our selection approach relies on profiles of people known to the user. While these profiles may be refined in the daily information exchange, there is still the question of how to select new contacts and create initial profiles. Therefore, we made an attempt to exploit information explicitly expressed by members of a social network service for user selection support in our ubiquitous computing setting. The goal was not only to reuse information previously specified by the user, but also to exploit the network's structure in order to obtain sharing candidates, which are socially close to the user (and thus more likely willing to reply) and which are considered by the community to be experts regarding a set of task-specific constraints. Our prototype extracts from HTML pages of BakeSpace (<http://bakespace.com>) information such as user profiles, friend lists, and recipes. When the system is queried for an expert (e.g., for vegetarian dishes), it searches the user's social network and computes the expertise of neighboring users according to relevancy to the given query based on a probabilistic language model [6]. To find an expert who is socially close to a user, we employ Breadth First Search due to its strength of finding the target closest to the source (which matches our requirement to find socially close experts). In addition, we consider the network centrality of an expert. In social network analysis, several ways to measure centrality for sociological interpretation of network structure have been proposed [7]. We employ the simplest measure, called *degreeness*, which is based on the number of a node's links. Here, our assumption is that if an expert has less links, he or she can be more accessible than other experts who have many links therefore are overloaded with many seekers.

References

1. Gemmell, J., Aris, A., Lueder, R.: Telling stories with MyLifeBits. In: ICME 2005 (2005)
2. Avesani, P., Massa, P., Tiella, R.: A trust-enhanced recommender system application: Moleskiing (2004)
3. Schneider, M.: The Semantic Cookbook: Sharing Cooking Experiences in the Smart Kitchen. In: Proceedings of the 3rd International Conference on Intelligent Environments (IE'07) (to appear, 2007)
4. Consolvo, S., Smith, I.E., Matthews, T., LaMarca, A., Tabert, J., Powledge, P.: Location disclosure to social relations: why, when, & what people want to share. In: CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, pp. 81–90. ACM Press, New York (2005)
5. Matsuo, Y., Hamasaki, M., Mori, J., Takeda, H., Hasida, K.: Ontological consideration on human relationship vocabulary for FOAF. In: Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and Semantic Web (2004)
6. Balog, K., Azzopardi, L., Rijke, M.: Formal models for expert finding in enterprise corpora. In: Proceedings of SIGIR'06 (2006)
7. Freeman, L.C.: Centrality in social networks: Conceptual clarification. *Social networks* 1, 215–239 (1979)

Semantic Reflection – Knowledge Based Design of Intelligent Simulation Environments

Marc Erich Latoschik

AI & VR Lab, AI Group, Bielefeld University
marcl@techfak.uni-bielefeld.de

Abstract. This paper introduces Semantic Reflection (SR), a design paradigm for intelligent applications which represents applications' objects and interfaces on a common knowledge representation layer (KRL). SR provides unified knowledge reflectivity specifically important for complex architectures of novel human-machine interface systems.

1 Introduction

A principle found in intelligent virtual environments [4] is a semantic representation of scene content [2,3,5,7,8]. Reflecting semantic information on a dedicated KRL has shown to be beneficial for several domains of computational intelligence, from novel – e.g. multimodal – man-machine interactions to virtual agents, or advanced computer games. Semantic models strongly influence recent semantic web efforts and have also gained interest in OOP [6] as enriched representations for object reflection. In the software engineering domain, recent approaches explore the usefulness of ontologies to describe the engineering process of complex systems [1].

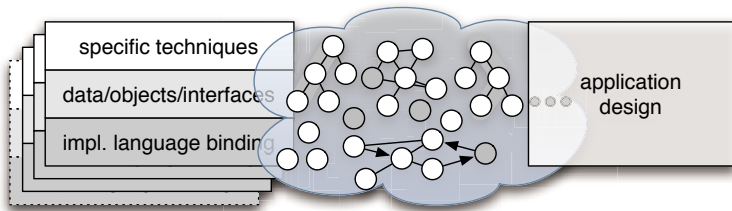


Fig. 1. Semantic Reflection reflects objects, interfaces, and techniques from various modules on a unified knowledge representation layer for high-level application design

Semantic Reflection. (SR) combines and advances these directions. It is a design principle based on a unified semantic description and implementation layer for modular but closely coupled applications which require a built-in knowledge layer support. Integrated into a modular architecture as illustrated in figure 1, SR first establishes a semantic binding to the implementation languages of given modules. Second, it reflects a module's low level design primitives, i.e., the chosen

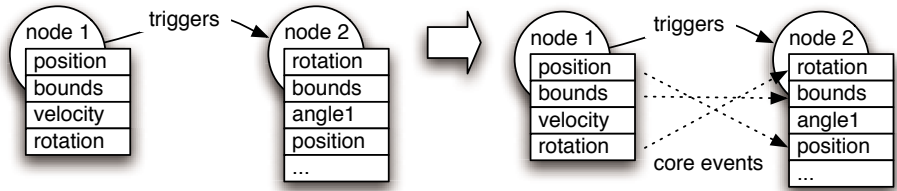


Fig. 2. Semantic representation of application graph technique. Each object is reflected by a semantic net node. A new **triggers** relation is defined which denotes a required event propagation between the objects. The triggers relation is implemented by the FESN’s core event system by establishing core event connections between compatible slots of the connected nodes (see right side).

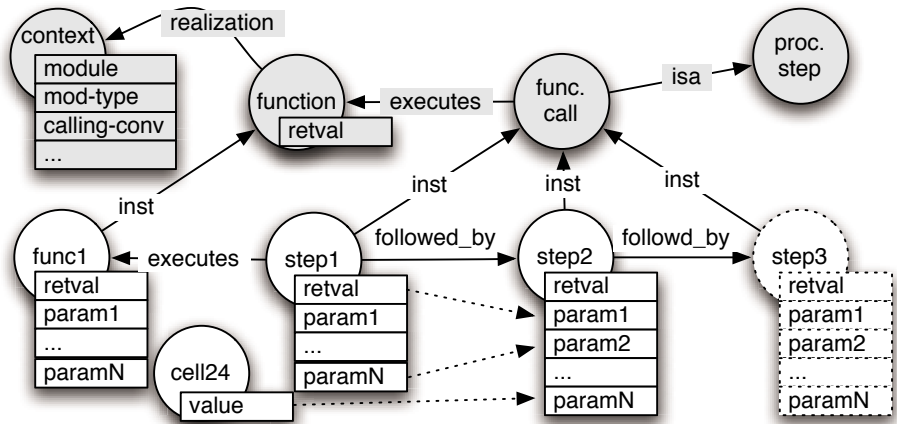


Fig. 3. Basic scheme of function call representation. Functions are lifted to the KRL where they serve as execution targets for processing steps (terminological knowledge represented as grey nodes). The processing steps representing function calls carry a list of slots for their return values and parameters according to the associated functions. The core event system is used to a) define the sources for required parameters as well as b) to finally trigger function execution. Decomposition into function description and processing steps allows for multiple arbitrary processing chains. Independent storage cells are used to insert arbitrary parameters into the call chain between the functions.

data structures, objects, and interfaces. Third, it reflects the modules’ particular techniques like state machines, scene graphs, application graphs etc. To provide Semantic Reflection as a central design feature, a dedicated Functional Extendable Semantic Net (FESN) base formalism provides performance optimizations like hashing and an internal core event system for implementing the procedural semantics for the reflected techniques and interface bindings.

2 Example Module Techniques and Interfaces

Figure 2 illustrates how the base formalism is used to design an application graph, a specific technique for a message system with limited types of events and a fixed chronology of event processing, on the KRL: triggers-relations are directed relations between two nodes. The procedural semantics for the assignment of triggers-relations between nodes is as follows: Core FESN-event connections are established between all compatible slots of connected nodes (see dotted lines in figure 2). This behavior is conveniently implemented using the FESN’s functional extensibility by deriving the triggers-relation from the base relation and then redefining its assert method. It is to the developers choice to implement different semantics, e.g., application designers might desire field connections to be established per field and not per object.

There are situations where either the provided modules’ techniques are insufficient in terms of a required inter-module data exchange or such dedicated techniques are plainly not provided and modules require direct access to the other modules’ interfaces. Hence, modules can export their interface to provide

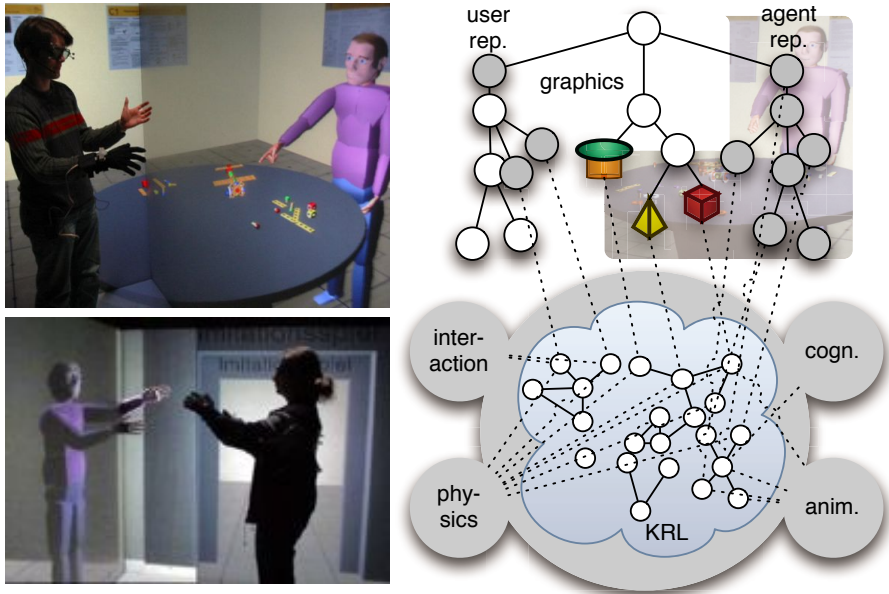


Fig. 4. Left: Interaction with a virtual agent. The agent’s perception components automatically access user interactions in the context of the current environment. For example, the agent’s vision is implemented as a view sensor monitoring the scene. Using semantic reflection, the agent derives the semantics of what he sees. His higher cognitive processes use this knowledge to further process incoming percepts and to generate appropriate multimodal responses.

interface reflection on the semantic layer for a high-level definition of function call execution as illustrated in figure 3.

3 Conclusion

Semantic reflection has proven to be extremely useful in recent application designs implemented at our lab. It is a novel design paradigm which effectively supports the development of complex but on the other hand extensible and reusable components. As one example, the left pictures in figure 4 are taken during interaction scenes with a virtual agent. The right side illustrates, how semantic reflection of the graphical scene is utilized to reflect the agent's perception as well as the agent's and the user's interaction.

Acknowledgment. This work is partly supported by the DFG under grant Wa 815/2 and the EU project PASION under contract number 27654 in FP6-IST.

References

1. Calero, C., Ruiz, F., Piattini, M.: *Ontologies for Software Engineering and Technology*. Springer, Heidelberg (2006)
2. Kalogerakis, E., Christodoulakis, S., Moutoutzis, N.: Coupling ontologies with graphics content for knowledge driven visualization. In: *Proceedings of the IEEE VR2006*, pp. 43–50. IEEE Computer Society Press, Los Alamitos (2006)
3. Latoschik, M.E., Schilling, M.: Incorporating VR Databases into AI Knowledge Representations: A Framework for Intelligent Graphics Applications. In: *Proceedings of the Sixth International Conference on Computer Graphics and Imaging, IASTED*, pp. 79–84. ACTA Press (2003)
4. Luck, M., Aylett, R.: Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments. *Applied Artificial Intelligence* 14(1), 3–32 (2000)
5. Lugin, J.-L., Cavazza, M.: Making Sense of Virtual Environments: Action Representation, Grounding and Common Sense. In: *Proceedings of the Intelligent User Interfaces IUI'07* (2007)
6. Meseguer, J., Talcott, C.: Semantic models for distributed object reflection. In: Magnusson, B. (ed.) *ECOOP 2002*. LNCS, vol. 2374, pp. 1–36. Springer, Heidelberg (2002)
7. Peters, S., Shrobe, H.: Using semantic networks for knowledge representation in an intelligent environment. In: *PerCom '03: 1st Annual IEEE International Conference on Pervasive Computing and Communications*, Ft. Worth, TX, USA, March 2003, IEEE Computer Society Press, Los Alamitos (2003)
8. Soto, M., Allongue, S.: Modeling methods for reusable and interoperable virtual entities in multimedia virtual worlds. *Multimedia Tools Appl.* 16(1-2), 161–177 (2002)

Prolog-Based Real-Time Intelligent Control of the Hexor Mobile Robot*

Piotr Matyasik, Grzegorz J. Nalepa, and Piotr Zięćik

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
ptm@agh.edu.pl, gjn@agh.edu.pl, kosmo@agh.edu.pl

Abstract. The paper presents a concept of an intelligent control platform for the Hexor mobile robot, based on the XTT knowledge representation method for rule-based systems. The control systems is implemented in Prolog, with use of the Embedded Prolog Platform. The paper presents real-time control capabilities provided by this solution.

1 Introduction

The paper presents a research in the area of intelligent control of embedded real-time systems. A knowledge-based hierarchical control platform has been developed for the *Hexor* mobile robot. The platform combines a high-level control logic expressed with use of the XTT-based representation [1], and an embedded runtime environment using the *Embedded Prolog Platform* [2]. In the paper practical enhancements of the XTT towards effective control of reactive systems in real-time are proposed.

2 The Hexor Platform

HexorII is an autonomous 6-legged intelligent robot, developed by *Stenzel* (www.stenzel.com.pl) company as a didactic platform. It has a modular construction and can be easily extended by additional components (e.g. compass, laser sensors, etc.). The company provides a simple Basic-based software development environment. However, *HexorII* lacks some features needed for advanced control algorithm development. The transmission protocol implemented in Hexor does not allow to send data independently. Environmental information can only be pooled from the robot by the host software. Writing microcontroller software with a single control loop in Basic is easy. Unfortunately, this approach results in performance loss and *domino effect* while modifying subsystems.

Because of the software platform limitations, a new Hexor's internal controlling software architecture is proposed. Simple Basic program with one control loop was replaced by a real-time embedded operating system. High-level control logic is knowledge-based with use of the XTT representation.

* The paper is supported by the Hekate Project funded from 2007–2009 resources for science as a research project.

3 New Knowledge-Based Control Platform Architecture

Several possibilities were taken into consideration while developing the new Hexor low-level software. The first one was a dedicated application running directly on hardware. This one was abandoned because of problems with controlling whole hardware with a complicated state machine for each task, running off an interrupt timer. Thus embedded operating system became an obvious choice.

New Hexor controlling software is written on top of FreeRTOS (www.freertos.org). It is an open source portable hard real-time operating system. It can run with small memory footprint (700 bytes for OS, depending on configuration). The new *HexorNG* software architecture (Fig. 1) features: a multi-layer, easy extensible design, an intelligent knowledge-based control, based on the XTT rule-based representation, ability to distribute computations between many processing units, reliable real-time operating system based hardware control.

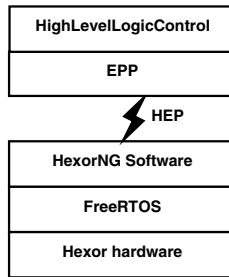


Fig. 1. HexorNG system architecture

HexorNG software substitutes all of the functions of the original one (low-level hardware control, movement execution, sensors reading, communication). The intelligent behavior of the controlled system can be described using a high-level, rule-based model. Following assumptions are taken into consideration in constructing the knowledge base model: a top down approach, visual form, hierarchical structure, ability to verify system properties, high-level of abstraction from the hardware, distinguishing *time* as a special attribute for time-oriented verification and time-related behavior.

The knowledge base is described using the *XTT* (*eXtended Tabular Trees*) method (1). It provides implementation agnostic approach for rule-based systems (3,4), and allows for fast prototyping of the knowledge-based models with Prolog. A very important feature of the *XTT* method is the visual, hierarchical form of representation. *XTT*-based development allows for fast implementation of new, high level control algorithms. Moreover, with its ability to verify knowledge during the design phase, it allows for avoiding problems related to system completeness, determinism, or optimization. Development of knowledge-based control system in *XTT*-based environment is presented on Fig. 2. *XTT* representation allows for generating a Prolog-based control logic prototype. In order

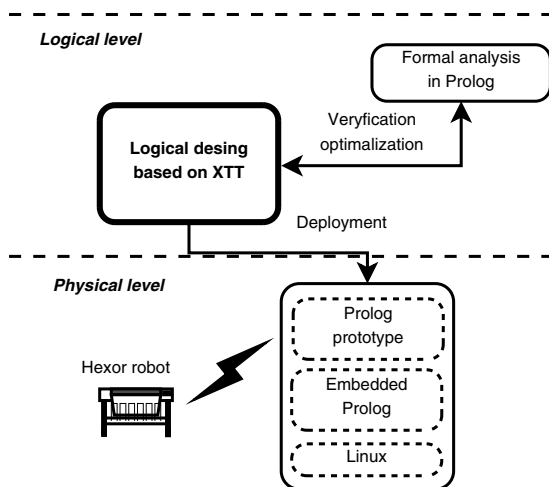


Fig. 2. XTT-based development cycle

to execute this logic, a Prolog interpreting environment must be provided. In this approach the *Embedded Prolog Platform* is used [2].

4 XTT Enhancements and Real-Time Control Features

XTT lacks some features needed for control applications like event handling and interchanging information with controlled object. Originally *XTT* execution starts from a single entrance point. Handling multiple events forced to change this feature. Below an extension to *XTT* is introduced based on the following postulates: every event is represented as *tabular tree*, every event is processed separately but may share attributes with the others, events can be of two kinds: asynchronous (coming from environment, for example if sensor finds obstacle), and synchronous (generated by the timers). Events are executed according to the assigned priorities, events with higher priorities are executed before those with lower ones, events with the same priorities are queued with FIFO policy.

Extended graphical notation is presented on Fig. 3. Program consist of two kinds of *XTT* table sets. The first one is responsible for handling the timed events (above). It is represented by marking a starting point with a clock icon. After the slash in event name a priority is specified. Second one handles the asynchronous events. Starting points of those events are marked with an arrow and also contain name and priority information.

Embedded Prolog Platform (EPP) [2] introduces real-time capabilities and hardware drivers for Prolog. *EPP* consists of a Prolog layer, with an interpreter for executing control logic code, a supervising middleware, for connecting the interpreter with the operating system, and an operating system itself, general purpose or embedded for interfacing hardware, possibly with real-time extensions.

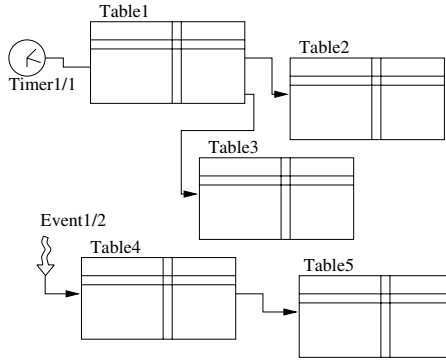


Fig. 3. XTT event handling control logic view

EPP it implements events as asynchronous calls to Prolog, that can be used in similar way as the interrupts. But events in EPP can carry data, and provide bi-directional communication with Prolog.

5 Future Work

The research presented in this paper should be considered a work in progress. Future work will be concentrated on improving the visual representation of the knowledge in editor, and more EPP integration. The original contribution of this paper includes the concept of using an embedded Prolog-based logic for real-time control of reactive systems, and the practical enhancements of the XTT knowledge representation method towards effective control of such systems.

References

1. Nalepa, G.J., Ligeza, A.: A graphical tabular model for rule-based logic programming and verification. *Systems Science* 31(2), 89–95 (2005)
2. Nalepa, G.J., Zięcik, P.: Integrated embedded prolog platform for rule-based control systems. In: Napieralski, A. (ed.) *MIXDES 2006: MIXed DESign of integrated circuits and systems: proceedings of the international conference, Gdynia, Poland, 22–24 June 2006, Łódź, Technical University Lodz. Department of Microelectronics and Computer Science*, pp. 716–721 (2006)
3. Liebowitz, J. (ed.): *The Handbook of Applied Expert Systems*. CRC Press, Boca Raton (1998)
4. Ligeza, A.: *Logical Foundations for Rule-Based Systems*. Springer, Heidelberg (2006)

Improving the Detection of Unknown Computer Worms Activity Using Active Learning

Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher,
Roman Englert, and Yuval Elovici

Deutsche Telekom Laboratories at Ben-Gurion University,
Be'er Sheva, 84105 Israel
{robertmo,nirni,stopel,clint,englert,elovici}@bgu.ac.il

Abstract. Detecting unknown worms is a challenging task. Extant solutions, such as anti-virus tools, rely mainly on prior explicit knowledge of specific worm signatures. As a result, after the appearance of a new worm on the Web there is a significant delay until an update carrying the worm's signature is distributed to anti-virus tools. We propose an innovative technique for detecting the presence of an unknown worm, based on the computer operating system measurements. We monitored 323 computer features and reduced them to 20 features through feature selection. Support vector machines were applied using 3 kernel functions. In addition we used active learning as a selective sampling method to increase the performance of the classifier, exceeding above 90% mean accuracy, and for specific unknown worms 94% accuracy.

Keywords: Classification, Active Learning, Support Vector Machines, Malcode Detection.

1 Introduction

The detection of malicious code (malcode) transmitted over computer networks have been researched intensively in recent years. One type of abundant malcode is *worms*, which proactively propagate across networks while exploiting vulnerabilities in operating systems or in installed programs. Nowadays, excellent technology (i.e., antivirus software packages) exists for detecting *known* malicious code, typically, through detection of known signatures. Nevertheless, it is based on prior explicit knowledge of malcode signatures rendering helpless against unknown malcode. This solution has obvious demerits, however, since worms propagate very rapidly. [1].

Intrusion detection, commonly done at the network level, a more local technique is *Host-based Intrusion Detection (HIDs)*. Our suggested approach can be classified as *HIDs*, but the novelty here is that it is based on the computer behavior, reflected by the operating system measurements, in which the knowledge is acquired automatically based on a set of known worms. In a previous study we performed this task using several classification algorithms [2]. In this study we employ Support Vector Machines (SVM), which are known in their outperforming capabilities in binary classification tasks. Active Learning is commonly used to reduce the amount

of labeling required from an expert – often time consuming and costly. However, in this study, in which all the examples are labeled, we are using the Active Learning approach, as a selective sampling method to improve the classification accuracy.

3 Methods

3.1 Support Vector Machines and Feature Selection

SVM is a binary classifier, which finds a linear hyperplane that separates the given examples of two classes, known to handle large amount of features. Given a training set of labeled examples in a vector format: $x_i = \langle f_1, f_2, \dots, f_m, y_i \rangle$, where f_i' is a feature, and its label $y_i = \{-1, +1\}$. The SVM attempts to specify a linear hyperplane that has the maximal margin, defined by the maximal (perpendicular) distance between the examples of the two classes. The examples lying closest to the hyperplane are the "supporting vectors". The Normal vector of the hyperplane, denoted as w in formula 1, is a linear combination of the supporting vectors, multiplied by Lagrange multipliers (alphas). Often the dataset cannot be linearly separated, thus a kernel function K is used. The SVM actually projects the examples into a higher dimensional space to create a linear separation of the examples. We examined the 3 commonly used kernels: *Linear*, *Polynomial* and *RBF*. Generally, the SVM classifier is in the form shown in formula 1, in which n is the number of the training examples. We used the Lib-SVM implementation [3] which also provides a multiple classification.

$$f(x) = \text{sign}(w \cdot \Phi(x)) = \text{sign}\left(\sum_1^n \alpha_i y_i K(x_i, x)\right) \quad (1)$$

To reduce the amount of features we employed three *filtering* feature selection measures: *Chi-Square (CS)*, *Gain Ratio* [4, 5] (*GR*) and *ReliefF* [6]. After having the ranks for each feature, the top 5, 10, 20 and 30 features were selected, based on each measure and their ensemble.

3.2 Active Learning

Active Learning (A.L) is usually used to reduce the efforts in labeling examples, a commonly time consuming and costly task, while maintaining a high accuracy rate. In *A.L* the learner actively asks to label specific examples from a given pool, which expected to result in a better classifier. In our study all the examples are labeled, however, we employed this approach as a selective sampling method to increase the accuracy. We implemented a pool based active learner which aims to reduce the expected *generalization error*, named *Error-Reduction*, presented by [7], in which, an example is acquired from a pool only if it dramatically expected to improve the confidence of the current classifier over all the examples in the pool. Through the use of a log-loss function the error degree of acquiring a specific example, having a specific label, enables the self-estimation of the mean error. Finally, the example with the lowest self-estimated mean error is selected and added to the training set.

3.3 DataSet Creation

Since there is no benchmark dataset we created a dataset. A controlled network of computers was deployed, into which we could inject worms, monitor and log the computer features. Finally all the data was aggregated into a vector of 323 features for every second. Five available computer worms, representing a variety of worms, were selected: (1) *W32.Dabber.A*, (2) *W32.Deborn.Y*, (4) *W32.Sasser.D*, (5) *W32.Slackor.A*. All the worms perform port scanning and possess different characteristics. A detailed description of the dataset is in [2]. To examine the influence of the machine hardware or software and user activity on the accuracy, we performed the monitoring operation on eight combinations, resulting from three binary aspects: *old* and *new* computer configuration, background activity of software or absent, and User activity or absent. More details on these aspects in [2]. Thus, we had eight datasets including six class examples of the five worms and none activity.

3.4 Evaluation Measures

For the purpose of evaluation we used the *True Positive (TP)* measure presenting the rate of instances classified as *positive* correctly (true), *False Positive (FP)* presenting the rate of *positive* instances misclassified and the *Total Accuracy* – the rate of the entire *correctly* classified instances, either positive or negative, divided by the entire number of instances.

4 Experiments and Results

In the first part of the study, we wanted to identify the best feature selection measure, the best kernel function. In the second part we wanted to measure the possibility of classifying unknown worms using a training set of known worms, and the possibility to increase the performance using selective sampling.

We performed a wide set of experiments, called *e1*, in which we evaluated each kernel function, feature selection and top selection combination to determine the outperforming combination. We trained each classifier on a single dataset *i* and tested iteratively on the other eight datasets. Note, the classification task included five worms or none (worm) activity. The mean performance of the top 20 features (Gain Ratio) runs outperformed, which we used in the next experiments.

4.1 Experiment II – Unknown Worms Detection

To evaluate the capability to classify an *unknown worm* activity, an experiment, called *e2*, was performed. In this experiment the training set consisted of $(5-k)$ worms and the testing set contained the k excluded worms, while the *none* activity appeared in both datasets. This process repeated for all the possible combinations of the k worms, for $k = 1$ to 4 . In these experiments there were two classes: (generally) *worm*, for any type of worm, and *none* activity.

Figure 1 presents the results of *e2*. A monotonic improvement observed, as more worms included in the training set, in which the *RBF* outperformed. However, in specific worms, when a single worm was excluded, 95% accuracy observed.

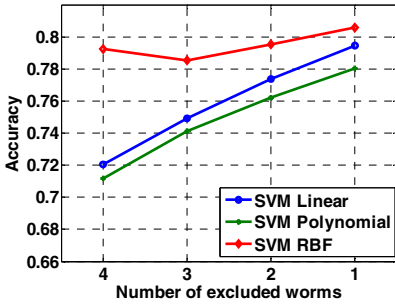


Fig. 1. The performance monotonically increases as fewer worms are excluded (and more worms appear in the training set), RBF kernel has quite different accuracy trend that had one decreasing.

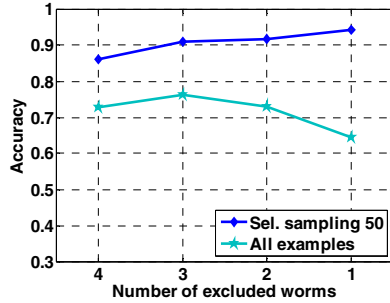


Fig. 2. The considerable improvement when using selective sampling compare to training the SVM on all the examples. When 1 worm was excluded there was almost 30% improvement in accuracy.

4.2 Experiment 3 – Using Selective Sampling

To maximize the performance achieved by the RBF kernel, in *e3* we employed selective sampling methods, using error reduction criterion. Generally, we performed the same experiment as in *e2*, however, the training set included all the eight datasets and the test set only one of them. For the selective sampling process, initially six examples from each class (worms and none) were selected randomly, and then in each AL iteration additional single example selected. Figure 2 presents the results of *e3*. The selective sampling had improved significantly the baseline performance even when 50 samples were selected, as shown in figure 2.

5 Conclusions and Future Work

We presented the concept of detecting *unknown* computer worms based on a host behavior, using the SVM classification algorithm with several kernels. Using feature selection we shown that even with 20 features a high accuracy is achieved. Often the *RBF kernel* outperformed the other kernels. In the detection of unknown worms (*e2*), a high level of accuracy was achieved (exceeding 80% in average); as more worms were in the training set. To reduce the noise in the training set and improve the performance we employed the A.L approach as a selective sampling method which increased the accuracy after selecting 50 examples to above 90% of accuracy and 94% when the training set contained four worms. These results are highly encouraging and show that unknown worms can be detected in real time. The advantage of the suggested approach is the automatic acquisition and maintenance of knowledge, based on inductive learning. This avoids the need for a human expert who is not always available and familiar with the general rules. This is possible these days, based on the existing amount of known worms, as well as the generalization capabilities of classification algorithms. However, in order to detect more sophisticated worms, we develop a temporal classification method.

References

1. Fosnock, C.: Computer Worms: Past, Present and Future. East Carolina University. Kabiri, P., Ghorbani, A.A. Research on intrusion detection and response: A survey. *International Journal of Network Security*, 1(2), 84–102 (2005)
2. Moskovitch, R., Gus, I., Pluderman, S., Stopel, D., Fermat, Y., Shahar, Y., Elovici, Y.: Host Based Intrusion Detection Using Machine Learning. In: *Proceedings of Intelligence and Security Informatics*, May 2007, Rutgers University (2007)
3. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
4. Quinlan, J.R.: *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
5. Mitchell, T.: *Machine Learning*. McGraw-Hill, New York (1997)
6. Liu, H., Motoda, H., Yu, L.: A Selective Sampling Approach to Active Selection. *Artificial Intelligence* 159, 49–74 (2004)
7. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: *Proceedings of ICML-2001, 18th International Conference on Machine Learning*, pp. 441–448 (2001)

The Behaviour-Based Control Architecture iB2C for Complex Robotic Systems

Martin Proetzsch, Tobias Luksch, and Karsten Berns

University of Kaiserslautern, Robotics Research Lab
Gottlieb-Daimler-Str., 67663 Kaiserslautern, Germany
{proetzsch,luksch}@informatik.uni-kl.de
<http://www.agrosy.informatik.uni-kl.de>

Abstract. This paper¹ presents the behaviour-based control architecture iB2C (integrated Behaviour-Based Control) used for the development of complex robotic systems. The specification of behavioural components is described as well as the integration of behaviour coordination and hierarchical abstraction. It is considered how the design process can be supported by guidelines and by tools for development as well as analysis. Finally some application platforms are presented.

Keywords: behaviour-based control, system analysis, system design.

1 Introduction

In the development of complex robotic applications the process of building up a control system should be supported by an adequate methodology to help overcoming difficulties like ensuring secure operation, modularity, or keeping track of a system of growing complexity. Behaviour-based approaches have proven to handle such difficulties rather well but still the problem of controlling complex robotic systems is not solved by this paradigm alone. Difficulties are the coordination of multiple behaviours and the identification of error sources in a control that shows an emergent system behaviour. Additionally there is the matter of how the architecture can help structuring the design process, e.g. giving support in the process of selecting the best set of behaviours and coordinating their action. The architecture proposed in this paper tries to address most of the issues just mentioned.

The problem of controlling complex autonomous robots has resulted in many different kinds of behaviour-based architectures, e.g. [1], [2], [3], and evaluations about the coordination problems have been undertaken [4]. However, the design and analysis aspect is still one of the key issues. In [5] the *Behaviour Oriented Design (BOD)* is proposed as development process for a modular architecture. While this concept reflects procedures on high reasoning levels, there seems to be a lack of low level motion control coordination, whereas iB2C is intended to cover the whole span from low-level to high-level behaviours.

¹ A detailed version of this paper can be found at <http://agrosy.informatik.uni-kl.de/en/publications/>

2 Components of iB2C and Their Interaction

The fundamental unit of the proposed control architecture is the behaviour module (see Fig. 1, left) as already introduced in [6] and [7] and modified towards its present form in [8]. Each atomic behaviour is wrapped into such a module with a uniform interface.

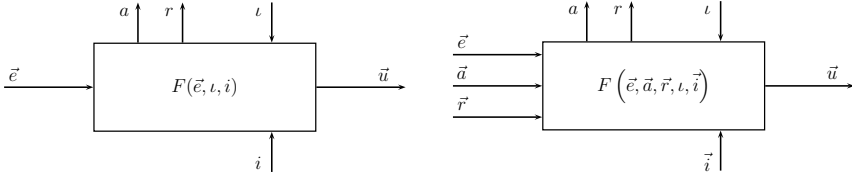


Fig. 1. Basic iB2C behaviour module (left) and fusion behaviour module (right)

Behaviours can be described as three-tuples of the form $\mathcal{B} = (r, a, F)$, where r is the target rating function, a is the activity function, and F is the transfer function of the behaviour. Additionally each behaviour receives an input vector e , an activation ι , and an inhibition i and generates an output vector u .

Coordination within the behaviour network can be achieved by so called fusion behaviours (see Fig. 1, right) which are integrated in the case of competing behaviours. The transfer function of fusion behaviours is the fusion function $f(a, e)$ which processes input values to a merged output control vector u using the activity of the coordinated behaviours.

3 Design Guidelines

In iB2C the development begins by figuring out the relevant degrees of freedom (DOF), e.g. rotation and velocity of a vehicle. Each of the DOF is divided into positive and negative direction, leading to two control data paths for every motion possibility. The conflation of the data flow is accomplished using a fusion behaviour for each of the DOFs. In order to fulfil basic safety requirements behaviours reacting on safety related sensor input are added leading to an interface for higher level behaviours and encapsulating the functionality of a safety behaviour system. High-level behaviours are then added using a top-down task-oriented approach (e.g. as proposed in [5])

In order to simplify the structure and to clarify the functionality iB2C features the concept of hierarchical abstraction using behavioural groups. These are groups in the sense of the embedding programming framework – in this case the *modular control architecture MCA* – i.e. a collection of modules or further groups with a new interface and dedicated connections between group and modules.

The main challenge when coping with systems growing in complexity is making statements about the current system status. In this sense it becomes invaluable having a common interface of behaviours representing their internal state in an abstract way – in iB2C the meta information signals *activity* (a) and *target rating* (r) which can be used for deadlock detection, risk determination, effort, oscillation detection, etc. iB2C also makes extensive use of the Modular Controller Architecture (MCA, [10]) and the provided tools **Builder**, **MCAGUI**, and **MCABrowser** for creation and supervision of behaviour networks. Besides these tools for analysing the behaviour network and the overall system performance it might be necessary to guarantee certain system properties. In this case an approach for the formal verification of a subset of behaviours can be followed as described in [11].

4 Applications

Figure 2 shows some platforms controlled by an iB2C network, reflecting the flexibility of the approach. The complexity of the systems makes it indispensable to apply concepts as described before. The outdoor robot RAVON [8] is intended for rough offroad terrain to fulfil exploration and navigation tasks. The indoor robots ARTOS and MARVIN [12] deal as service robots in home and office environments. ROMAN [13] is a humanoid head for interaction using emotional states. Finally the dynamic control of bipeds is in development exploiting the features of iB2C on a low actuator level.

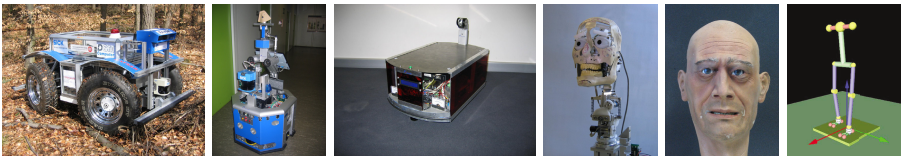


Fig. 2. Robots of the Robotics Research Lab controlled by a iB2C system: RAVON, MARVIN, ARTOS, ROMAN (skeleton and skin) and simulated humanoid biped

5 Conclusion and Future Work

This paper presented a behaviour-based architecture appropriate for the development of a multitude of complex robotic systems. A fixed interface for all behavioural components involved provides a means of abstraction allowing the analysis of the functionality of the behaviour network. The modular characteristic of the architecture increases the reusability of behaviours while the design and analysis of iB2C-networks is facilitated by tools and plugins giving complete access to all relevant information.

Future work includes the extension of tooling functionality and design strategies as well as the investigation about the degree of representation required at different layers of the control system.

References

1. Brooks, R.A.: A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2(1), 14–23 (1986)
2. Mataric, M.J.: Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, Special issue on Software Architectures for Physical Agents 9(2-3), 323–336 (1997)
3. Langer, D., Rosenblatt, J., Hebert, M.: A Behaviour-Based System for Off-Road Navigation. *IEEE Journal of Robotics and Automation* (1994)
4. Scheutz, M., Andronache, V.: Architectural Mechanisms for Dynamic Changes of Behavior Selection Strategies in Behavior-Based Systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics* 2377–2395 (2004)
5. Bryson, J.: Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents. Dissertation, Massachusetts Institute of Technology (September 2001)
6. Albiez, J., Luksch, T., Berns, K., Dillmann, R.: An Activation-Based Behavior Control Architecture for Walking Machines. *The International Journal on Robotics Research* 22, 203–211 (2003)
7. Albiez, J.: Verhaltensnetzwerke zur adaptiven Steuerung biologisch motivierter Laufmaschinen. GCA Verlag (2007)
8. Proetzsch, M., Luksch, T., Berns, K.: Fault-Tolerant Behavior-Based Motion Control for Offroad Navigation. In: 20th IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain, April 18-22, 2005, IEEE Computer Society Press, Los Alamitos (2005)
9. Arkin, R.C.: Moving up the food chain: Motivation and Emotion in behavior-based robots. Technical report, Mobile Robot Laboratory, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA (2003)
10. Scholl, K.U., Albiez, J., Gassmann, G.: MCA – An Expandable Modular Controller Architecture. In: 3rd Real-Time Linux Workshop, Milano, Italy (2001)
11. Proetzsch, M., Berns, K., Schuele, T., Schneider, K.: Formal Verification of Safety Behaviours of the Outdoor Robot RAVON. In: Fourth International Conference on Informatics in Control, Automation and Robotics (ICINCO), Angers, France, May 2007, INSTICC Press (2007)
12. Schmidt, D., Luksch, T., Wettach, J., Berns, K.: Autonomous Behavior-Based Exploration of Office Environments. In: 3rd International Conference on Informatics in Control, Automation and Robotics - ICINCO, Setubal, Portugal, August 1-5, 2006, pp. 235–240 (2006)
13. Berns, K., Hirth, J.: Control of facial expressions of the humanoid robot head ROMAN. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 9-15, 2006, Beijing, China (2006)

Concept for Controlled Self-optimization in Online Learning Neuro-fuzzy Systems

Nils Rosemann and Werner Brockmann

University of Osnabrück, Albrechtstr. 28, 49076 Osnabrück

1 Introduction

Many modern control systems, e.g., in automotive or robotic applications get increasingly complex and hard to design. This is due to the complex interactions of their internal subsystems, but additionally, these systems operate in a dynamically changing, complex environment. The Organic Computing (OC) initiative tries to cope with the resulting engineering demands by introducing emergence and self-x properties into the systems (e.g., self-organization, self-optimization). Within this context, we focus on control systems which adapt their behavior autonomously by learning.

To be more specific, these systems are required to control a complex interaction with the environment online and in a closed loop way. Thus, these systems have to react continuously, in real time, and learning has to occur incrementally. As this closed loop may lead to chaotic system behavior, one needs a strategy of *guiding* the process of learning online which will be presented in this paper.

2 Safe Self-adaptation

2.1 Basic Architecture

In order to be self-adapting, a system needs to correct its behavior solely on the basis of incoming data. But this cannot be done by unsupervised learning (safety reasons) or by supervised learning based on *external* training samples. Our approach is thus to use direct adaptation similar to [1,2]. This architecture tunes the control systems behavior directly by another system which determines the changes required for optimization.

In our case, the actual behavior lies in a zero order Takagi-Sugeno fuzzy system [3,4] with triangular membership functions and a normalized rule base. The conclusions of the individual rules, i.e., non-fuzzy singletons, are changed by learning. A learning stimulus is distributed over all rules contributing to a corresponding wrong output. Because of the triangular membership functions, every learning stimulus only acts locally. This local fuzzy learner can learn quickly to drive the process under control to a certain working point. But as the learning architecture allows an arbitrary systems behavior, it may become chaotic.

¹ <http://www.organic-computing.de>

Algorithm 1. The SILKE algorithm with a sample template applied to the current state s and a local fuzzy learner L with adjustment rate λ .

```

PerformLearning( $L, s$ )
active_rules  $\leftarrow$  all rules of  $L$  which are applicable to state  $s$ 
for all  $a \in$  active_rules do
     $silke_a \leftarrow$  mean value of conclusions of all neighbors of  $a$  {example template}
end for
for all  $a \in$  active_rules do
     $conclusion_a \leftarrow (1 - \lambda) \cdot conclusion_a + \lambda \cdot silke_a$  {adapt rule conclusions}
end for

```

Prior work has shown how to prevent critical system states and how to control learning in non-critical regions [2]. A further approach called SILKE (system to immunize learning knowledge based elements) aims at meta level control of the learning process, e.g, monotony and limited steepness [5].

2.2 Controlling Direct Adaptation with the SILKE Approach

As described above, the learning system is based on a local fuzzy learner with rules forming the knots of a fixed grating over the input space. The basic idea of meta level control of learning is based on the notion of *neighborhood* of the fuzzy rule conclusions. If a certain rule conclusion is not in accordance with its neighbors after a learning step, then it is deemed pathologic. This decision is done by a local operator, the so called *template function*, in order to preserve the local function approximation characteristic.

A template function is defined on the neighborhood of the current active rule and calculates a real number usable to correct this rule. This way, each template represents meta level characteristics which the neighboring conclusions should fulfill. The SILKE algorithm with a sample template is shown in Alg. 1. The following policy is used: The mean value of the neighbors is calculated for every rule which was involved in the last learning step. Afterwards, the conclusions of the investigated rules are corrected towards their specific mean value.

2.3 Extension of the SILKE Approach

Applying the SILKE approach acts similar to regularization because it enforces specific meta properties [6]. For instance, in case of an averaging template like the one used in these experiments, the SILKE approach enforces the learning system to approach a smooth (linear) function.

In order to adapt the SILKE approach to a given application, we introduce the *adjustment rate* λ to steer its effect: the higher λ , the stronger the influence on the learning process. An adjustment rate of 1 causes a complete match with the template policy, and an adjustment rate of 0 switches the SILKE approach off. The adjustment rate is easily incorporated into the basic SILKE approach as Alg. 1 shows. It is a parameter weighting the correction term determined by the

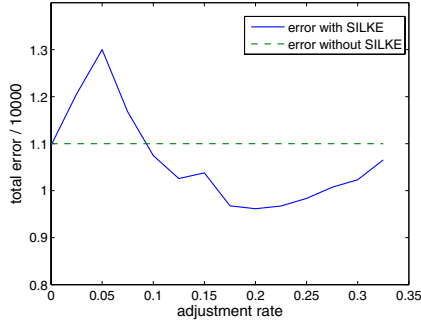


Fig. 1. The sum of the absolute errors between cart position and target position for different adjustment rates over a period of 210 seconds

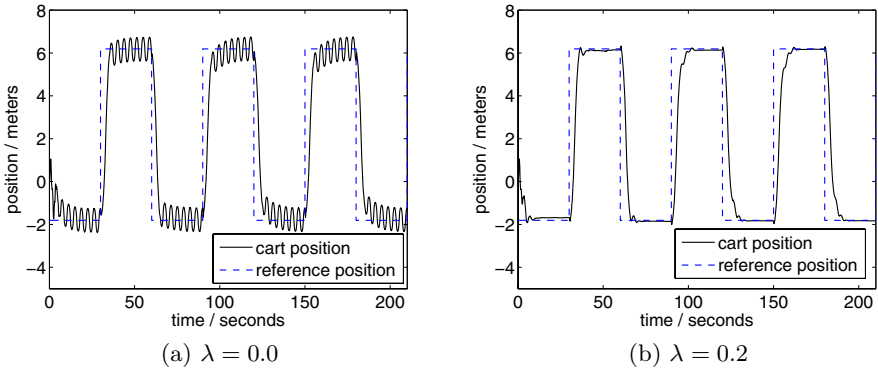


Fig. 2. The cart position (a) without the SILKE approach and (b) with optimal adjustment rate

template in relation to the rule conclusion. By selecting a suited template function and by tuning the adjustment rate, the learning can be improved concerning its dynamics and convergence as the following example demonstrates.

3 Demonstrational Example

To show the effect of controlling online learning by the SILKE approach, we used the same set up as in [5], namely a simulated pole balancing cart. The task of the cart was to balance the pendulum as close as possible to a changing target location s_0 . The control software was composed of two cascaded self-optimizing control subsystems, P and A , both were able to self-adapt independently giving rise to complex (internal) dynamic interactions. The task of P was to map the current speed of the cart and the remaining distance to s_0 to an appropriate target pendulum angle p_0 . The task of A was to map the current angular speed

of the pendulum and the difference between the current pendulum angle and the target angle p_0 to an appropriate motor voltage. The target position s_0 was switched every 30 seconds.

The simulation was repeated with different adjustment rates. For each run, the total absolute position error was accumulated. The effect of λ on this error is shown in Fig. 1. Fig. 2 shows the dynamic behavior of the pole balancing cart without the SILKE approach and for the optimal adjustment rate.

4 Conclusion

The results of the experiment clearly show that the SILKE approach can influence the learning process and its dynamics advantageously. Although it works only locally, the SILKE approach ensures global meta properties. As in this case, even two strongly interacting learning systems can be guided quickly towards a stable solution. Convergence is improved significantly concerning speed and accuracy. By introducing the adjustment rate λ , the effect of the SILKE approach can be optimized by a single global parameter.

Acknowledgments

This project is funded by the German Research Foundation in the framework of the *Organic Computing* priority research program under No. BR 1980/1-1.

References

1. Shao, S.: Fuzzy self-organizing controller and its application for dynamic processes. *Fuzzy Sets and Systems* 26(2), 151–164 (1988)
2. Brockmann, W.: Online Machine Learning For Adaptive Control. *IEEE Int. Workshop on Emerging Technologies and Factory Automation*, pp. 190–195 (1992)
3. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Systems, Man, and Cybernetics* 15, 116–132 (1985)
4. Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: survey in soft computing framework. *IEEE Trans. Neural Networks* 11(3), 748–768 (2000)
5. Brockmann, W., Horst, A.: Stabilizing the Convergence of Online-Learning in Neuro-Fuzzy Systems by an Immune System-inspired Approach. *IEEE Int. Conf. On Fuzzy Systems - FUZZ-IEEE 2007* (to appear, 2007)
6. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. *Neural Computation* 7(2), 219–269 (1995)

LiSA: A Robot Assistant for Life Sciences

Erik Schulenburg¹, Norbert Elkmann¹, Markus Fritzsche¹, Angelika Girstl²,
Stefan Stiene³, and Christian Teutsch¹

¹ Fraunhofer Institute for Factory Operation and Automation, Sandtorstrasse 22,
39106 Magdeburg, Germany

<firstname>.<lastname>@iff.fraunhofer.de

² Sympalog Voice Solutions, Karl-Zucker-Strasse 10, 91052 Erlangen, Germany
girstl@sympalog.de

³ University of Osnabrück, Albrechtstrasse 28, 49069 Osnabrück, Germany
sstiene@informatik.uni-osnabrueck.de

Abstract. This paper presents a project that is developing a mobile service robot to assist users in biological and pharmaceutical laboratories by executing routine jobs such as filling and transporting microplates. A preliminary overview of the design of the mobile platform with a robotic arm is provided. Moreover, the approaches to localization and intuitive multimodal human-machine interaction using speech and touchpad input are described. One focus of the project is aspects of safety since the robot and humans will share a common environment.

Keywords: service robotics, mobile robot, human-machine interaction.

1 Introduction

Biological and pharmaceutical research entails a great deal of repetitive manual work, e.g. preparing experiments or loading equipment such as drying chambers and centrifuges. Classical automation uses band conveyors or indexing tables to interconnect such units. The basic idea behind the Life Science Assistant (LiSA) is to employ a mobile service robot to interconnect equipment. This makes automated experiment cycles flexible, while simultaneously allowing stations to be used for other purposes. In addition, the robot helps employees prepare experiments, e.g. by collaboratively executing transportation tasks or filling microplates. The LiSA project is constructing a demonstrator that executes the aforementioned tasks.

Safety is an important aspect in service robotics since robots and humans share a common environment. Previous projects [1,2] only marginally examined relevant requirements. Safety assumes even greater importance in the life sciences because a robot may handle toxic or hazardous substances. The LiSA project reflects this in its manifold safety sensors.

Figure 1 presents a design study of the particular robot currently under development. The development work will converge in the construction and testing of the final service robot by March 2009.

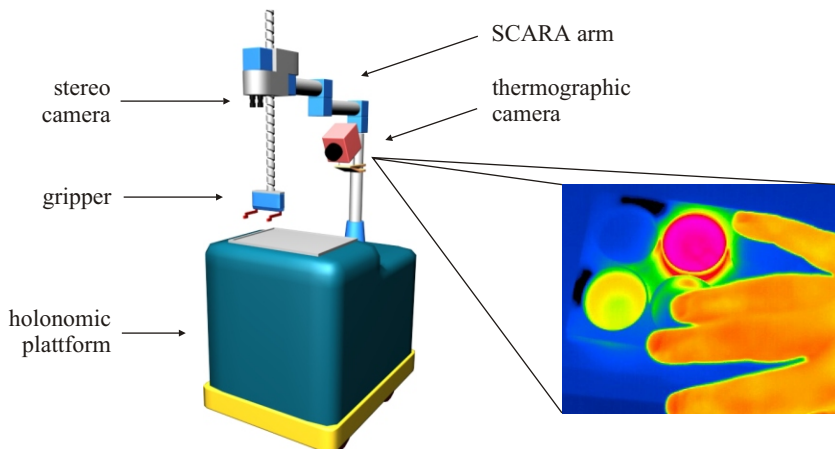


Fig. 1. LiSA platform design study (left) and thermographic image (right)

2 Hardware and Safety Components

The robot assistant's design consists of a custom-built robotic arm mounted on a mobile platform as depicted in Figure 1.

The mobile platform is equipped with a holonomic drive. For navigation and obstacle avoidance it is equipped with a gyroscope, wheel encoders and six 2-D laser scanners. The laser scanners provide an alert area and a protection area. If an obstacle violates the alert field of a laser scanner, the mobile platform slows down. Activating the protection area or one of the bumpers mounted all around the bottom edges of the platform results in an immediate stop.

The robotic arm is covered by a pressure-sensitive artificial skin for collision detection. The SCARA design selected gives the manipulator clearly defined directions of movement (horizontal for the joints, horizontal and vertical for the linear axis). Therefore, tactile sensor elements only have to cover specific areas. Torque measurement and contouring error control are integrated in the joints as additional electronic safety functions. Moreover, the manipulator is padded to prevent injuries in the case of a collision.

The robotic arm is equipped with two camera systems for camera-guided movement. A stereo camera system is installed near the linear axis and a combined camera system is mounted at the base of the robot arm. The stereo camera determines the 3-D pose of exchange positions and the microplates with high precision (< 0.5 mm) based on a photogrammetric approach. Corresponding pixel pairs in both cameras are identified by using statistical correlation between image segments on the epipolar lines [3]. This information is used to guide the robotic arm. The combined camera device consists of two calibrated cameras, one for the infrared and one for the visible spectrum. The thermographic component detects human interaction in front of the robotic arm and its gripper to ensure the safety of the manipulation process (see Figure 1 (right)).

3 Localization and Navigation

For localization the LiSA robot employs a novel sensor configuration that increases safety by enabling it to navigate with full 3-D obstacle avoidance, produced by combining 6 laser scanners to a robot centered 360° 3-D laser scanner as depicted in Figure 2 (left). Two laser scanners (SICK s300) are mounted on opposite corners of the robot. The scanners' 270° field of view generate a 360° field of permanent 2-D view with overlapping regions. This combined 360° scanner is

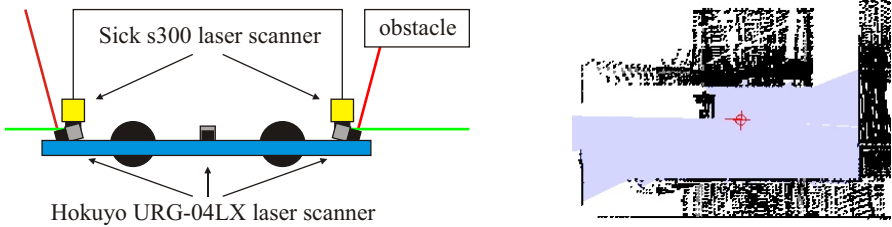


Fig. 2. Laser configuration of the LiSA robot (left) and combined sensor data (right)

used for localization in an a-priori map and to avoid collisions with humans. It is inadequate for general obstacle avoidance, since obstacles may interfere with the robot in its complete bounding box. Thus, the setup is extended by four Hokuyo URG-04LX laser scanners, each of which is mounted at the bottom of one of the robot's sides and angled upward, enabling the robot to detect obstacles in the respective data. If this occurs, the 3-D laser data points (belonging to the obstacle) are projected onto the floor plane and inserted into a local perception map. Thus, the robot generates a detailed perception map while moving (see Figure 2 (right)). For obstacle avoidance, the horizontal localization scanners are combined with the perception map regarding the current robot position.

The complete system has been tested in the robot simulation environment USARSim [4]. The simulation environment is connected to the hardware abstraction layer of Player/Stage [5]. Figure 2 (right) shows the standard player sensor data visualization tool as well as the combined sensor data (blue) and the map generated by the Hokuyo scanners (black). The robot is aware of its entire environment including tabletops. A classical horizontal sensor configuration would only be able to detect the chair and table legs.

4 Multimodal Interaction

Interaction with LiSA is multimodal, i.e. spoken and touchpad input is possible. Speech recognition is speaker-independent. The commercial dialog engine used for LiSA supports mixed-initiative, natural language dialogs and conversation in full sentences. It has been expanded for multimodal input based on experiences from various projects [6,7].

The dialog engine extracts all pieces of information from a spoken utterance and touchpad input and enters them into a predefined XML form, requesting missing pieces of information and forwarding a completed form to the LiSA Task Manager. Spoken commands and touchpad input can be used in combination or independently throughout the dialog. This includes combinations of touchpad input and speech signals in a single utterance, e.g. the sentence “take the sample from this point to that point” is combined with two touchpad input events on the map displayed. The dialog engine interacts with a knowledge database that stores information on the location of laboratory inventory such as the fluorescence reader or drying chambers and their location in the different rooms of the lab. All these features generate intuitive mixed-initiative, multimodal interaction between laboratory assistants and LiSA.

Acknowledgments. This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the Framework Concept “Research for Tomorrow’s Production” (fund number 02PB2170-02PB2177) and managed by the Project Management Agency Forschungszentrum Karlsruhe, Production and Manufacturing Technologies Division (PTKA-PFT).

References

1. Prassler, E., Dillmann, R., Fröhlich, C., Grunwald, G., Hägele, M., Lawitzky, G., Lay, K., Stopp, A., von Seelen, W.: Morpha: Communication and interaction with intelligent, anthropomorphic robot assistants. In: Proceedings of the International Status Conference – Lead Projects Human-Computer-Interactions, Saarbrücken (Germany) (2001)
2. Scherer, T., Poggendorf, I., Schneider, A., Westhoff, D., Zhang, J., Lutkemeyer, D., Lehmann, J., Knoll, A.: A service robot for automating the sample management in biotechnological cell cultivations. In: Emerging Technologies and Factory Automation. Proceedings. ETFA '03. IEEE Conference, vol. 2, pp. 383–390. IEEE Computer Society Press, Los Alamitos (2003)
3. Shi, J., Tomasi, C.: Good features to track. In: Proc. Computer Vision and Pattern Recognition (CVPR'94), pp. 593–600 (1994)
4. Albrecht, S., Hertzberg, J., Lingemann, K., Nüchter, A., Sprickerhof, J., Stiene, S.: Device level simulation of kurt3d rescue robots. In: Third Intl. Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (SRMED 2006). CDROM Proceedings (2006)
5. Gerkey, B., Vaughan, R., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of the International Conference on Advanced Robotics (ICAR 2003), Coimbra, Portugal, June 30 - July 3, 2003, pp. 317–323 (2003)
6. Sonntag, D., Engel, R., Herzog, G., Pfalzgraf, A., Pflieger, N., Romanelli, M., Reithinger, N.: Smart web handheld – multimodal interaction with ontological knowledge bases and semantic web services. In: Proc. International Workshop on AI for Human Computing (in conjunction with IJCAI), Hyderabad, India (2007)
7. Horndasch, A., Rapp, H., Röttger, H.: SmartKom-Public. In: SmartKom: Foundations of Multimodal Dialogue Systems, Springer, Heidelberg (2006)

Semantic Graph Visualisation for Mobile Semantic Web Interfaces

Daniel Sonntag and Philipp Heim

German Research Center for Artificial Intelligence
66123 Saarbrücken, Germany
sonntag@dfki.de

Abstract. Information visualisation benefits from the Semantic Web: multimodal mobile interfaces to the Semantic Web offer access to complex knowledge and information structures. Natural language dialogue systems are ideal interfaces to personal digital assistants (PDAs) or other handheld clients. We explore more fine-grained co-ordination of multimodal presentations as answers to natural language questions about a specific domain by graph-based visualisation and navigation in ontological RDF result structures. Semantic Navigation on mobile devices leverages graphical user interface activity for dialogical interaction in mobile environments. Constraint-based programming helps to find optimised multimedia graph visualisations.

Introduction. For every specific type of information there are certain categories of visual representations that are more suitable than others. The use of a graph for the visualisation of information has the advantage that it can capture a detailed knowledge structure. Therefore graphs are suitable for conveying semantic relations between individual information items and for providing an understanding of the overall information structure. Apart from that dialogue systems are very useful for interacting with Web-based information services in mobile environments.

The challenge we address is the intuitive navigation in a structured, semantically organised information space on small interaction devices such as PDAs. Our aim is to implement and evaluate mobile Semantic Web interfaces by application of direct structure mapping from RDF¹ graphs toward their multimedia visualisations. Displaying RDF data in a user-friendly manner is a problem addressed by various types of applications using different representation paradigms¹. At least the following types can be identified: keyword search, e.g. Swoogle², faceted browsing², explicit queries, e.g. Sesame³, and graph visualisations. In our approach we use an automatic layouter for dynamic constrained graph display tailored to suit small displays as valuable extension to other graph visualisation techniques. By additional graph presentation capabilities in a primary linguistic

¹ <http://www.w3.org/RDF/>

² <http://swoogle.umbc.edu/>

³ <http://www.openrdf.org/>



Fig. 1. Graphical user interface and semantic navigation (centre and right)

question answering scenario, the users would become more engaged in the dialogue, navigate through the incrementally presented result space, and would be encouraged to pose follow-up questions in natural language. In our most recent dialogue system project SMARTWEB [3] following experiences in [4,5,6], we try to provide intuitive multimodal access to a rich selection of Web-based information services; especially the handheld scenario is tailored toward multimodal interaction with ontological knowledge bases and Semantic Web services [7] in the football domain. For example, the user can ask questions like *How many goals has Michael Ballack scored this year?* or *How did Germany play against Argentina in the FIFA World Cup?* The summarised answer to the last question, SMARTWEB provides and synthesises, is *5 Spiele* (5 matches). Figure 1 shows the PDA interaction device. When the structure of the dynamic graph changes, a new optimal layout is computed server-side. A further click on the *Ergebnis* (result) node results in displaying the information: **5:3** n. E., 1:1 n. V. (**1:1, 0:0**), *Ereignis* (incidence) reveals red card for player *Cufre*, for example. A shared representation and a common ontological knowledge base ease the data flow among components and avoid costly transformation processes [8]. This applies to the visualisation process, too. We use the ontological RDF metadata to arrange information pieces in automatically layout graphs with respect to their semantic relations extracted from RDF results obtained from our knowledge servers. Humans themselves may encode information based upon its meaning [9]. Users feel familiar with this way of information arrangement at least.

Integration. We integrated the semantic graph visualisation approach into our distributed dialogue system. In SMARTWEB, the graph-based user interface on the client is connected to the automatic graph layouter that resides on the server. All data transfer between server and client is organised by special XML structures transmitted over socket connections in both directions. We extended this XML structure by an new *dynamic graph* environment, for the graph structure data to be exchanged, the graph node layout positions, and the user interactions. The data flow is shown in figure 2. The presentation capabilities after integration

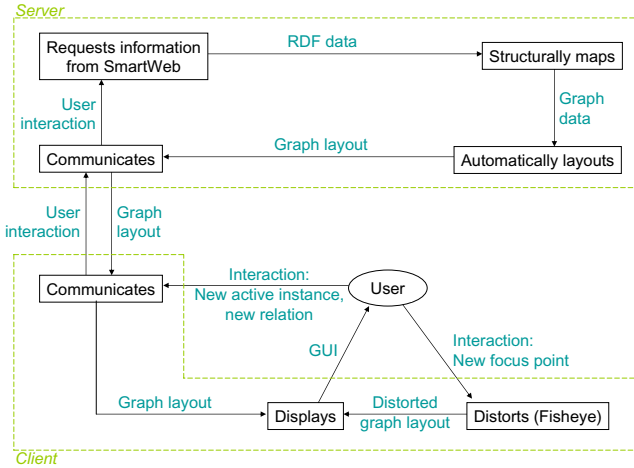


Fig. 2. Semantic graph visualisation data flow. Graph layouts for arbitrary RDF graph data are calculated on the server.

include (1) summarising multimodal result and finding an appropriate mapping toward a lower-level visual object and its attributes which we model in the interaction ontology, (2) finding out visual pattern interrelationships, (3) automating the visualisation of multimodal graph information which complements natural language generation output, and (4) provide consecutive information displays communicated from the server to the client for semantic navigation.

Conclusion and Future Work. Semantic graph visualisation as presented relies on ontological formulation of interaction and presentation constraints, as well as on highly structured RDF data as result structures in the question answering scenario we model. Since the RDF result data are already in a graph-like format, we explored how to map this RDF data into a graph structure and how the resulting graph can be visualised, as an example of how to visualise the Semantic Web. During the development of our system, we used two evaluation phases that involved 20 users in testing design ideas and to get their feedback at an early stage of development. These feedbacks were useful sources of suggestions for the further improvement of our graph presentation system, and show additionally, that graph visualisations and interactions are generally welcomed alternatives for

highly structured result data in question answering scenarios; 85% describe the graph interaction possibilities as easy to understand (after an initial demonstration), 95% easily understand the difference between instance nodes and relation nodes. We conclude by further motivating the use of ontologies and Semantic Web data structures [10] for multimodal interaction design and implementation, and in particular, for visualising graph-like information spaces on mobile PDA devices. We hope that in the future graph visualisation approaches as presented here can help to provide an answer to the question how conceptual data models facilitate the generation of semantic navigation structures on mobile devices.

Acknowledgements. The research presented here is sponsored by the German Ministry of Research and Technology (BMBF) under grant 01IMD01A (SmartWeb). We thank our project partners, our research assistants, and the evaluators. The responsibility for this papers lies with the authors.

References

1. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A browser-independent presentation vocabulary for RDF. In: International Semantic Web Conference, pp. 158–171 (2006)
2. Yee, K.P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, pp. 401–408. ACM Press, New York (2003)
3. Wahlster, W.: SmartWeb: Mobile Applications of the Semantic Web. In: Dadam, P., Reichert, M. (eds.) GI Jahrestagung 2004, pp. 26–27. Springer, Heidelberg (2004)
4. Wahlster, W. (ed.): VERBMOBIL: Foundations of Speech-to-Speech Translation. Springer, Heidelberg (2000)
5. Wahlster, W.: SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In: Krahl, R., Günther, D. (eds.) Proc. of the Human Computer Interaction Status Conference 2003, Berlin, Germany, pp. 47–62. DLR (2003)
6. Wahlster, W.: SmartKom: Foundations of Multimodal Dialogue Systems (Cognitive Technologies). pringer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
7. Sonntag, D., Engel, R., Herzog, G., Pfalzgraf, A., Pflieger, N., Romanelli, M., Reithinger, N.: Smartweb handheld–multimodal interaction with ontological knowledge bases and semantic web services. In: Proceedings of the International Workshop on Artificial Intelligence for Human Computing at IJCAI 2007 (2007)
8. Oberle, D., Ankolekar, A., Hitzler, P., Cimiano, P., Sintek, M., Kiesel, M., Mougouie, B., Vembu, S., Baumann, S., Romanelli, M., Buitelaar, P., Engel, R., Sonntag, D., Reithinger, N., Loos, B., Porzel, R., Zorn, H.P., Micelli, V., Schmidt, C., Weiten, M., Burkhardt, F., Zhou, J.: Dolce ergo sumo: On foundational and domain models in SWIntO (SmartWeb Integrated Ontology). Technical report, AIFB, Karlsruhe (July 2006)
9. Myers, D.G.: Psychology. Worth Publishers (2004)
10. Fensel, D., Hendler, J.A., Lieberman, H., Wahlster, W.: Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. The MIT Press, Cambridge (2005)

A Qualitative Model for Visibility Relations

Francesco Tarquini, Giorgio De Felice, Paolo Fogliaroni, and Eliseo Clementini

University of L'Aquila Department of Electrical and Information Engineering
67040 Poggio di Roio, L'Aquila, Italy
tarquini@ing.univaq.it , giorgiodf@tiscali.it,
p.fogliaroni@hotmail.it, eliseo@ing.univaq.it

Abstract. The visibility concept is related to many application fields such as robot navigation, computer graphics and telecommunication systems. In this paper we propose a new qualitative model for visibility relations based on properties of the projective space. Within the model we present a set of seven ternary relations among convex regions. Our model is capable to determine the visibility relation between a primary object A with the respect to a “region of view” C and an obstacle B. We developed the reasoning system, which allows the prediction of ternary relations between specific regions.

Keywords: Qualitative Spatial Reasoning, Projective Relations, Visibility.

1 Introduction

One of the most common definition of visibility says that “visibility is the maximum distance an object may be seen considering air conditions”. In this paper we restrict the concept to the drawing a line from a point to another one without crossing any obstacle. This definition of visibility is very close to the line-of-sight concept: a calculus of line-of-sight relations, developed in [4], defines the different qualitative relations in which one object in a person’s visual field can be positioned in relation to another one. The relations of the occlusion calculus, presented in [6], qualitatively describe configurations of two convex objects in the projective view from a 3D scene. In our approach, the visibility concept is related to the study of projective characteristics: a qualitative model about projective relations based on collinearity was presented in [1] [2] [3]. The motivation of having a qualitative model arises from the fact that in many applications there is no need to have the whole expressivity of a metric space. For example in robot navigation [7] the robot could take decisions about the route to take on the basis of few information obtained by the knowledge on qualitative projective relations without calculating each time its exact position with the respect to other robots or obstacles. The region occlusion calculus, presented in [8], is a first-order logical theory that describes the spatial relations between bodies as seen from a robot’s viewpoint. The most important novelty of our model, with the respect to the previously mentioned models [4] [6] [8], is that our “point of view” is not a simple point but any convex geometry. Furthermore, since the point of view is not implicit, we need to model the visibility relation as a ternary relation, leading to a more general model.

2 Acceptance Areas for Visibility

We will define visibility relations that check if the primary object A is visible with an obstacle object B from a “point of view” object C .

In order to model the visibility relations, we need to identify their acceptance areas. In detail, we define a *Shadow Zone (SZ)*, a *Twilight Zone (TZ)* (possibly split into *Right Twilight Zone (RTZ)* and *Left Twilight Zone (LTZ)*), and a *Light Zone (LZ)* as shown in Fig. 1.

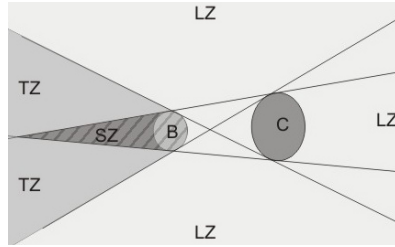


Fig. 1. Acceptance Areas

3 Visibility Relations

In this section, we use the acceptance areas defined in Section 2 (SZ, TZ, LZ) to build a jointly exhaustive and pair-wise disjoint (JEPD) model for visibility relations among three convex regions A, B, C . Let us consider the following matrix of empty/nonempty intersections of a region A with these three areas:

| | | |
|------------------|------------------|------------------|
| $A \cap LZ(B,C)$ | $A \cap TZ(B,C)$ | $A \cap SZ(B,C)$ |
|------------------|------------------|------------------|

In the matrix, a value 0 indicates an empty intersection, while a value 1 indicates a nonempty intersection. This matrix can have 2^3-1 different configurations (all empty values configuration (0 0 0) is not possible because the primary object must intersect at least a region of the space). Each configuration corresponds to a visibility relation among three regions A, B, C .

If we consider the basic configurations of the matrix, with only one non-empty value, we can define the visibility relations of Fig. 2.

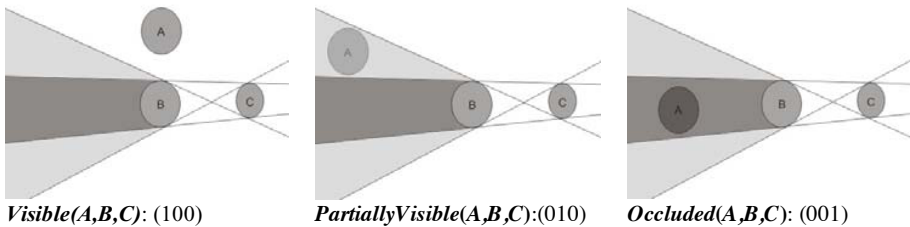


Fig. 2. Basic visibility relations

Considering the other configurations of the matrix we can define the relations of Fig. 3.

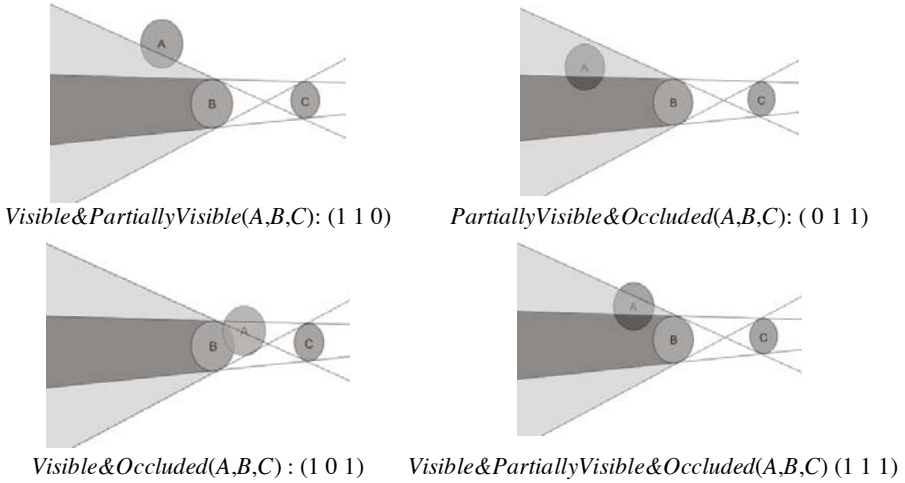


Fig. 3. Compound visibility relations

5 Reasoning

A full discussion of properties of ternary relations is outside the scope of this paper. A relation algebra of ternary relations has been introduced in [5]. It is possible to build a reasoning system, which allows the prediction of ternary relations between specific regions, on the basis of the following rules:

- (1) $r(A,B,C) \rightarrow p(A,C,B)$ (converse)
- (2) $r(A,B,C) \rightarrow q(C,A,B)$ (rotation)
- (3) $r_1(A,B,C) \oplus r_2(B,C,D) \rightarrow r_3(A,C,D)$ (composition)

For the sake of brevity, we will use short names for visibility relations corresponding to the capital letters of full relation names. We denote by U the universal disjunctive visibility relation. Moreover, for some cases the result is

Table 1. Permutation table

| $r(A,B,C)$ | $p(A,C,B)$ | $q(C,A,B)$ |
|------------|---------------|--------------------|
| V | U | U |
| P | V,P,O,PO | V,O,VO,VPO |
| O | V,P,O,PO | V,P,O,PO |
| VP | V,VP | V,P,VP,PO,VO,VPO |
| PO | V,P,O,PO | V,P,O,PO |
| VO | V,VP,VO,VPO | P,O,PO |
| VPO | V,VP,VO,VPO | U |

Table 2. Composition table

| $r_2 \setminus r_1$ | <i>V</i> | <i>P</i> | <i>O</i> | <i>VP</i> | <i>PO</i> | <i>VO</i> | <i>VPO</i> |
|---------------------|--------------------|----------------------|--------------------|-------------------------|-------------------------|-------------------|------------------------|
| <i>V</i> | <i>U</i> | <i>U</i> | <i>U</i> | <i>U</i> | <i>U</i> | <i>IMP</i> | <i>U</i> |
| <i>P</i> | <i>V</i> | <i>V,P,VP</i> | <i>U</i> | <i>V,P,VP</i> | <i>U</i> | <i>U</i> | <i>U</i> |
| <i>O</i> | <i>V</i> | <i>V,P,VP</i> | <i>U</i> | <i>V,P,VP</i> | <i>U</i> | <i>U</i> | <i>U</i> |
| <i>VP</i> | <i>V,VP,VO,VPO</i> | <i>V,P,VP,PO,VPO</i> | <i>U</i> | <i>V,P,VP,PO,VO,VPO</i> | <i>U</i> | <i>IMP</i> | <i>U</i> |
| <i>PO</i> | <i>V</i> | <i>P,VP</i> | <i>O,PO,VPO</i> | <i>V,P,VP</i> | <i>P,O,VP,PO,VPO</i> | <i>V,O,PO,VPO</i> | <i>V,P,O,VP,PO,VPO</i> |
| <i>VO</i> | <i>V,VP,VO,VPO</i> | <i>P,VP,PO,VPO</i> | <i>O,PO,VO,VPO</i> | <i>V,P,VP,PO,VO,VPO</i> | <i>P,O,VP,PO,VO,VPO</i> | <i>IMP</i> | <i>U</i> |
| <i>VPO</i> | <i>V,VP,VO,VPO</i> | <i>P,VP,PO,VPO</i> | <i>O,PO,VO,VPO</i> | <i>V,P,VP,PO,VO,VPO</i> | <i>P,O,VP,PO,VO,VPO</i> | <i>IMP</i> | <i>U</i> |

impossible, denoted by *IMP*. For any relation $r(A,B,C)$ in the model, Table 1 gives the corresponding relations resulting from permutation rules (1) and (2). Table 2 gives the corresponding relations resulting from composition rule (3).

References

- [1] Billen, R., Clementini, E.: Introducing a reasoning system based on ternary projective relations. In: Fisher, P. (ed.) Developments in Spatial Data Handling, 11th International Symposium on Spatial Data Handling, Leicester, UK, pp. 381–394 (2004)
- [2] Billen, R., Clementini, E.: Semantics of collinearity among regions. In: Meersman, R., Tari, Z., Herrero, P. (eds.) On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops. LNCS, vol. 3762, pp. 1066–1076. Springer, Heidelberg (2005)
- [3] Clementini, E., Billen, R.: Modeling and computing ternary projective relations between regions. *IEEE Transactions on Knowledge and Data Engineering* 18, 799–814 (2006)
- [4] Galton, A.: Lines of Sight. In: Keane, M., Cunningham, P., et al. (eds.) AI and Cognitive Science '94 Proceedings of the Seventh Annual Conference, September 8-9, 1994, pp. 103–113. Trinity College Dublin (1994)
- [5] Isli, A., Cohn, A.G.: A new approach to cyclic ordering of 2D orientations using ternary relation algebras. *Artificial Intelligence* 122, 137–187 (2000)
- [6] Kohler, C.: The Occlusion Calculus. In: Workshop on Cognitive Vision Zurich Switzerland (2002)
- [7] Oommen, B., Iyengar, S., Rao, N., Kashyap, R.: Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case. *IEEE Journal of Robotics and Automation* 3, 672–681 (1987)
- [8] Witkowski, M., Shanahan, M., Santos, P., Randell, D.: Cognitive Robotics: On the Semantic Knife-edge. In: 3rd British Conf. on Autonomous Mobile Robotics and Autonomous Systems (2001)

Author Index

- Abdenebaoui, Larbi 427
Albayrak, Sahin 1
Artmann, Stefan 431
Autexier, Serge 435
- Baader, Franz 52
Baiget, Pau 279
Basselin, Nathalie 477
Bastos, Rogério C. 452
Beetz, Michael 19, 129, 473
Benazera, Emmanuel 337
Benzmüller, Christoph 435
Berger, Ralf 440
Berns, Karsten 464, 494
Bibel, Wolfgang 2
Bidot, Julien 367
Birk, Andreas 293
Biundo, Susanne 367
Böhnstedt, Lutz 352
Braun, Tim 464
Brockmann, Werner 498
Buss, Martin 19
- Christaller, Thomas 43
Clementini, Eliseo 510
- Dietrich, Dominik 435
Doherty, Patrick 460
Dorffner, Georg 235
Drescher, Conrad 68
- Edelkamp, Stefan 382, 444
Edgington, Mark 205
Elkmann, Norbert 502
Elovici, Yuval 44, 489
Engel, Ralf 448
Englert, Roman 489
- Fangmeier, Thomas 175
Feher, Clint 489
Felice, Giorgio De 510
Fernández Tena, Carles 279
Ferrein, Alexander 352
Ferro, Humberto F. 452
Figuroa, Alejandro 144
- Finthammer, Marc 114
Fogliaroni, Paolo 510
Fox, Dieter 51
Frintrop, Simone 456
Fritz, Gerald 235
Fritzsche, Markus 502
Fürstenau, Norbert 251
- Galán-Marín, Gloria 397
Gea, Jose de 205
Girstl, Angelika 502
Glezer, Chanan 44
González, Jordi 279
Gottfried, B. 308
Gretton, Charles 412
- Hafner, Roland 220
Hahn, Hernsoo 323
Hahn, Kwangsoo 323
Hammer, Barbara 190
Han, Youngjoon 323
Hasenfass, Alexander 190
Heim, Philipp 506
Heintz, Fredrik 460
Helmert, Malte 412
Herzog, O. 308
Hildebrand, Lars 382
Hirschberger, Andreas 468
Hirth, Jochen 464
Hitzler, Pascal 84
Hofmann, Martin 468
- Irran, Jörg 235
- Jain, Dominik 129
- Kahl, Kenneth 382
Kassahun, Yohannes 205, 427
Kern-Isberner, Gabriele 114
Kintzler, Florian 235
Kirchlechner, Bernhard 129
Kirchner, Elsa A. 427
Kirchner, Frank 205, 427
Kirsch, Alexandra 473
Kissmann, Peter 444

- Kitzelmann, Emanuel 468
 Kröner, Alexander 477

 Lakemeyer, Gerhard 352
 Lämmel, Gregor 440
 Lange, Sascha 220
 Latoschik, Marc Erich 481
 Lauer, Martin 220
 López-Rodríguez, Domingo 397
 Luksch, Tobias 494

 Markov, Stefan 293
 Matyasik, Piotr 485
 Matzner, Tobias 84
 Meilicke, Christian 99
 Mérida-Casermeiro, Enrique 397
 Merke, Artur 220
 Metzen, Jan Hendrik 205
 Mori, Junichiro 477
 Moskovitch, Robert 44, 489
 Muhl, Claudia 264
 Müller, Heiko 220

 Nagai, Yukie 264
 Nalepa, Grzegorz J. 485
 Neumann, Günter 144
 Nissim, Nir 489

 Oliveira, Cláudio M. de 452
 Ortiz-de-Lazcano-Lobato, Juan M. 397

 Paletta, Lucas 235
 Peñaloza, Rafael 52
 Proetzsch, Martin 494

 Ragni, Marco 175
 Richter, Silvia 412
 Riedmiller, Martin 220
 Ritterskamp, Manuela 114
 Roca, Xavier 279
 Rosemann, Nils 498
 Rudol, Piotr 460

 Sagerer, Gerhard 264
 Schattenberg, Bernd 367
 Schenk, Simon 160
 Schiller, Marvin 435
 Schleipen, Stefan 175
 Schmid, Ute 468
 Schneider, Michael 477
 Schuldt, A. 308
 Schulenburg, Erik 502
 Shabtai, Asaf 44
 Sonntag, Daniel 448, 506
 Stiene, Stefan 502
 Stopel, Dima 489
 Stuckenschmidt, Heiner 99
 Suntisrivaraporn, Boontawee 52

 Tahan, Gil 44
 Tarquini, Francesco 510
 Teutsch, Christian 502
 Thielscher, Michael 68

 Wazlawick, Raul S. 452
 Wollherr, Dirk 19

 Zięcik, Piotr 485