

Florin Manea
Russell G. Miller
Dirk Nowotka (Eds.)

LNCS 10936

Sailing Routes in the World of Computation

14th Conference on Computability in Europe, CiE 2018
Kiel, Germany, July 30 – August 3, 2018
Proceedings



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology Madras, Chennai, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7407>

Florin Manea · Russell G. Miller
Dirk Nowotka (Eds.)

Sailing Routes in the World of Computation

14th Conference on Computability in Europe, CiE 2018
Kiel, Germany, July 30 – August 3, 2018
Proceedings

Editors

Florin Manea
Kiel University
Kiel
Germany

Dirk Nowotka
Kiel University
Kiel
Germany

Russell G. Miller
Queens College, CUNY
Queens, NY
USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-94417-3 ISBN 978-3-319-94418-0 (eBook)
<https://doi.org/10.1007/978-3-319-94418-0>

Library of Congress Control Number: 2018947191

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Computability in Europe 2018: Sailing the Routes of Computation Kiel, Germany, July 30 to August 3, 2018



The conference Computability in Europe (CiE) is organized yearly under the auspices of the Association CiE, a European association of mathematicians, logicians, computer scientists, philosophers, physicists, biologists, historians, and others interested in new developments in computability and their underlying significance for the real world. CiE promotes the development of computability-related science, ranging over mathematics, computer science, and applications in various natural and engineering sciences, such as physics and biology, as well as related fields, such as philosophy and history of computing. CiE 2018 had as its motto “Sailing Routes in the World of Computation,” a nod to the host city of Kiel and its rich nautical traditions, but also a reminder of the possibility of widely varied solutions to problems in computability and the importance of considering the different possible routes that can be followed when studying and answering open questions.

CiE 2018 was the 14th conference in the series, and the second one to take place in Germany. The conference was organized by the Department of Computer Science of Kiel University. The 13 previous CiE conferences were held in Amsterdam (The Netherlands) in 2005, Swansea (Wales) in 2006, Siena (Italy) in 2007, Athens (Greece) in 2008, Heidelberg (Germany) in 2009, Ponta Delgada (Portugal) in 2010, Sofia (Bulgaria) in 2011, Cambridge (UK) in 2012, Milan (Italy) in 2013, Budapest (Hungary) in 2014, Bucharest (Romania) in 2015, Paris (France) in 2016, and Turku (Finland) in 2017. CiE 2019 will be held in Durham (UK). Currently, the annual CiE conference is the largest international meeting focused on computability-theoretic issues. The proceedings containing the best submitted papers, as well as extended abstracts of invited, tutorial, and special session speakers, for all these meetings are published in the Springer series *Lecture Notes in Computer Science*.

The CiE conference series is coordinated by the CiE Conference Series Steering Committee consisting of Alessandra Carbone (Paris), Liesbeth De Mol (Lille), Mathieu Hoyrup (Nancy), Natasha Jonoska (Tampa FL), Benedikt Löwe (Amsterdam), Florin Manea (Kiel, Chair), Klaus Meer (Cottbus), Mariya Soskova (Sofia), and ex-officio members Paola Bonizzoni (Milan, President of the Association CiE), and Dag Normann (Oslo).

The Program Committee of CiE 2018 was chaired by Russell Miller (City University of New York, USA) and Dirk Nowotka (Christian-Albrechts-Universität, Kiel, Germany). The committee, consisting of 30 members, selected the invited and tutorial speakers and the special session organizers, and coordinated the reviewing process of all submitted contributions.

Structure and Program of the Conference

The Program Committee invited six speakers to give plenary lectures at CiE 2018: Kousha Etessami (Edinburgh, UK), Johanna Franklin (Hempstead, USA), Mai Gehrke (Paris, France), Alberto Marcone (Udine, Italy), Alexandra Silva (London, UK), and Jeffrey O. Shallit (Waterloo, Canada). The conference also had two plenary tutorials, presented by Pinar Heggernes (Bergen, Norway) and Bakhadyr Khossainov (Auckland, New Zealand).

In addition, the conference had six special sessions: Approximation and Optimization; Bioinformatics and Bio-inspired Computing; Computing with Imperfect Information; Continuous Computation; History and Philosophy of Computing (celebrating the 80th birthday of Martin Davis); and SAT-solving. Speakers in these special sessions were selected by the respective special session organizers and were invited to contribute a paper to this volume.

Approximation and Optimization

Organizers: Leah Epstein and Klaus Jansen

Speakers:

Sebastian Berndt (Kiel University)

Thomas Erlebach (University of Leicester)

Kim-Manuel Klein (École Polytechnique Fédérale de Lausanne)

Asaf Levin (Technion, Haifa)

Bioinformatics and Bio-inspired Computing

Organizers: Andre Franke and Victor Mitran

Speakers:

Oliver Kohlbacher (Universität Tübingen)

Alfonso Paton (Universidad Politécnica de Madrid)

Bertil Schmidt (Johannes Gutenberg University Mainz)

Takashi Yokomori (Waseda University)

Computing with Imperfect Information

Organizers: Tim McNicholl and Mariya Soskova

Speakers:

Eric Astor (University of Connecticut)

Ethan McCarthy (University of Wisconsin-Madison)

Arno Pauly (Swansea University)

Paul Schupp (University of Illinois at Urbana-Champaign)

Continuous Computation

Organizers: Ulrich Berger and Dieter Spreen

Speakers:

Matthew de Brecht (Kyoto University)

Daniel Graça (University of Algarve)

Mathieu Hoyrup (LORIA, Nancy)

Dag Normann (University of Oslo)

History and Philosophy of Computing

Organizers: Liesbeth De Mol and Giuseppe Primiero

Speakers:

Christoph Benz Müller (Luxembourg/FU Berlin)

Martin Davis (New York University)

Paula Quinon (Lund University)

Wilfried Sieg (Carnegie Mellon University)

SAT-solving

Organizers: Olaf Beyersdorff and Vijay Ganesh

Speakers:

Oliver Kullmann (Swansea University)

Massimo Lauria (Sapienza University Rome)

Florian Lonsing (TU Vienna)

Joao Marques Silva (University of Lisbon)

The members of the Program Committee of CiE 2018 selected for publication in this volume and for presentation at the conference 26 of the 55 non-invited submitted papers. Each paper received at least three reviews by the Program Committee and their subreviewers. In addition to the accepted contributed papers, this volume contains 15 invited papers. The production of the volume would have been impossible without the diligent work of our expert referees, both Program Committee members and subreviewers. We would like to thank all of them for their excellent work.

Springer generously funded a Best Student Paper Award this year, awarded to a paper authored solely by students. The winner of this award was the paper “Elementary Bi-embeddability Spectra of Structures” by Dino Rossegger.

All authors who contributed to this conference were encouraged to submit significantly extended versions of their papers, with additional unpublished research content, to *Computability: The Journal of the Association CiE*.

The Steering Committee of the conference series CiE is concerned about the representation of female researchers in the field of computability. In order to increase female participation, the series started the Women in Computability (WiC) program in 2007. In 2016, after the new constitution of the Association CiE allowed for the possibility of creating special interest groups, a Special Interest Group named Women in Computability was established. Since 2016, the WiC program has been sponsored by ACM’s Women in Computing. This program includes a workshop, the annual WiC diner, the mentorship program and a grant program for young female researchers. The Women in Computability workshop continued in 2018, coordinated by Liesbeth De Mol.

The organizers of CiE 2018 would like to acknowledge and thank the following entities for their financial support (in alphabetical order): the Association for Symbolic Logic (ASL), the European Association for Theoretical Computer Science (EATCS), and Springer. We would also like to acknowledge the support of our non-financial sponsor, the Association Computability in Europe.

We gratefully thank all the members of the Organizing Committee of CiE 2018 for their work toward making the conference a successful event, and Gheorghe Iosif for designing the poster and banner of CiE 2018.

We thank Andrej Voronkov for his EasyChair system, which facilitated the work of the Program Committee and the editors considerably.

April 2018

Florin Manea
Russell Miller
Dirk Nowotka

Organization

Program Committee

Eric Allender	Rutgers University, USA
Arnold Beckmann	Swansea University, UK
Marco Benini	Università degli Studi dell'Insubria, Italy
Olaf Beyersdorff	University of Leeds, UK
Patricia Bouyer	LSV, CNRS and ENS Cachan, Université Paris Saclay, France
Alessandra Carbone	Sorbonne Université, France
Barbara Csima	University of Waterloo, Canada
Anuj Dawar	University of Cambridge, UK
Ekaterina Fokina	Vienna University of Technology, Austria
Peter Høyer	University of Calgary, Canada
Georgiana Ifrim	University College Dublin, Ireland
Lila Kari	University of Waterloo, Canada
Karen Lange	Wellesley College, USA
Benedikt Löwe	University of Amsterdam, The Netherlands
Florin Manea	Kiel University, Germany
Barnaby Martin	Durham University, UK
Klaus Meer	BTU Cottbus-Senftenberg, Germany
Russell Miller	Queens College and The Graduate Center, CUNY, USA
Angelo Montanari	University of Udine, Italy
Andrey Morozov	Sobolev Institute of Mathematics, Russia
Anca Muscholl	LaBRI, Université de Bordeaux, France
Dirk Nowotka	Kiel University, Germany
Arno Pauly	Swansea University, UK
Giuseppe Primiero	Middlesex University, UK
Henning Schnoor	Kiel University, Germany
Monika Seisenberger	Swansea University, UK
Shinnosuke Seki	University of Electro-Communications, Japan
Mariya Soskova	Sofia University, Bulgaria
Peter Van Emde Boas	ILLC-FNWI-Universiteit van Amsterdam (Emeritus), The Netherlands
Heribert Vollmer	Leibniz Universität Hannover, Germany

Organizing Committee

Joel D. Day
Karoliina Lehtinen
Florin Manea
Dirk Nowotka (Chair)
Anneke Twardzik

Additional Reviewers

Abel, Andreas
 Andrews, Uri
 Balbiani, Philippe
 Barmपालias, George
 Baskent, Can
 Beggs, Edwin
 Berger, Ulrich
 Berndt, Sebastian
 Brattka, Vasco
 Buss, Sam
 Calvert, Wesley
 Cenzer, Douglas
 Chandoo, Maurice
 Cholak, Peter
 Day, Joel
 De Mol, Liesbeth
 Dorais, François
 Downey, Rod
 Fan, Xiuyi
 Fazekas, Szilard Zsolt
 Fichte, Johannes
 Franklin, Johanna
 Frittaion, Emanuele
 Gacs, Peter
 Galeotti, Lorenzo
 Gambino, Nicola
 Gherardi, Guido
 Giannopoulos, Panos
 Giannopoulou, Archontia
 Graça, Daniel
 Haak, Anselm
 Hackbusch, Wolfgang
 Harrison-Trainor, Matthew
 Hirschfeldt, Denis
 Hoyrup, Mathieu
 Kach, Asher
 Kawamura, Akitoshi
 Kjos-Hanssen, Bjoern
 Kosub, Sven
 Kufleitner, Manfred
 Lempp, Steffen
 Longley, John
 Lück, Martin
 Maack, Marten
 Martin, Barnaby
 Martini, Simone
 McInerney, Michael
 Meier, Arne
 Melnikov, Alexander
 Mercas, Robert
 Metcalfe, George
 Molinari, Alberto
 Mottet, Antoine
 Nagel, Lars
 Naibo, Alberto
 Neumann, Eike
 Ng, Keng Meng Selwyn
 Ng, Timothy
 Nordvall Forsberg, Fredrik
 Ochem, Pascal
 Ochremiak, Joanna
 Okubo, Fumiya
 Panangaden, Prakash
 Patey, Ludovic
 Pikhurko, Oleg
 Policriti, Alberto
 Reinhardt, Klaus
 Rin, Benjamin
 Rossegger, Dino
 Sala, Pietro
 San Mauro, Luca
 Schlicht, Philipp
 Setzer, Anton
 Sigley, Sarah
 Soskova, Alexandra
 Stephan, Frank
 Stephenson, Jonny
 Todinca, Ioan
 Tucker, John
 van den Berg, Benno
 Variyam, Vinodchandran
 Vicedomini, Riccardo
 Westrick, Linda Brown
 Winslow, Andrew
 Zetsche, Georg

Algorithmic Randomness in Analysis (Invited Talk)

Johanna N. Y. Franklin

Hofstra University, Room 306, Roosevelt Hall, Hempstead,
NY 11549-0114, USA

johanna.n.franklin@hofstra.edu

http://people.hofstra.edu/Johanna_N_Franklin/

While there is a plethora of ways to formalize randomness algorithmically, they all result in a measure one set of random points. Furthermore, theorems in analysis tend to hold on a measure one set of points. In the past decade, there has been a concerted effort to formalize these theorems using computability theory and then to identify the set of points on which such a theorem holds as a set of random points (often, the appropriate randomness notion is Martin-Löf randomness or Schnorr randomness, but not always).

I will survey some recent work that demonstrates the connections between algorithmic randomness and theorems on ergodic theory, differentiability, and convergence of Fourier series and discuss some of the proof techniques used in each.

Contents

A Journey to Computationally Enumerable Structures (Tutorial Lectures)	1
<i>Bakh Khoussainov</i>	
Polynomial-Time Presentations of Algebraic Number Fields	20
<i>Pavel Alaev and Victor Selivanov</i>	
Multiple Permitting and Array Noncomputability	30
<i>Klaus Ambos-Spies</i>	
Degrees of Categoricity for Prime and Homogeneous Models	40
<i>Nikolay Bazhenov and Margarita Marchuk</i>	
Universality in Freezing Cellular Automata	50
<i>Florent Becker, Diego Maldonado, Nicolas Ollinger, and Guillaume Theyssier</i>	
A Deontic Logic Reasoning Infrastructure	60
<i>Christoph Benzmüller, Xavier Parent, and Leendert van der Torre</i>	
Optimized Program Extraction for Induction and Coinduction	70
<i>Ulrich Berger and Olga Petrovska</i>	
Computing Tree Width: From Theory to Practice and Back	81
<i>Sebastian Berndt</i>	
Using Structural Properties for Integer Programs	89
<i>Sebastian Berndt and Kim-Manuel Klein</i>	
From Eventually Different Functions to Pandemic Numberings	97
<i>Achilles A. Beros, Mushfeq Khan, Bjørn Kjos-Hanssen, and André Nies</i>	
Divide and Conquer Computation of the Multi-string BWT and LCP Array	107
<i>Paola Bonizzoni, Gianluca Della Vedova, Serena Nicosia, Yuri Pirola, Marco Previtalli, and Raffaella Rizzi</i>	
Some Observations on Infinitary Complexity	118
<i>Merlin Carl</i>	
Taming Koepke’s Zoo	126
<i>Merlin Carl, Sabrina Ouazzani, and Philip Welch</i>	

Online Computability and Differentiation in the Cantor Space	136
<i>Douglas Cenzer and Diego A. Rojas</i>	
Turing’s Vision and Deep Learning.	146
<i>Martin Davis</i>	
Computing and Scheduling with Explorable Uncertainty	156
<i>Thomas Erlebach</i>	
Diminishable Parameterized Problems and Strict Polynomial Kernelization. . .	161
<i>Henning Fernau, Till Fluschnik, Danny Hermelin, Andreas Krebs, Hendrik Molter, and Rolf Niedermeier</i>	
New Nonterminal Complexity Results for Semi-conditional Grammars	172
<i>Henning Fernau, Lakshmanan Kuppusamy, and Rufus O. Oladele</i>	
Kernelization Lower Bounds for Finding Constant-Size Subgraphs	183
<i>Till Fluschnik, George B. Mertzios, and André Nichterlein</i>	
On General Sum Approximations of Irrational Numbers	194
<i>Ivan Georgiev, Lars Kristiansen, and Frank Stephan</i>	
Computability of Ordinary Differential Equations	204
<i>Daniel S. Graça and Ning Zhong</i>	
Topological Analysis of Representations	214
<i>Mathieu Hoyrup</i>	
A Unified Framework for Designing EPTAS’s for Load Balancing on Parallel Machines	224
<i>Ishai Kones and Asaf Levin</i>	
Weak Reduction Principle and Computable Metric Spaces	234
<i>Margarita Korovina and Oleg Kudinov</i>	
Decidable and Undecidable Fragments of First-Order Concatenation Theory	244
<i>Lars Kristiansen and Juvenal Murwanashyaka</i>	
Algorithm Analysis Through Proof Complexity	254
<i>Massimo Lauria</i>	
Computing with SAT Oracles: Past, Present and Future	264
<i>Joao Marques-Silva</i>	
The Isometry Degree of a Computable Copy of ℓ^p	277
<i>Timothy H. McNicholl and Don Stull</i>	

Algorithmic Statistics and Prediction for Polynomial
 Time-Bounded Algorithms 287
Alexey Milovanov

A C.E. Weak Truth Table Degree Which Is Array Noncomputable
 and R-maximal 297
Martin Monath

The Complexity of Tukey Types and Cofinal Types 307
Marie Nicholson

Functionals of Type 3 as Realisers of Classical Theorems in Analysis 318
Dag Normann

Enumeration Degrees and Topology 328
Arno Pauly

A Taxonomy of Deviant Encodings. 338
Paula Quinon

Elementary Bi-embeddability Spectra of Structures 349
Dino Rossegger

A Generic m-Reducibility 359
Alexander Rybalov

Some Nonstandard Equivalences in Reverse Mathematics 365
Sam Sanders

Bit Complexity of Computing Solutions for Symmetric Hyperbolic
 Systems of PDEs (Extended Abstract) 376
Svetlana V. Selivanova and Victor L. Selivanov

What Is the *Concept* of Computation? 386
Wilfried Sieg

Witness Hiding Without Extractors or Simulators 397
André Souto, Luís Antunes, Paulo Mateus, and Andreia Teixeira

Algorithms and Geometric Constructions 410
Vladimir Uspenskiy and Alexander Shen

Computing with Multisets: A Survey on Reaction Automata Theory 421
Takashi Yokomori and Fumiya Okubo

Author Index 433



A Journey to Computably Enumerable Structures (Tutorial Lectures)

Bakh Khoussainov^(✉)

Department of Computer Science, University of Auckland, Auckland, New Zealand
bmk@cs.auckland.ac.nz

Abstract. The tutorial focuses on computably enumerable (c.e.) structures. These structures form a class that properly extends the class of all computable structures. A computably enumerable (c.e.) structure is one that has computably enumerable equality relation E such that the atomic operations and relations of the structure are induced by c.e. operations and relations that respect E . Finitely presented universal algebras (e.g. groups, rings) are natural examples of c.e. structures. The first lecture gives an introduction to the theory, provides many examples, and proves several simple yet important results about c.e. structures. The second lecture addresses a particular problem about finitely presented expansions of universal algebras with an emphasis to semigroups and groups. The lecture is based on the interplay between important constructions, concepts, and results in computability (Post's construction of simple sets), universal algebra (residual finiteness), and algebra (Golod-Shafarevich theorem). The third lecture is devoted to studying dependency of various properties of c.e. structures on their domains.

1 Introduction

The goal of the tutorial is to present a beautiful interplay between computability and the theory of algebraic structures. Here by an algebraic structure we mean the first order structures given by their domains, atomic operations and predicates on the domains. We always assume that the number of atomic operations and predicates in algebraic structures is finite and the structures are countable. For computability, we accept the Turing model of computation. How does one introduce computability into the study of algebraic structures? Traditionally, this depends on what one considers to be the primary object of interest:

- Assume the primary object is a particular structure (e.g. a graph). A natural way to introduce computability is to postulate that the domain, all the atomic operations and predicates of the structure are computable. Such structures are called *computable*. This is the line taken by Mal'cev [29], Rabin [32] that lead to the development of the theory of computable structures.
- Assume that the primary object is a particular first order theory T (closed under deduction). In this case, a natural way to introduce computability is to postulate that T is a decidable set. With this assumption, the Henkin's

construction of a model of T can now be carried out effectively. In particular, one can build a model \mathcal{M} of T such that the elementary diagram of \mathcal{M} is a decidable set. Such models of T are called *decidable* models. This set-up leads to computable model theory initiated by Yu. Ershov and A. Nerode, and developed by S. Goncharov, T. Millar, N. Khisamiev and L. Harrington.

In this tutorial, we propose an alternative way to introduce computability into the study of algebraic structures. As opposed to the two traditional approaches described above, our primary objects will be quotient sets of the form ω/E , where E is an equivalence relation on ω .

What do the quotient sets ω/E have to do with introducing computability theoretic considerations into the study of algebraic structures?

One answer comes from the homomorphism theorem. The theorem states that any algebra \mathcal{A} (that is, a structure of purely functional language) is a homomorphic image of the term algebra \mathcal{F} . Let us code the elements (terms) of the term algebra \mathcal{F} with natural numbers. So, we view each n natural number as a term. Therefore, the term algebra \mathcal{F} is an algebra such that the domain of \mathcal{F} is ω and the atomic operations of \mathcal{F} are computable operations. So, the algebra \mathcal{F} is computable. The homomorphism theorem implies the following:

Representation Theorem: *For any algebra \mathcal{A} there is a congruence relation E on the term algebra \mathcal{F} such that \mathcal{A} is isomorphic to the quotient algebra \mathcal{F}/E .*

The representation theorem tells us the following. Each $n \in \omega$ represents the element $[n]$ of the algebra \mathcal{A} , where $[n]$ is the E -equivalence class of n . Under this representation, the atomic operations of \mathcal{A} are induced by computable functions. Namely, for any atomic k -ary operation f , given representatives n_1, \dots, n_k of elements a_1, \dots, a_k , respectively, we can compute a representative of the element $f(a_1, \dots, a_k)$. The representative of $f(a_1, \dots, a_k)$ is just the term $f(n_1, \dots, n_k)$. Now we specify algebras whose domains are the quotient sets ω/E :

Definition 1. *An E -algebra is the quotient algebra of the form $(\omega/E; f_1, \dots, f_n)$ such that all f_1, \dots, f_n are computable operations that respect E .*

By the representation theorem, for each algebra \mathcal{A} there exists an equivalence relation E such that \mathcal{A} is isomorphic to an E -algebra.

Definition 2. *Let \mathcal{C} be a class of algebras and let E be an equivalence relation on the set ω of natural numbers. Let us set:*

$$\mathcal{K}(\mathcal{C}, E) = \{\mathcal{A} \in \mathcal{C} \mid \text{algebra } \mathcal{A} \text{ is isomorphic to an } E\text{-algebra}\}.$$

The definitions above place the domains ω/E as primary objects. Through these two definitions we introduce computability into the study of algebraic structures.

If \mathcal{C} is the class of *all* algebras, then instead of $\mathcal{K}(\mathcal{C}, E)$ we write $\mathcal{K}(E)$. Thus, from the representation theorem we have the following important equality:

$$\mathcal{C} = \bigcup_E \mathcal{K}(C, E).$$

The equality tells us that the class \mathcal{C} has been *sliced* into classes $\mathcal{K}(\mathcal{C}, E)$. Understanding the class \mathcal{C} is thus reduced to understanding the classes $\mathcal{K}(\mathcal{C}, E)$ for various E . In particular, many algebraic problems about the class \mathcal{C} , such as the classification problem of algebras up to isomorphism, can now be refined and studied in the *slices* $\mathcal{K}(\mathcal{C}, E)$. We note that the slices are not necessarily disjoint.

Definition 3. *If an algebra \mathcal{A} belongs to $\mathcal{K}(\mathcal{C}, E)$ then we say that E realises the algebra \mathcal{A} . Otherwise, we say that E omits \mathcal{A} .*

We now have a research agenda with many questions. For instance, consider the class $\mathcal{K}(E)$, all algebras isomorphic to E -algebras. Does $\mathcal{K}(E)$ contain a finitely generated algebra? Are there rings or Boolean algebras or trees in $\mathcal{K}(E)$? Can we say anything reasonable about structures in the class $\mathcal{K}(E)$? Does $\mathcal{K}(E)$ contain the term algebra \mathcal{F} ? What algebraic properties of algebras from $\mathcal{K}(E)$ are implied by computability-theoretic properties of E ? Similarly, what algebraic properties of structures from $\mathcal{K}(E)$ imply computability-theoretic properties of E ? Can we describe isomorphism invariants of algebras from $\mathcal{K}(E)$?

One goal of this tutorial is to set up a framework that will address all the questions above. One way to start the investigation of the classes $\mathcal{K}(E)$ is to put computability-theoretic assumptions on E . So, let \mathcal{A} be an E -algebra.

Definition 4. *We call \mathcal{A} a Σ_n^0 -algebra if E is a Σ_n^0 -relation. Similarly, we call \mathcal{A} a Π_n^0 -algebra if E is a Π_n^0 -set. Σ_1^0 -algebras are called *computably enumerable (c.e.) algebras*, and Π_1^0 -algebras are called *co-c.e. algebras*.*

We identify algebras up to the isomorphisms. So, we say that an algebra is *c.e. presentable (co-c.e. presentable)* if it is isomorphic to a c.e. (co-c.e.) algebra. Thus, c.e. (co-c.e.) presentability is the isomorphism type property.

The first lecture of this tutorial is an introduction with many examples and important results. The second lecture solves a particular problem about finitely presented expansions of semigroup, groups, and algebras. The lecture is based on the interplay between important constructions, concepts, and results in computability (Post's construction of simple sets), universal algebra (residual finiteness), and algebra (Golod-Shafarevich theorem). The third lecture studies dependency of various properties of c.e. structures on their domains and compares classes $\mathcal{K}(\mathcal{C}, E)$ for various E . Each tutorial ends with a compact reference list for the topics covered in the tutorial. There is a good amount of research on c.e. equivalence relations (e.g. [1, 2, 5, 8, 11, 28, 30]) but we present an alternative and more algebraic view towards understanding the equivalence relations.

2 Lecture 1: Implications of Non-computability

In this lecture, we *postulate* that E is a c.e. but *not* computable equivalence relation. Just with this assumption, we deduce several results of algebraic nature. In particular, we describe types of algebras that are omitted by the class $\mathcal{K}(E)$.

Theorem 5 (Mal'cev [29]). *The class $\mathcal{K}(E)$ omits all finitely generated algebras whose nontrivial quotients are all finite.*

Proof. Let \mathcal{A} be a finitely generated E -algebra that is a counter-example. For elements $[n], [m]$, let $E(n, m)$ be the congruence relation of \mathcal{A} generated by the pair $([n], [m])$. The congruence $E(n, m)$ is a c.e. relation. If $[n] \neq [m]$ in \mathcal{A} , then the algebra $\mathcal{A}/E(n, m)$ is finite. This implies that the complement of E can be computably enumerated. Thus, E is computable. Contradiction. \square

Corollary 6. *The class $\mathcal{K}(E)$ omits the following: (a) the successor algebra $(\omega; S)$, (b) the arithmetic $(\omega; +, \times)$, and (c) the algebra $(Z; S, S^{-1})$.* \square

Theorem 7 (Ershov [9]). *The class $\mathcal{K}(E)$ omits all algebras that possess finitely many congruences only.*

Proof. Let \mathcal{A} be an E -algebra that is a counter-example to the statement. For simplicity, assume that \mathcal{A} has only 3 non-trivial congruences E_1, E_2 , and E_3 . There must exist pairs $([x_1], [y_1]), ([x_2], [y_2]), ([x_3], [y_3])$ such that the following properties hold: (1) in algebra \mathcal{A} we have $[x_1] \neq [y_1], [x_2] \neq [y_2]$, and $[x_3] \neq [y_3]$, and (2) For every $(n, m) \notin E$ the congruence relation $E(n, m)$ contains one of the pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Hence, the complement of E is computably enumerable. Therefore, the equivalence relation E is computable. \square

Recall that an algebra is simple if it has no non-trivial congruence relations.

Corollary 8. *The class $\mathcal{K}(E)$ omits fields as well as simple groups.* \square

Next we use two fundamental concepts from algebra. The first is the notion of residually finite algebra. Malcev and MacKenzie started studying this notion already in the 1940s. The second is the concept of finitely presented algebra. Dehn initiated the study of finitely presented groups in the 1910s.

Definition 9. *An algebra \mathcal{A} is residually finite if for all $x, y \in \mathcal{A}, x \neq y$, there is a homomorphism h of \mathcal{A} onto a finite algebra \mathcal{B} such that $h(x) \neq h(y)$.*

For instance, finitely generated abelian groups, the term algebras, the successor algebra $(Z; S, 0)$ are residually finite, and the algebra $(\omega; S, S^{-1})$ is not.

Finite presentability of groups is generalised to the setting of universal algebra as follows. A *finite presentation* S is a finite set of equations $t = p$, where t and p are terms that might contain variables. Examples are (1) semi-group axioms, (2) group axioms, and (3) ring axioms. The class of algebras satisfying S is a variety; the class is closed under sub-algebras, products, and homomorphisms.

Let S be a finite presentation. Let \mathcal{F} be the term algebra generated by constants present in the language of S . The elements of \mathcal{F} are usually called ground terms. Let $E(S)$ be the congruence relation on \mathcal{F} generated by S :

$$E(S) = \{(p, q) \mid p, q \text{ are ground terms and } S \vdash p = q\}.$$

Definition 10. *The algebra $\mathcal{F}_S = \mathcal{F}/E(S)$ is called finitely presented by S .*

The finitely presented algebra possesses several nice properties: (a) \mathcal{F}_S satisfies S , (b) \mathcal{F}_S is an $E(S)$ -algebra, (c) \mathcal{F}_S is a c.e. algebra, and (d) \mathcal{F}_S is universal and unique. In particular, item (d) says that any algebra generated by the constants and that satisfies S is a homomorphic image of \mathcal{F}_S .

Theorem 11 (Mal'cev and MacKenzie [29]). *The class $\mathcal{K}(E)$ omits all finitely presented and residually finite algebras.*

Proof. Let \mathcal{A} be an E -algebra that is a counter-example. Let S be a finite presentation of \mathcal{A} . Effectively list $\mathcal{B}_0, \mathcal{B}_2, \dots$ all finite algebras satisfying S . These are homomorphic images of \mathcal{A} . For all $[x], [y] \in \mathcal{A}$, $[x] \neq [y]$, there is an algebra \mathcal{B}_i in which images of $[x]$ and $[y]$ are distinct. Hence, the complement of E is computably enumerable. Hence, the equivalence relation E is computable. \square

Corollary 12. *The class $\mathcal{K}(E)$ omits the following algebras: the term algebra \mathcal{F} , the free group with n generators, the free abelian group Z^n , the free non-associative algebra over a finite field with n non-commuting variables.*

Proof. The algebras listed are all finitely presented and residually finite. \square

Next we connect locally finite algebras with immunity property. For the equivalence relation E , let $tr(E)$ be the set consisting of all minimal representatives of the E -equivalence classes. So, $tr(E) = \{x \mid \forall y (y < x \rightarrow (x, y) \notin E)\}$.

Definition 13. *An algebra \mathcal{A} is locally finite if every finitely generated sub-algebra of \mathcal{A} is finite.*

For instance, the direct sum of infinitely many finite groups is a locally finite algebra. Here is the definition of immunity for subsets X of ω . Post gave this definition in 1944 and proved that immune sets exist:

Definition 14. *An infinite set $X \subseteq \omega$ is immune if X contains no computably enumerable infinite subset.*

One subclass of immune sets that is constantly present in the study of Turing degrees is the class of hyperimmune sets. Immunity of sets and locally finite algebras are connected via the hyperimmune sets. Now we explain that connection.

Consider the listing $x_0 < x_1 < \dots$ of the set X . Call the set X *hyperimmune* if no computable function f exists such that $x_n \leq f(n)$ for all n . Hyperimmune sets are immune and they exist [33].

Theorem 15 (Kasymov and Khossainov [25]). *If $tr(E)$ is hyperimmune then all algebras in $\mathcal{K}(E)$ are locally finite.*

Proof. Let $X = \{[x_1], \dots, [x_k]\}$ be a finite subset of \mathcal{A} . Consider the sequence: $X_0 = \{x_1, \dots, x_k\}$, $X_{n+1} = X_n \cup \{f(\bar{s}) \mid \bar{s} \in X_n, f \text{ is atomic operation}\}$, $n \geq 0$. If the sub-algebra generated by X is infinite, then $\max(X_0), \max(X_1), \dots$ is a computable sequence, and $\max(X_i)$ is greater than or equal to the i th element of the transversal. This is a contradiction. \square

Theorem 16 (Kassymov [20]). *Let $\mathcal{A} \in \mathcal{K}(E)$ be such that all nontrivial quotients of \mathcal{A} are finite. Then either E is computable or $\text{tr}(E)$ is hyperimmune.*

Proof. Let f be a computable function witnessing that $\text{tr}(E)$ is not hyperimmune. For $[x] \neq [y]$, consider the congruence $E(x, y)$. The algebra $\mathcal{A}/E(x, y)$ is finite. This finiteness can effectively be determined through f . Hence, the complement of E is computably enumerable. This is a contradiction. \square

Corollary 17. *Assume that all non-trivial quotients of $\mathcal{A} \in \mathcal{K}(E)$ are finite. Then \mathcal{A} is residually finite and one of the atomic operations of \mathcal{A} has arity > 1 .*

Proof. For the first part, let $[a] \neq [b]$ be such that in all non-trivial quotients of \mathcal{A} the image of $[a]$ equals the image of $[b]$. Then $[x] \neq [y]$ if and only if $(a, b) \in E(x, y)$. Hence E is computable.

For the second part, assume that all atomic operations of \mathcal{A} are unary. Let $\text{Sub}(X)$ be the sub-algebra generated by a finite set X . By the theorem above, $\text{tr}(E)$ is hyperimmune. Hence, the sub-algebra $\text{Sub}(X)$ is finite. Define the following congruence relation E' : $(a, b) \in E' \iff a, b \in \text{Sub}(X) \vee a = b$. This implies that the quotient algebra \mathcal{A}/E' is infinite. This is a contradiction. \square

References to this lecture are [16, 20, 25].

3 Lecture 2: On Finitely Presented Expansions of Algebras

We start the lecture with the following innocent question: *Given an algebra \mathcal{A} , is \mathcal{A} finitely presented?* If S is a finite presentation of \mathcal{A} , then \mathcal{A} is of the form $\mathcal{F}_S = \mathcal{F}/E(S)$, where \mathcal{F} is the term algebra generated by the constants of S ; so, \mathcal{A} is finitely generated by the constants in the language of S , has c.e. equality relation $E = E(S)$ with atomic computable operations respecting the equality. Thus, necessarily, finitely presented algebras are finitely generated and c.e. algebras. It is not surprising that these necessary conditions are not sufficient for algebras to be finitely presented. For instance, Baumslag [6] and Bergstra and Tucker [3] proved (respectively) that the wreath product groups $Z_k \wr Z^n$, with $k, n > 1$, and the algebra $(\omega; x + 1, 2^x, 0)$ are not finitely presented. Thus, given that not every finitely generated c.e. algebra is finitely presented, a natural way to address finite presentability problem is to use expansions of algebras:

Definition 18. *An expansion of $\mathcal{A} = (A; f_1, \dots, f_n, c_1, \dots, c_r)$ is any algebra of the form $\mathcal{A}' = (A; f_1, \dots, f_n, g_1, \dots, g_s, c_1, \dots, c_r)$.*

The idea is to search for a finitely presented expansion of algebra \mathcal{A} in case \mathcal{A} is not finitely presented. As observed above, necessarily, the algebra \mathcal{A} must be a c.e. algebra generated by the constants c_1, \dots, c_r , that is, an E -algebra of the form $(\mathcal{F}/E; f_1, \dots, f_n, c_1, \dots, c_r)$. Also, necessarily, expansions of \mathcal{A} must be computable functions. It is important to note that not every computable function can be used in expansion; the function must respect the relation E .

Bergstra and Tucker [3, 4] proved that every computable algebra has a finitely presented expansion. This solves finite presentability problem for computable algebras. In Lecture 1, we noted that there are finitely presented algebras which are not computable. In the mid 1980s Bergstra-Tucker [4], and independently Goncharov [27], stated the finite presentability problem: Does every finitely generated and c.e. algebra have a finitely presented expansion?

Kassymov [20] and, independently, Khoussainov [23] gave a negative answer. Both gave examples of algebras whose all atomic operations were unary; Khoussainov's algebra was defined based on Kolmogorov complexity. The examples are specific and *ad hoc* tailored towards the solution of the problem. Here we aim to solve this problem in the setting of classical algebra:

Are here finitely generated and computably enumerable semigroups, groups, and algebras that have no finite presentable expansions?

Note that in this question the word algebra refers to rings that are vector spaces over fields rather than algebraic structures in pure functional languages.

3.1 The Non Finite Presentability Theorem (The NFP Theorem)

Our answer to the question above is based on the concept of immunity (mentioned in Lecture 1) borrowed from computability theory. We need to adapt this concept in the context of universal algebras.

Definition 19. *An algebra $\mathcal{A} = \mathcal{F}/E$ is effectively infinite if there is an infinite c.e. sequence t_0, t_1, \dots of pairwise distinct elements of \mathcal{A} . Otherwise, call the algebra \mathcal{A} immune if the algebra is infinite.*

Theorem 20 (Hirschfeldt and Khoussainov [18]). *There exists a computably enumerable, finitely generated, and immune semigroup.*

Proof (Outline). Consider $\mathbf{A} = (\{a, b\}^*; \cdot)$ the free semigroup, where the operation is the concatenation of strings. Let $X \subseteq \{a, b\}^*$ be a nonempty subset. Define: $u \equiv_X v \iff u = v \vee u$ and v have substrings from X .

The equivalence relation \equiv_X has one infinite equivalence class. The class consists of all strings that contain substrings from X . All other equivalence classes are singletons $\{u\}$, where u contains no substring from X . The concatenation operation respects \equiv_X . Hence, the semigroup $\mathbf{A}(X)/\equiv_X$ is well defined.

We claim the following. There is a simple set X such that $\mathbf{A}(X)/\equiv_X$ is infinite, and hence immune. For the proof of the claim we use Post's adapted construction. Consider a standard listing W_0, W_1, \dots of all c.e. subsets of $\{a, b\}^*$. Put string y into X if y is the first string of length $\geq i + 5$ that appeared in W_i for some i . An easy combinatorial argument (similar to the original argument by Post) shows the set X is simple and that there are infinitely many strings that contain no substrings from X . Hence, $\mathbf{A}(X)/\equiv_X$ is infinite. \square

A question arises if there are finitely generated immune groups? Note that such groups answer the generalised Burnside problem. Miasnikov and Osin in [31] constructed such examples. Such groups are now called Dehn monsters. Later we too construct c.e. finitely generated immune groups; but our groups will also be residually finite thus answering the question of Miasnikov and Osin [31].

Now we would like to list some simple properties of immune algebras:

Lemma 21. *Let $\mathcal{A} = \mathcal{F}/E$ be an immune c.e. algebra. Then (1) Each expansion of \mathcal{A} is immune if E is c.e. (2) Each finitely generated sub-algebra of \mathcal{A} is immune or finite. (3) For every term $t(x)$ the trace $a, t(a), tt(a), \dots$ is eventually periodic. (4) Infinite homomorphic images of \mathcal{A} are immune.*

Proof. Items (2), (3), and (4) are easy to prove. For item (1), assume that there is an infinite c.e. sequence t_0, t_1, \dots of pairwise distinct elements of some expansion \mathcal{A}' of \mathcal{A} . Given t_i one can effectively find a term q_i in the language of \mathcal{A} such that $t_i = q_i$ in \mathcal{A} . Hence, \mathcal{A} is effectively infinite which is a contradiction. \square

Lemma 22 (Separator Lemma). *If $\mathcal{A} = \mathcal{F}/E$ is residually finite then for all distinct $x, y \in A$ there is a separator subset $S(x, y) \subset F$ such that $S(x, y)$ is computable and E -closed, and $x \in S(x, y)$ and $y \in F \setminus S(x, y)$.*

Proof. Let h be a homomorphism from $\mathcal{A} = \mathcal{F}/E$ onto finite algebra \mathcal{B} such that $h(x) \neq h(y)$. Consider the set of all pre-images of $h(x)$ in the free algebra \mathcal{F} . In other words, $S(x, y)$ is the set of all ground terms that map onto $h(x)$ under the natural homomorphism from \mathcal{F} onto \mathcal{B} . This set is the desired $S(x, y)$. \square

Lemma 23 (Main Lemma [24]). *If $\mathcal{A} = \mathcal{F}/E$ is immune and residually finite, then so are all expansions of \mathcal{A} .*

Proof. Let x, y be distinct elements of an expansion \mathcal{A}' of \mathcal{A} . Define:

$$[a] \equiv_{(x,y)} [b] \iff \text{no elements in } S(x, y) \text{ and its complement are identified by the congruence relation } E(a, b) \text{ on } \mathcal{A}' \text{ generated by } (a, b).$$

Our goal is to show that the relation $\equiv_{(x,y)}$ is a congruence relation of finite index. For this we need tools borrowed from universal algebra [17]. The first is a characterisation of congruence relations through algebraic terms. The second is Malcev's lemma. These two claims are stated below. Proofs are easily verified. For the first claim, recall that a term for an algebra is *algebraic* if it is of the form $t(x, \bar{g})$ where x is a variable and \bar{g} are parameters from the algebra.

Claim 1: An equivalence relation E on algebra is a congruence relation if and only if all algebraic terms for the algebra respect E .

Claim 2 (Malcev's lemma): Consider the congruence relation $E(a, b)$. Then $(c, d) \in E(a, b)$ iff there are sequences of elements e_0, \dots, e_n , and terms $t_1(x, \bar{g}), \dots, t_n(x, \bar{g})$ such that $e_0 = c$, $e_n = d$, $\{e_0, e_1\} = \{t_1(a, \bar{g}), t_0(b, \bar{g})\}$, $\{e_1, e_2\} = \{t_2(a, \bar{g}), t_1(b, \bar{g})\}$, \dots , $\{e_{n-1}, e_n\} = \{t_n(a, \bar{g}), t_n(b, \bar{g})\}$.

Using these two claims, we now proceed to proving the main lemma.

Property 1: The relation $\equiv_{(x,y)}$ forms an equivalence relation.

Assume $a \equiv_{(x,y)} b$ and $b \equiv_{(x,y)} c$ but not $a \equiv_{(x,y)} c$. This implies that $a, b, c \in S(x, y)$ or $a, b, c \notin S(x, y)$. Since $a \not\equiv_{(x,y)} c$, there is an algebraic term with $t(x)$ such that one of $t(a), t(c)$ belongs to $S(x, y)$ and the other does not. If $t(a) \in S(x, y)$, then $t(b) \in S(x, y)$ since $a \equiv_{(x,y)} b$. Hence, $t(c) \in S(x, y)$ since $b \equiv_{(x,y)} c$. Contradiction.

Property 2: $\equiv_{(x,y)}$ is a congruence relation on \mathcal{A}' .

Otherwise, there exist $a, b \in \mathcal{A}'$ and an algebraic term $t(x)$ with $a \equiv_{(x,y)} b$ and $t(a) \not\equiv_{(x,y)} t(b)$. So, there is an algebraic term $p(y)$ with either $p(t(a)) \in S(x, y)$ and $p(t(b)) \notin S(x, y)$ or $p(t(a)) \notin S(x, y)$ and $p(t(b)) \in S(x, y)$. Contradiction with the fact that $a \equiv_{(x,y)} b$.

Property 3: The relation $\equiv_{(x,y)}$ is a co-c.e. relation.

We have $a \not\equiv_{(x,y)} b$ iff $\exists(c, d) \in E(a, b)(c \in S(x, y) \ \& \ d \notin S(x, y))$ iff there is an algebraic term $t(x, \bar{g})$ such that $c = t(a, \bar{g})$ and $d = t(b, \bar{g})$ and $t(a, \bar{g}) \in S(x, y)$ and $t(b, \bar{g}) \notin S(x, y)$. Hence, the complement of $\equiv_{(x,y)}$ is computably enumerable.

Property 4: The quotient $\mathcal{A}' / \equiv_{(x,y)}$ is finite.

Suppose not. Then the transversal $Tr(\equiv_{(x,y)})$ is a c.e. set. Indeed, t is a minimal element in its equivalence class iff $z \not\equiv_{(x,y)} t$ for all $z < t$. Therefore, $\mathcal{A}' / \equiv_{(x,y)}$ is effectively infinite. Hence \mathcal{A} is effectively infinite. Contradiction.

To complete the proof of the main lemma, note that in the quotient algebra $\mathcal{A}' / \equiv_{(x,y)}$ the images of x and y are distinct. \square

Now we prove our meta-theorem that provides sufficient conditions of finitely generated computably enumerable algebras not to have finite presentation.

Theorem 24 (The NFP Theorem [24]). *Let $\mathcal{A} = \mathcal{F}/E$ be a f.g., c.e., immune, and residually finite algebra. Then \mathcal{A} has no finitely presented expansions.*

Proof. Let \mathcal{A}' be a finitely presented expansion of \mathcal{A} . By the main lemma, \mathcal{A}' is residually finite. Since \mathcal{A}' is residually finite, by Malcev/McKenzie theorem that we proved in Lecture 1, the equality E in \mathcal{A}' is decidable. Contradiction. \square

3.2 Application of the NFP Theorem

We apply the NFP theorem to build c.e. finitely generated semigroups, algebras, and groups that have no finitely presented expansions.

Theorem 25 (Semigroups case [24]). *There exists a c.e., finitely generated, and immune semigroup that has no finitely presented expansion.*

Proof. The semigroup $\mathbf{A}(X)$ that built in Theorem 20 is c.e., finitely generated, immune, and residually finite. Apply the NFP theorem to the semigroup. \square

To apply the NFP theorem to algebras and groups, we employ the celebrated theorem of Golod and Shafarevich. Here is the set-up for GS Theorem.

Let K be a finite field. Let $\mathcal{P} = K\langle x_1, x_2, \dots, x_s \rangle$ be the algebra of polynomials in non-commuting variables. Here the term algebra refers to a non-commutative ring that forms a vector space over K . View \mathcal{P} as the direct sum

$$\mathcal{P} = \sum_n \mathcal{P}_n$$

where \mathcal{P}_n is the vector space spanned over s^n monomials of degree n .

Let H be a set of homogeneous polynomials in the algebra \mathcal{P} . Recall that a polynomial is homogeneous if all of its monomials have the same degree. Let $I = \langle H \rangle$ be the two sided ideal generated by H .

Theorem (Golod and Shafarevich [13, 14]). *Let r_n be the number of polynomials in H of degree n . Let ϵ be such that $0 < \epsilon < s/2$ and $r_n \leq \epsilon^2 \cdot (s - 2\epsilon)^{n-2}$ for all n . Then the algebra*

$$\mathcal{A} = \mathcal{P}/I = \sum_n \mathcal{P}_n/I$$

is infinite dimensional.

Theorem 26 (Algebra case [24]). *There exists a c.e., finitely generated, and immune algebra that has no finitely presented expansion.*

Proof (Outline). Use Post's type of construction to build a simple set H of homogeneous polynomials. Also, satisfy the assumption of Golod-Shafarevich theorem. Here is an outline of the construction. List W_i , $i \in \omega$ all c.e. subsets of \mathcal{F} . Let f and g be polynomials in W_i such that:

1. $f = f_1 + f_2$, $g = g_1 + g_2$, $f_1 = g_1$, and
2. The degrees of homogeneous polynomials occurring in both f_2 and g_2 are greater than $i + 10$.

Put f_2 and g_2 into H . Let $I = \langle H \rangle$ be the ideal generated by H . Define: $\mathcal{A} = \mathcal{P}/I$. The algebra \mathcal{A} is residually finite because the ideal I is generated by homogeneous polynomials. Now apply the NFP theorem to the algebra. \square

Theorem 27 (Groups case [24]). *There exists a c.e., finitely generated and immune group G that has no finitely presented expansion.*

Proof (Outline). We use Golod's conversion of the algebra \mathcal{A} built in the previous theorem to a group [13]. For simplicity assume that the algebra \mathcal{A} above is built from \mathcal{P} with exactly two variables x and y . Consider the semigroup $G = G(\mathcal{A})$ generated by $(1 + x)/I$ and $(1 + y)/I$. The semigroup is a group as observed by Golod. The group G is infinite and residually finite. It is immune as well. Apply the NFP theorem to the group. \square

References to this lecture are [18, 24].

4 Lecture 3: Reducibilities on Equivalence Relations

The previous two lectures show deep interactions between computability theoretic properties of the domains ω/E and algebraic properties of E -algebras. As mentioned in Lecture 1, for any class \mathcal{C} of algebras, the following equality holds:

$$\mathcal{C} = \bigcup_E \mathcal{K}(\mathcal{C}, E)$$

This implies that the domains ω/E can be used as taming tools of algebras from class \mathcal{C} . This lecture aims at formalising these interactions.

We extend the definition of E -algebra to general algebraic structures. An n -ary relation R on ω respects E if for all $s_1, \dots, s_n, q_1, \dots, q_n$ such that $(s_i, q_i) \in E$ for $i = 1, \dots, n$, we have $R(s_1, \dots, s_n)$ iff $R(q_1, \dots, q_n)$. So, if R respects E then the truth-value of R does not depend on representatives of equivalence classes.

Definition 28. *The quotient structure $(\omega/E; f_1, \dots, f_n, P_1, \dots, P_k)$, where f_1, \dots, f_n are computable operations and all P_1, \dots, P_k are c.e. relations that all respect E , is called an E -structure.*

Let \mathcal{C} be a class of structures. The central definition of the third lecture is this. The terminology in this definition is borrowed from computability theory:

Definition 29. *For equivalence relations E_1 and E_2 we say that E_1 is \mathcal{C} -reducible to E_2 , written $E_1 \leq_{\mathcal{C}} E_2$, if every structure realised by E_1 is also realised by E_2 . The equivalence relations E_1 and E_2 have the same \mathcal{C} -degree, written $E_1 \equiv_{\mathcal{C}} E_2$, if $E_1 \leq_{\mathcal{C}} E_2$ and $E_2 \leq_{\mathcal{C}} E_1$. The \mathcal{C} -degree of E is denoted by $E/\equiv_{\mathcal{C}}$.*

Every class \mathcal{C} of structures defines the preorder $\leq_{\mathcal{C}}$ between equivalence relations E (or domains ω/E). Thus, we have the following partially ordered set:

$$\mathcal{E}_{\mathcal{C}} = (\{E/\equiv_{\mathcal{C}} \mid E \text{ is c.e. equivalence relation}\}, \leq_{\mathcal{C}}).$$

Our aim is to study the orders $\mathcal{E}_{\mathcal{C}}$. We restrict ourselves to c.e. E equivalence relations because we would like our domains ω/E to be (in some ways) effective.

4.1 The Class Alg of Algebras

We exhibit basic properties of $\mathcal{E}_{\mathit{Alg}}$. Call a function $f : A^n \rightarrow A$ *trivial* if it is a projection function or f is a constant function, that is, either there is an i such that $f(x_1, \dots, x_n) = x_i$ or there is an $a \in A$ for which $f(x_1, \dots, x_n) = a$ for all $(x_1, \dots, x_n) \in \omega^n$. The identity function is obviously a trivial function. Call an algebra *trivial* if all of its atomic operations are trivial. The following is clear:

Lemma 30. *For all E every trivial algebra is an E -algebra. In other words, the classes $\mathcal{K}(\mathit{Alg}, E)$ all realise all trivial algebras. \square*

It turns out there exists an equivalence relation E that realises trivial algebras only. In particular, this implies that the partial order $\mathcal{E}_{\mathit{Alg}}$ has the least element:

Theorem 31 ([22]). *There is a c.e. equivalence relation E such that the class $\mathcal{K}(\text{Alg}, E)$ consists of trivial algebras only. Hence, E is the minimal \equiv_{Alg} -degree.*

Proof (Outline). A set $X \subseteq \omega$ is called E -closed if it is a union of E -equivalence classes. For the proof we use a c.e. equivalence relation E that satisfies the following properties: (1) the quotient set ω/E is infinite, (2) every c.e. E -closed set is either ω or \emptyset or a finite union of E -equivalence classes, (3) No c.e. set W exists such that both $\text{Tr}(E) \cap W$ and $\text{tr}(E) \cap (\omega \setminus W)$ are infinite. Such equivalence relations exist [8, 22]. Let us fix one such equivalence relation E .

A routine combinatorial argument will show that every computable function that respects E is trivial on the domain ω/E . To make our argument easy we assume that f respects E and f is a unary computable operation on ω .

Assume that f is not a constant function on ω/E . Define the following equivalence relation E' : $(n, m) \in E'$ iff $\exists k((f^k(n), m) \in E \vee (f^k(m), n) \in E)$, where $f^0(a) = a$, $f^{k+1}(a) = f(f^k(a))$ for $k \in \omega$. Note that E' is c.e. and $E \subseteq E'$. The relation must have infinitely many equivalence classes since, otherwise, there would be an E -closed c.e. set that is a union of infinitely many E -equivalence classes. By the choice of E , this is impossible. Also, all but finitely many equivalence classes of E' must coincide with equivalence classes of E . Otherwise, for the set $W = \omega \setminus \text{tr}(E')$ both $\text{Tr}(E) \cap W$ and $\text{tr}(E) \cap (\omega \setminus W)$ are infinite. Hence, the set $A = \{x \mid (f(x), x) \in E\}$ is computably enumerable, E -closed, and consists of infinitely many E -equivalence classes. Therefore $A = \omega$. \square

Thus, for the minimal Alg -degree E we can fully characterise the isomorphism types of E -algebras. These are just trivial algebras. This is a good example showing the way the domains ω/E tame the isomorphism types of E -structures.

Theorem 32 ([12]). *Suppose that the class $\mathcal{K}(\text{Alg}, E)$ contains a finitely generated algebra. Then E is a maximal \equiv_{Alg} -degree.*

Proof. Let \mathcal{A} be a finitely generated E -algebra. Let E' be a c.e. equivalence relation such that there is an E' -algebra \mathcal{B} isomorphic to \mathcal{A} . We need to show that $\mathcal{K}(\text{Alg}, E) = \mathcal{K}(\text{Alg}, E')$. Let $[n_1]_E, \dots, [n_k]_E$ be the generators of \mathcal{A} . For each element $[n]_E$ of \mathcal{A} there is a term $t(x_1, \dots, x_k)$ such that $[n]_E = [t(n_1, \dots, n_k)]_E$. Let $[m_1]_{E'}, \dots, [m_k]_{E'}$ be the images of $[n_1]_E, \dots, [n_k]_E$ in \mathcal{B} . Then by mapping $t(n_1, \dots, n_k)$ to $t(m_1, \dots, m_k)$ we can set up a computable mapping $g : \omega \rightarrow \omega$ that induces a bijection from ω/E onto ω/E' . Through this computable function one can show that any E -algebra is isomorphic to an E' -algebra. \square

In contrast to the theorem above, Khoussainov et al. [26] provide an example of an equivalence relation E such that E is a maximal \equiv_{Alg} -degree but the class $\mathcal{K}(\text{Alg}, E)$ contains no finitely generated algebra.

4.2 The Class LO of Linear Orders

It is important to clarify the definition of linear order. We always assume that linear orders are defined by binary relations that are reflexive, antisymmetric and transitive. Other definitions of linear orders might lead to different results.

Let $\mathcal{L} = (L, \leq)$ be a linear order. An element a is *isolated* if either it has immediate successor and immediate predecessor or it is the left-most element with immediate successor or it is the right-most element with immediate predecessor.

Lemma 33. *If a linear order \mathcal{L} is an E -structure, then every isolated element $a = [n]$ of \mathcal{L} is a computable E -equivalence class.*

Proof. Assume that a has immediate successor $[x]$ and immediate predecessor $[y]$, then $\omega = \{i \mid i \leq y\} \cup [a] \cup \{j \mid x \leq j\}$. Thus, the set $[a]$ is computable. \square

Corollary 34. *Let id_ω be the identity relation on ω . If E contains a non-computable c.e. equivalence class then $id_\omega \not\leq_{LO} E$.* \square

For the rest of this subsection we consider the equivalence relations $E(X)$, where $X \subseteq \omega$ is an infinite and co-infinite set, defined as follows:

$$(n, m) \in E(X) \iff (n = m) \vee (n \in X \ \& \ m \in X).$$

Each equivalence class of $E(X)$ is thus either a singleton of the form $\{n\}$, where $n \notin X$, or the infinite set X . We aim to study the classes $\mathcal{K}(LO, E(X))$ and the partial order \leq_{LO} restricted to the equivalence relations of type $E(X)$:

$$\mathcal{P} = \{E(X) / \equiv_{LO} \mid X \text{ is an infinite and co-infinite c.e. set}\}.$$

We explain some interesting properties of this partial order.

We need semi-recursive sets introduced by Jockusch [19]. A set $A \subseteq \omega$ is *semi-recursive* if there exists a computable function $f : \omega^2 \rightarrow \omega$ such that for all $x, y \in \omega$ we have (a) $f(x, y) = x$ or $f(x, y) = y$ and (b) if $x \in A$ or $y \in A$ then $f(x, y) \in A$. Jockusch [19] proves the following result that we state as a lemma:

Lemma 35 (Jockusch [19]). *A set A is semi-recursive if and only if A is an initial segment of some computable linear order on ω .* \square

We use this lemma in the following characterisation theorem:

Theorem 36 ([12]). *Let X be a coinfinite c.e. set. Then we have the following:*

1. *The equivalence relation $E(X)$ realises a linear order in which X represents an isolated point iff X is computable.*
2. *The equivalence relation $E(X)$ realises a linear order with X being an end point iff X is semirecursive;*
3. *The equivalence relation $E(X)$ realises a linear order iff X is one-one reducible to the join of two c.e. semirecursive sets.*

Proof. Part (1) follows from Lemma 33. Part (2) is Jockusch lemma stated above.

We prove (3). Assume that $E(X)$ realises a linear order \sqsubseteq and $x \in X$. Then one can define the sets $Y = \{y \mid y \sqsubseteq x\}$ and $Z = \{z \mid x \sqsubseteq z\}$. Both Y and Z are semirecursive. Now it is routine to show that $X \leq_1 Y \oplus Z$.

Let X be one-one reducible via f to the join $Y \oplus Z$ of two c.e. semirecursive sets. Equip Y and Z with computable orderings \sqsubseteq_Y and \sqsubseteq_Z , respectively, where Y is the upper end point of \sqsubseteq_Y and Z is the lower end point of \sqsubseteq_Z . Take the disjoint sum of \sqsubseteq_Y and \sqsubseteq_Z , and pull this linear order back via f to ω . The result is a c.e. binary relation \sqsubseteq' that induces a linear order on $\omega/E(X)$. \square

Corollary 37. *If $E(X)$ realises a linear order then X is neither hyperhyper-simple nor creative nor r -maximal nor simple and non-hypersimple.* \square

We now fully characterise the isomorphism types of linear orders realised by equivalence relations $E(X)$, where X is a simple set:

Theorem 38 ([12]). *If X is simple, then one of the following occurs:*

1. *If X is not one-one reducible to the join of two semirecursive sets then $E(X)$ does not realise any linear order;*
2. *If X is semirecursive $E(X)$ realises the linear orders $\omega + n$, $n + \omega^*$ and $\omega + 1 + \omega^*$ for all n ;*
3. *If X is one-one reducible to the join of two semirecursive sets but not semirecursive then $E(X)$ realises exactly the linear order which is $\omega + 1 + \omega^*$.*

Proof. Part (1) follows from the previous theorem. We note that simple sets satisfying the hypothesis exist (e.g. maximal sets).

For part (2), let X be simple and semirecursive. Let \sqsubseteq be a c.e. order respecting $E(X)$ and y represents an element larger than X then there are only finitely many y with $x \sqsubseteq y$ due to simplicity; furthermore, if y represents an element below X then there are only finitely many y with $y \sqsubseteq x$. Hence only the orders $\omega + n$, $n + \omega^*$ and $\omega + 1 + \omega^*$ can be represented. Furthermore, there is a linear order represented by $E(X)$ and this linear order has X as its endpoint; hence $\omega + 1$ and $1 + \omega^*$ can be represented; and thus $\omega + n$ and $n + \omega^*$ can also be represented. Arguments similar to the proof of Part (3) of Theorem 36 provide an order of the type $\omega + 1 + \omega^*$ realised by $E(X)$.

For part (3), let X be simple and one-one reducible to the join of two semirecursive sets and not semirecursive. We use the proof of Part (3) of Theorem 36. That proof shows that $E(X)$ realises an infinite order and X is not an end point of this order. As one can modify the ordering on finitely many elements and X cannot be made an end point of the ordering, the ordering cannot be of the form $\omega + n$ or $n + \omega^*$. Hence the order must be $\omega + 1 + \omega^*$. \square

Consider the subset \mathcal{S} of LO-degrees from \mathcal{P} : $\mathcal{S} = \{E(X)/\equiv_{LO} \mid X \text{ is simple}\}$. From the results above we get the next corollary:

Corollary 39. *The partial order \leq_{LO} on \mathcal{S} is a 3-element linear order.*

Proof. Let X_1 , X_2 , and X_3 be simple sets such that (1) $E(X_1)$ realises no linear order; (2) The only linear order realized over $E(X_2)$ is of the type $\omega + 1 + \omega^*$; (3) The only linear orders realized over $E(X_3)$ are of the form, $\omega + 1 + \omega^*$, $\omega + n$ or $n + \omega^*$, where $n \in \omega$. Clearly, $E(X_1) <_{LO} E(X_2) <_{LO} E(X_3)$. As mentioned above, these are the only possibilities that occur for simple sets. \square

Corollary 40. *The partial order \leq_{LO} on \mathcal{P} has a least and greatest element.*

Proof. The least element is witnessed by $E(X)$, where X is a maximal set. The largest element $E(Y)$ is witnessed by a computable set Y , as every linear order realized over any $E(Z)$ has a computable copy, that is, realized over $E(Y)$. For the proof of the last part, that requires a deeper argument, see [12]. \square

Corollary 41. *\mathcal{S} is an initial segment of \mathcal{P} .*

Proof. Let X be a non-simple set and $(\omega/E(X), \trianglelefteq)$ be a linear order. Fix infinite computable $C \subseteq \omega \setminus X$. Construct a new linear order by restricting to \trianglelefteq on $\omega \setminus C$ and placing C to the right of $\omega \setminus C$. That is, we define $a \hat{\trianglelefteq} b \iff a \trianglelefteq b$ for $a, b \in \omega \setminus C$, and $a \hat{\trianglelefteq} b$ for $a \in \omega \setminus C$ and $b \in C$. We then define $\hat{\trianglelefteq}$ on C to be any computable linear order. Thus $E(X)$ is not \leq_{LO} -below any LO -degree in \mathcal{S} . \square

4.3 Class *Part*

A *graph* is a structure of the form $(V; Edge)$, where *Edge* consists of unordered pairs. So, these are just undirected graphs where self-loops are not allowed.

Definition 42. *A graph $G = (V, Edge)$ is a partition graph if there is a partition A_0, A_1, \dots of V such that $Edge = \{\{x, y\} \mid \exists i \exists j (x \in A_i \ \& \ y \in A_j \ \& \ i \neq j)\}$.*

Sets A_i from the definition above are called *anti-clique components* of \mathcal{G} . Denote the class of all partition graphs by *Part*. We study the partial order \mathcal{E}_{Part} .

There are two trivial partition graphs. One is the complete graph K_ω , and the other is the graph *Isl* whose all vertices are isolated.

Theorem 43 ([15]). *The graphs K_ω and *Isl* have the following properties: (1) If K_ω is realised by E then E is computable. (2) Every E realises *Isl*. \square*

Important equivalence relations are precomplete equivalence relations [30]. A c.e. equivalence relation E is *precomplete* if for every partial-recursive function $\psi : \omega \rightarrow \omega$ there is a total-recursive function f such that for all $n \in dom(\psi)$, we have $\psi(n) E f(n)$. Lachlan [28] showed that all precomplete universal equivalence relations are computably isomorphic. No two distinct equivalence classes of precomplete equivalence relations are recursively separable [30].

Theorem 44 ([15]). *Every precomplete equivalence relation realises only the trivial partition graph *Isl*. So, the order \mathcal{E}_{Part} has the least element.*

Proof. Let E be a precomplete equivalence relation. Assume that a partition graph $G = (\omega/E, Edge)$ is realised by E and \mathcal{G} has at least two anti-clique components. Let us select x and y from these two anti-clique components. For every z we define $f(z) = 0$ if $\{x, z\}$ is enumerated into *Edge* before $\{y, z\}$ and define $f(z) = 1$ if $\{y, z\}$ is enumerated into *Edge* before $\{x, z\}$. The function f recursively separates the E -equivalence classes of x and y . Hence, G is *Isl*. \square

The natural question is if \mathcal{E}_{Part} has the greatest element. In [15] a positive answer is given; the equivalence relation id_ω is the greatest *Part*-degree in \mathcal{E}_{Part} .

Lemma 45. *Let $[x]$ be a non-computable equivalence class of E . Then if G is realised by E , then $[x]$ belongs to an infinite anti-clique component.*

Proof. Let $G = (V, Edge)$ be realised by E and that the anti-clique component of \mathcal{G} containing $[x]$ is finite. Let x, y_1, y_2, \dots, y_n be representatives of the anti-clique component. Decide the equivalence class of x as follows. On input z , search until either $x E z$ or $z E y_m$ for $m \in \{1, 2, \dots, n\}$ or $(x, z) \in Edge$. These conditions are disjoint c.e. events. If $x E z$ then $z \in [x]$. Otherwise, $z \notin [x]$. \square

Corollary 46. *If E realises a partition graph whose all anti-clique components are finite, then each equivalence class of E is recursive.* \square

Let E_0, E_1, \dots be an effective enumeration of all c.e. equivalence relations. Define: $(\langle x, y \rangle, \langle x', y' \rangle) \in univ \Leftrightarrow x = x' \wedge (y, y') \in E_x$ [30]. The relation $univ$ is a universal c.e. equivalence relation [7] in the sense that all c.e. equivalence relations are m -reducible to it.

Let E_z be a computably enumerable equivalence relation realising a partition graph $G = (\omega/E; Edge')$ with at least one infinite anti-clique component. Fix z' , a member of an infinite anti-clique component. Define $Edge$ relation on $\omega/univ$:

$$\begin{aligned} (\langle x, y \rangle, \langle x', y' \rangle) \in Edge &\Leftrightarrow (x = z \wedge x' = z \wedge (y, y') \in Edge') \\ &\vee (x = z \wedge x' \neq z \wedge (y, z') \in Edge') \\ &\vee (x \neq z \wedge x' = z \wedge (y', z') \in Edge'). \end{aligned}$$

The graph $(\omega; Edge)/univ$ is isomorphic to the graph \mathcal{G} . This observation with the lemma above implies the following characterisation theorem:

Theorem 47 ([15]). *A c.e. partition graph $G = (V; Edge)$ is realised by the relation $univ$ iff one of the anti-clique components of \mathcal{G} is infinite.* \square

Now we concentrate on special subclass of partition graphs. Call an infinite partition graph \mathcal{G} *finitary* if the number of its anti-clique components is finite.

Definition 48. *A c.e. equivalence relation E finitary if all partition graphs realised by E are finitary. Let \mathcal{F} be the class of all c.e. finitary equivalence relations.*

It is easy to fully characterise isomorphism types of finitary partition graphs.

Definition 49. *The isomorphism invariant of a finite partition graph \mathcal{G} is the tuple (i, m, k_1, \dots, k_m) , where i is the number of infinite anti-clique components, m is the number of finite anti-clique components and k_1, \dots, k_m is the sequence, in non-decreasing order, of the cardinalities of all finite anti-clique components of the graph \mathcal{G} . Call the pair (i, m) the type of the graph \mathcal{G} .*

Two finitary partition graphs are isomorphic iff they have the same isomorphism invariants. The following two lemmas are easy:

Lemma 50. *Let $G = (\omega/E; Edge)$ be a c.e. finitary partition graph. Then all anti-clique components of G are computable.*

Proof. Let x_1, \dots, x_k be representatives of the anti-clique components. The anti-clique component of x_i is $A_i = \{y \mid (\forall j \neq i, 1 \leq j \leq k)[(y, x_j) \in Edge]\}$. Since the A_i 's partition ω , we have that A_i is computable. \square

Lemma 51. *If E is a c.e. equivalence relation with no computable classes, then all anti-clique components in partition graphs realised by E are infinite.* \square

Lemma 52. *For a finitary equivalence relation E , let $n(E)$ be the maximum i such that E realises a partition graph with i many infinite anti-clique components, and $m(E)$ be the maximum i such that E realises a partition graph with i many finite anti-clique components. Then for all (n, m) with $1 \leq n < 1 + n(E)$, $m < 1 + m(E)$, E realises finitary partition graphs of type (n, m) .*

Proof. By Lemma 45 every equivalence class $[x]_E$ that belongs to a finite anti-clique component of some E -partition graph must be computable. Hence, E has exactly $m(E)$ computable equivalence classes.

Case 1: $n(E) = \omega$ & $m(E) = \omega$. Take an isomorphism invariant (n, m, k_1, \dots, k_m) . Let \mathcal{G} be any E -partition graph of type (n', m') , where $n' \geq n$. Select an infinite anti-clique component, say C , in \mathcal{G} . We change \mathcal{G} to \mathcal{G}' by (1) adding all $[x]_E$ that belong to finite anti-clique components of \mathcal{G} to C , and (2) combining $n' - n$ many infinite anti-clique components all different from C with C . In this way, we changed \mathcal{G} to \mathcal{G}' in which the original C has enlarged to a new anti-clique component. The new graph \mathcal{G}' has type $(n, 0)$. Now select $k_1 + \dots + k_m$ recursive E -equivalence classes and change \mathcal{G}' to \mathcal{G}'' by forming new m many anti-clique components (using these $k_1 + \dots + k_m$ E -equivalence classes) of cardinalities k_1, \dots, k_m . The resulting graph \mathcal{G} has type (n, m) .

The other 3 cases when $n(E) = \omega$ & $m(E) = r < \omega$, $n(E) < \omega$ & $m(E) = \omega$, and $n(E) < \omega$ & $m(E) < \omega$ are treated in a similar fashion. \square

Definition 53. *A c.e. equivalence relation E has type (n, m) if n and m are the largest cardinalities such that for all i, j with $1 \leq i < n$ and $j < m$, the relation E realises finitary partition graphs of type (i, j) .*

We would like to construct finitary equivalence relations.

Theorem 54 ([15]). *Let X be a simple set. Then $E(X)$ realises a partition graph \mathcal{G} iff \mathcal{G} is finitary whose type is of the form $(1, m)$, where $m \in \omega$.*

Proof. Suppose $G = (\omega/E(X); Edge)$ is a partition graph realised by $E(X)$. Then the anti-clique component of G containing X is infinite by Lemma 45. Let $x \in X$. The set $S = \{z \mid (z, x) \in Edge\}$ is c.e., respects E , and is contained in the complement of X . Thus, S is finite. So, G is of type $(1, m)$ for some m .

Let G be a partition graph with isomorphism invariant $(1, m, k_1, \dots, k_m)$. To build an $E(X)$ -graph isomorphic to \mathcal{G} , select a set $T \subset \omega - X$ of size $k_1 + \dots + k_m$, and build a partition graph with anti-clique components $\omega - T, T_1, \dots, T_m$, where T is the disjoint union of T_1, \dots, T_m of cardinalities k_1, \dots, k_m , respectively. \square

Thus, for simple sets X , the c.e. equivalence relation $E(X)$ has type $(2, \omega)$. By Theorem 44, a precomplete c.e. equivalence relation E has type $(2, 1)$. The next lemma gives us a full description of finitary c.e. equivalence relations. The proof is a combinatorial argument that uses the equivalence relation $E(X)$, where X is simple, precomplete equivalence relations, and their finite joins and products.

Lemma 55. *For each pair (n, m) such that $1 < n \leq \omega$ and $1 \leq m \leq \omega$ there exists a c.e. equivalence relation of type (n, m) . \square*

The lemmas above imply the full characterisation of *Part*-reducibility in the class \mathcal{F} of all finitary c.e. equivalence relations.

Theorem 56 ([15]). *The degree-structure of \mathcal{F} ordered by *Part*-reducibility is isomorphic to the two-dimensional grid-order $(\{(n, m) \mid n, m \in \omega \cup \{\omega\}\}; \leq)$, where \leq is the component-wise order on the set of pairs. \square*

References for this last lecture with many more results are [10, 12, 15].

References

1. Andrews, U., Sorbi, A.: Joins and meets in the structure of Ceers. [arXiv:1802.09249](https://arxiv.org/abs/1802.09249) (2018, submitted)
2. Andrews, U., Lempp, S., Miller, J., Ng, K., Mauro, L.S., Sorbi, A.: Universal computably enumerable equivalence relations. *J. Symb. Log.* **79**(1), 60–88 (2014)
3. Bergstra, J.A., Tucker, J.V.: Initial and final algebra semantics for data type specifications: two characterization theorems. *SIAM J. Comput.* **12**, 366–387 (1983)
4. Bergstra, J.A., Tucker, J.V.: Algebraic specifications of computable and semi-computable DataTypes. *Theor. Comput. Sci.* **50**, 137–181 (1987)
5. Bernardi, C., Sorbi, A.: Classifying positive equivalence relations. *J. Symb. Log.* **48**(3), 529–538 (1983)
6. Baumslag, G.: Wreath products and finitely presented groups. *Math. Z.* **75**, 22–28 (1960/1961)
7. Ershov, Y.L.: Positive equivalence. *Algebra Log.* **10**(6), 378–394 (1974)
8. Ershov, Y.L.: *Theory of Numberings*. Nauka, Moscow (1977). (in Russian)
9. Ershov, Y.L., Goncharov, S.S.: *Constructive models*. Transl. from the Russian. (English) *Siberian School of Algebra and Logic*, vol. xii, 293 p. Consultants Bureau, New York (2000)
10. Fokina, E., Khoussainov, B., Semukhin, P., Turetsky, D.: Linear orders realized by ce equivalence relations. *J. Symb. Log.* **81**(2), 463–482 (2016)
11. Gao, S., Gerdes, P.: Computably enumerable equivalence relations. *Stud. Logica* **67**(1), 27–59 (2001)
12. Gavryushkin, A., Khoussainov, B., Stephan, F.: Reducibilities among equivalence relations induced by recursively enumerable structures. *Theoret. Comput. Sci.* **612**, 137–152 (2016)
13. Golod, E.S.: On nil-algebras and finitely approximable p-groups. *Izv. Akad. Nauk SSSR Ser. Mat.* **28**, 273–276 (1964). (Russian)
14. Golod, E.S., Shafarevich, I.R.: On the class field tower. *Izv. Akad. Nauk SSSR* **28**, 261–272 (1964). (in Russian)

15. Gavruskin, A., Jain, S., Khoussainov, B., Stephan, F.: Graphs realised by R.E. equivalence relations. *Ann. Pure Appl. Logic* **165**(7–8), 1263–1290 (2014)
16. Goncharov, S., Khoussainov, B.: Open problems in the Theory of Constructive Algebraic Systems, *Contemporary Mathematics* 257, *Computability Theory and Its Applications (Current Trends and open Problems)*, pp. 145–170 (2000)
17. Gratzner, G.: *Universal Algebra*. Van Nostrand, Princeton (1968)
18. Hirschfeldt, D., Khoussainov, B.: On finitely presented expansions of computably enumerable semigroups. *Algebra Log.* **51**(5), 435–444 (2012)
19. Jockusch, C.J.: Semirecursive sets and positive reducibility. *Trans. Am. Math. Soc.* **131**, 420–436 (1968)
20. Kassymov, N.K.: On finitely approximable and C.E. representable algebras. *Algebra Log.* **26**(6) (1986)
21. Khalimulin, I., Khoussainov, B., Melnikov, A.: Limit-wise monotonicity and the degree spectra of structures. *Proc. Am. Math. Soc.* **141**, 3275–3289 (2013)
22. Khoussainov, B.: Quantifier free definability on infinite algebras. In: *Proceedings of Logic in Computer Science Conference, LICS 2016*, pp. 730–738 (2016)
23. Khoussainov, B.: Randomness, computability, and algebraic specifications. *Ann. Pure Appl. Logic* **91**(1), 1–15 (1998)
24. Khoussainov, B., Miasnikov, A.: Finitely presented expansions of groups, semigroups, and algebras. *Trans. Amer. Math. Soc.* **366**(3), 1455–1474 (2014)
25. Kasymov, N., Khoussainov, B.: Finitely generated enumerable and absolutely locally finite algebras. *Vychislitelnye Sistemy* **116**, 3–15 (1986). (in Russian)
26. Khoussainov, B., Lempp, S., Slaman, T.A.: Computably enumerable algebras, their expansions, and isomorphisms. *Int. J. Algebra Comput.* **15**, 437–454 (2005)
27. Ershov, Y., Goncharov, S. (eds.): *Logic Notebook (Open questions in Logic)*. Novosibirsk University Press (1989)
28. Lachlan, A.H.: A note on positive equivalence relations. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* **33**, 43–46 (1987)
29. Mal'cev, A.I.: Constructive algebras. I. *Uspehi Mat. Nauk* **16**(3(99)), 3–60 (1961)
30. Maltsev, A.I.: Towards a theory of computable families of objects. *Algebra i Logika* **3**(4), 5–31 (1963)
31. Miasnikov, A., Osin, D.: Algorithmically finite groups. *J. Pure Appl. Algebra* **215**(11), 2789–2796 (2011)
32. Rabin, M.O.: Computable algebra, general theory and theory of computable fields. *Trans. Amer. Math. Soc.* **95**, 341–360 (1960)
33. Soare, R.I.: *Recursively Enumerable Sets and Degrees*. Springer, Berlin (1987)



Polynomial-Time Presentations of Algebraic Number Fields

Pavel Alaev^{1,2(✉)} and Victor Selivanov^{3,4}

¹ S. L. Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia
alaev@math.nsc.ru

² Novosibirsk State University, Novosibirsk, Russia

³ A. P. Ershov Institute of Informatics Systems SB RAS,
Novosibirsk, Russia
vseliv@iis.nsk.su

⁴ Kazan Federal University, Kazan, Russia

Abstract. Using an extension of the notion of polynomial time presentable structure we show that some natural presentations of the ordered field \mathbb{R}_{alg} of algebraic reals and of the field \mathbb{C}_{alg} of algebraic complex numbers are polynomial-time equivalent to each other and are polynomial time. We also establish upper complexity bounds for the problem of rational polynomial evaluation in \mathbb{C}_{alg} and for the problem of root-finding for polynomials in $\mathbb{C}_{\text{alg}}[x]$ which improve the previously known bound.

Keywords: Algebraic number · Ordered field · Polynomial Polynomial-time presentable structure · Complexity bound

1 Introduction

Based on the notion of a computable structure, the computability issues in algebra and model theory were thoroughly investigated. In particular, a rich and useful theory of computable fields was developed (see e.g. [1, 2] and references therein). For instance, Rabin [3] has shown that the algebraic closure of a computable field is computably presentable, and Ershov [4] has shown that the real algebraic closure of a computable ordered field is computably presentable. Since the ordered field \mathbb{Q} of rationals is computably presentable, the field $\mathbb{C}_{\text{alg}} = (C_{\text{alg}}; +, \times, 0, 1)$ of complex algebraic numbers and the ordered field $\mathbb{R}_{\text{alg}} = (R_{\text{alg}}; \leq, +, \times, 0, 1)$ of algebraic reals are computably presentable.

In applications one of course has to pay attention to the complexity of implemented algorithms and of structure presentations. The complexity of structure

P. Alaev—The work of first author was funded by RFBR, the research project 17-01-00247.

V. Selivanov—The work of second author was funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities, project No 1.12878.2018/12.1.

presentations was studied e.g. in [5–7] where, in particular, the notion of a polynomial-time (p-time) structure was introduced. To our knowledge, the complexity issues for presentations of fields were not studied in computability theory so far. At the same time, there exists a well-developed theory of symbolic computations (closely related to computer algebra) which investigates the complexity of algorithms in fields, of concrete presentations of fields and rings, and aims to implement these in computer systems. In particular, there is a vast literature around the Tarski theorems on decidability of the theories of algebraically closed fields and of real closed fields (see e.g. [8] and references therein). Although the mentioned theories are obviously intimately related, they developed apparently independently and there are essentially no references between them.

This paper has two aims. First, we partially fill the gap between the mentioned theories by applying the notions of computability theory to the investigation of some presentations of \mathbb{R}_{alg} and \mathbb{C}_{alg} . In particular, we show that the notion of p-time presentable structure is not applicable to some presentations of \mathbb{R}_{alg} and \mathbb{C}_{alg} in the literature. We introduce in Sect. 2 a more general notion of p-computable quotient-structure and show that several natural presentations of \mathbb{R}_{alg} and \mathbb{C}_{alg} are p-time equivalent to each other, and are p-time computable. Thus, we clear up the conceptual basis for the complexity theory of structure presentations.

The second aim is to study the complexity of some algorithmic problems in \mathbb{R}_{alg} and \mathbb{C}_{alg} . In Sect. 3 we find an optimal (in a sense) complexity bound for the problem of rational polynomial evaluation. In Sect. 4 we find an upper bound for the problem of root-finding for polynomials in $\mathbb{C}_{\text{alg}}[x]$ which improves the bounds previously known in the literature on computer algebra.

2 Presentations of Algebraic Number Fields

We use multi-tape Turing machines as our basic model of computation (see e.g. [9, §1.6] for additional details). Let Σ be a finite alphabet. Suppose that $f : A \rightarrow \Sigma^*$, where $A \subseteq (\Sigma^*)^n$. We say that f is *computable on a k -tape Turing machine T in $t(\bar{x})$ steps*, where $t : A \rightarrow \mathbb{N}$, if $k \geq n + 1$ and we can write the words $\bar{x} = x_1, \dots, x_n$ from A on the first n tapes, run the machine T , which stops in at most $t(\bar{x})$ steps, and obtain $f(\bar{x})$ written on $(n + 1)$ -st tape. More precise definitions may be found, for example, in [10]. We say that f is *computable in polynomial time (p-computable)* if we can choose $t(\bar{x}) = c|\bar{x}|^m$ for $|\bar{x}| \neq 0$, where $c, m \in \mathbb{N}$ are constants and $|\bar{x}| = \max\{|x_i| \mid i \leq n\}$.

Recall that a structure $(A; \sigma)$ of a finite signature σ is *p-computably presentable* if it is isomorphic to a polynomial-time computable (p-computable) structure $(B; \sigma)$ where $B \subseteq \Sigma^*$ is p-computable (for a finite alphabet Σ), as well as all the signature functions and predicates.

We illustrate the introduced notion for the structures \mathbb{R}_{alg} and \mathbb{C}_{alg} , using some standard notions and facts from [11, 12]. With any $\alpha \in R_{\text{alg}}$ we associate the unique pair (p_α, k) where $p_\alpha \in \mathbb{Q}[x]$ is the minimal (hence, irreducible) unitary polynomial of degree ≥ 1 with $p_\alpha(\alpha) = 0$, and k satisfies $\alpha = \alpha_k$ where $\alpha_1 < \dots < \alpha_m$ is the increasing sequence of all real roots of p_α .

The standard binary encoding $b : \mathbb{Q} \rightarrow \{0, 1, *\}^*$ induces an encoding $b : \mathbb{Q}[x] \rightarrow \{0, 1, *\}^*$, which associates with a polynomial $p(x) = a_n x^n + \dots + a_1 x + a_0$, $a_n \neq 0$ if $n \neq 0$, the code $b(a_n) * \dots * b(a_0)$. Now we associate with any $\alpha \in R_{\text{alg}}$ the word $b(p_\alpha) * b(k)$ where (p_α, k) is the pair from the previous paragraph, which yields an injection $b : R_{\text{alg}} \rightarrow \{0, 1, *\}^*$.

Let now $\mathbb{R}_1 = (R_1; \leq, +, \times, 0, 1)$, where $R_1 = b(R_{\text{alg}})$, be the isomorphic copy of \mathbb{R}_{alg} induced by b ; we call it the *order presentation of \mathbb{R}_{alg}* . The bijection $b : R_{\text{alg}} \rightarrow R_1$ and the Gauss representation $z = x + iy$ of complex numbers induce a bijection between $R_1 \times R_1$ and C_{alg} . By encoding again the elements of $R_1 \times R_1$ by words in a finite alphabet in a standard way, we obtain a bijection $b : C_{\text{alg}} \rightarrow C_1$ which induces an isomorphism $g : \mathbb{C}_{\text{alg}} \rightarrow \mathbb{C}_1 = (C_1; +, \times, 0, 1)$. Informally, \mathbb{C}_1 is the product $\mathbb{R}_1 \times \mathbb{R}_1$.

In all cases, let $L(x)$ denote the length $|b(x)|$, where the exact sense of $b(x)$ should be clear from the context. Sometimes we will identify x and $b(x)$, considering it as the standard encoding.

Theorem 1. *The structures \mathbb{R}_1 and \mathbb{C}_1 are p-computable, and the operations $-x$ and $1/x$ in these fields are also p-computable. As a corollary, \mathbb{R}_{alg} and \mathbb{C}_{alg} are p-computably presentable.*

Proof Sketch. We provide very short proof hints based on well known deep facts about symbolic computations. Another proof (not using the resultant calculus), which is suggested in our work, is based on results of Sects. 3 and 4, in particular, on Theorem 2. The proof for \mathbb{C}_1 follows rather easily from the proof for \mathbb{R}_1 , so we concentrate on the latter structure.

The p-computability of R_1 and of \leq follow from p-computability of finding the greatest common divisor and of the factorization problem in $\mathbb{Q}[x]$, the Scturm root separation algorithm, and Mahler’s estimate for the lower bound of the length of root separating intervals [13–15]. The p-computability of $+$, \times follows from the formulas concerning resultants in [14, 16]. Although those formulas refer to the interval presentation of \mathbb{R}_{alg} (recalled below), they can easily be adjusted to the order presentation using the abovementioned algorithms. \square

To describe some other popular presentations of \mathbb{R}_{alg} and \mathbb{C}_{alg} , we need an extension of the notion of a p-computable structure. Let $A \subseteq \Sigma^*$ and let $E \subseteq A^2$ be an equivalence relation on A . We call the quotient set of the form $\bar{A} = A/E = \{[a] \mid a \in A\}$ a *quotient-set in Σ^** . In this paper we abbreviate “quotient-set in Σ^* ”, to “*quotient-set*”. A quotient-set $\bar{A} = A/E$ is *p-computable* if A and E are p-computable. A function $f : \bar{A} \rightarrow \bar{B}$ between quotient-sets is *p-computable* if there is a p-computable function $f_0 : A \rightarrow B$ such that $f([x]_E) = [f_0(x)]_F$ for $x \in A$. Note that in this case $(x, y) \in E$ implies $(f_0(x), f_0(y)) \in F$. In a similar way one can define the notion of a p-computable function $f : (\bar{A})^n \rightarrow \bar{B}$ or $f : (\bar{A})^n \rightarrow \Sigma^*$. Any $A \subseteq \Sigma^*$ maybe identified with A/id_A , so the notion of a p-computable function between quotient-sets extends that of a p-computable function.

A structure $\bar{\mathfrak{A}}$ of a finite signature is called a *quotient-structure* if its universe is a quotient-set. A quotient-structure $\bar{\mathfrak{A}} = (\bar{A}; \dots)$ is *p-computable* if \bar{A}

is a p-computable quotient-set and all signature functions and predicates are p-computable on \bar{A} . Quotient-structures $\bar{\mathfrak{A}}$ and $\bar{\mathfrak{B}}$ are *p-computably isomorphic* if there is an isomorphism $f : \bar{\mathfrak{A}} \rightarrow \bar{\mathfrak{B}}$ such that both f and f^{-1} are p-computable functions. Identifying the set A with A/id_A , we can consider the usual structures as quotient-structures. Quotient-structures arise when we cannot define elements of an abstract structure by a unique “canonical” word from Σ^* .

Next we formulate a useful characterization of quotient-structures p-computably isomorphic to \mathbb{R}_1 . It involves some properties of root-finding. In the formulation we use a standard encoding $a_0 * \dots * a_n$ of finite sequences of words $a_i \in \Sigma^*$ (assuming that the symbol $*$ is not in Σ).

Proposition 1. *Suppose that $\bar{\mathfrak{A}} = (\bar{A}; \leq, +, \times, 0, 1)$ is a p-computable quotient-structure isomorphic to \mathbb{R}_{alg} . Let \mathbb{Q}_1 be the least subfield of \mathbb{R}_1 and \mathbb{Q}_A be the least subfield of $\bar{\mathfrak{A}}$, which correspond to the set of rational numbers. Then $\bar{\mathfrak{A}}$ is p-computably isomorphic to \mathbb{R}_1 iff*

- (a) \mathbb{Q}_A is p-computably isomorphic to \mathbb{Q}_1 ;
- (b) there is a p-computable function f that finds from a given $a \in A$ a word $f(a) = a_n * \dots * a_1 * a_0$, $[a_i] \in \mathbb{Q}_A$, such that $[a_n] \neq 0$ and $[a]$ is a root of $[a_n]x^n + \dots + [a_1]x + [a_0]$ in $\bar{\mathfrak{A}}$;
- (c) there is a p-computable function that finds from a given word $a_n * \dots * a_1 * a_0$, $[a_i] \in \mathbb{Q}_A$, $[a_n] \neq 0$, a word $b_1 * \dots * b_m$, $m \geq 0$, $b_j \in A$, such that $[b_1], \dots, [b_m]$ are exactly all the roots of $[a_n]x^n + \dots + [a_1]x + [a_0]$ in $\bar{\mathfrak{A}}$.

This proposition shows that \mathbb{R}_1 is a most natural p-computable presentation of \mathbb{R}_{alg} in some sense. In [10], it is proved that for every infinite p-computable structure $\bar{\mathfrak{A}}$, there exists an isomorphic p-computable structure $\bar{\mathfrak{A}}'$ such that they are not p-computably isomorphic. In particular, such $\bar{\mathfrak{A}}'$ exists for \mathbb{R}_1 .

Now we define other natural presentations of \mathbb{R}_{alg} and \mathbb{C}_{alg} known in the literature. For any polynomial $p \in \mathbb{Q}[x]$ of degree ≥ 1 without multiple roots, let $p'(x), p''(x), \dots, p^{(n-1)}(x)$ be the sequence of its derivative polynomials. For any $x \in \mathbb{R}$, let $\bar{\varepsilon}_p(x) = (\varepsilon_1(x), \dots, \varepsilon_{n-1}(x))$ where $\varepsilon_i(x) = 1, 0, -1$ iff $p^{(i)}(x)$ is resp. positive, zero, or negative. As follows from Thom’s Lemma [17], $\bar{\varepsilon}_p(\alpha) \neq \bar{\varepsilon}_p(\beta)$ whenever α and β are distinct roots of $p(x)$. Associate with any $\alpha \in R_{\text{alg}}$ the unique pair $(p_\alpha, \bar{\varepsilon}_{p_\alpha}(\alpha))$, and let R_2 be the set of codes of such pairs in a natural word encoding based on the above-mentioned encoding of rational polynomials and a natural encoding of sequences of 1, 0, -1. Let $\mathbb{R}_2 = (R_2; <, +, \times, 0, 1)$ be the isomorphic copy of \mathbb{R}_{alg} induced by the bijection $\alpha \mapsto (p_\alpha, \bar{\varepsilon}_{p_\alpha}(\alpha))$. The presentation \mathbb{R}_2 of \mathbb{R}_{alg} (which we call the *sign presentation*) was introduced in [17].

We can also code a real $\alpha \in \mathbb{R}_{\text{alg}}$ by a pair $(p(x), I)$, where $p(x) \in \mathbb{Q}[x] \setminus \{0\}$, $p(\alpha) = 0$, and $I = (a, b]$ is an isolating rational interval for α including α and no other roots of $p(x)$. Call two pairs *equivalent*, $(p_1(x), I_1) \sim (p_2(x), I_2)$, if they encode the same real. Let

$$A_3 = \{b(p(x)) * b(a) * b(b) \mid p(x) \in \mathbb{Q}[x], a, b \in \mathbb{Q} \text{ and } (p(x), I = (a, b]) \text{ encodes some } \alpha \in \mathbb{R}\},$$

let $E_3 \subseteq A_3 \times A_3$ be the relation corresponding to the equivalence of pairs, and let $R_3 = A_3/E_3$ be the corresponding quotient-set. Let $\mathbb{R}_3 = (R_3; <, +, \times, 0, 1)$ be the corresponding isomorphic copy of \mathbb{R}_{alg} . We call this presentation of \mathbb{R}_{alg} the *interval presentation* (see e.g. [16]).

A similar interval presentation of \mathbb{C}_{alg} is also known in the literature. We say that a triple (p, I, K) , where p is a polynomial and I, K are rational intervals as above, defines the number $z \in \mathbb{C}$ if z is the unique root of p in the rectangle $I+iK$. Let C be the set of codes of such triples (p, I, K) in a natural encoding, $\gamma : C \rightarrow \mathbb{C}_{\text{alg}}$ be the surjection defined similarly to the previous paragraph (of course, γ is not a bijection), and E be the corresponding equivalence relation on C . Then we have a presentation of \mathbb{C}_{alg} as a quotient-structure $\mathbb{C}_2 = (C/E; +, \times, 0, 1)$. The following fact follows from Proposition 1 using the results mentioned in the proof sketch of Theorem 1.

Corollary 1. *The quotient-structures $\mathbb{R}_2, \mathbb{R}_3$ are p -computably isomorphic to \mathbb{R}_1 and are therefore p -computable. The quotient-structure \mathbb{C}_2 is p -computably isomorphic to \mathbb{C}_1 and is therefore p -computable.*

3 Complexity of Polynomial Evaluation

Here we establish an optimal upper complexity bound for the problem of rational polynomial evaluation in \mathbb{R}_1 and \mathbb{C}_1 .

Let $<$ be the lexicographic order on the set ω^* of finite strings of natural numbers. If $I \subseteq \omega^*$, $I = \{i_0 < i_1 < \dots < i_n\}$, a_i for $i \in I$ are words in some alphabets and b is a new symbol, then we define $C_{i \in I}^b a_i = a_{i_0} b a_{i_1} b \dots b a_{i_n}$. Fix a sequence of variables x_1, x_2, \dots . Any polynomial $t(\bar{x})$ in $\mathbb{Q}[x_1, \dots, x_n]$ is representable in the form

$$t(\bar{x}) = \sum_{(s_1, \dots, s_k) \in I} q_{s_1, \dots, s_k} x_1^{s_1} x_2^{s_2} \dots x_k^{s_k},$$

where $I \subseteq \omega^*$, $I \neq \emptyset$, $s_k \neq 0$ for $k \neq 0$, $q_{s_1, \dots, s_k} \in \mathbb{Q}$, and $q_{s_1, \dots, s_k} \neq 0$ for $k \neq 0$. We set

$$b(t(\bar{x})) = 1^m * C_{(s_1, \dots, s_k) \in I}^+ b(q_{s_1, \dots, s_k}) * b(s_1) * \dots * b(s_k),$$

where $m = \max_{(s_1, \dots, s_k) \in I} \{s_i\}$. By $L(t(\bar{x}))$ we denote the length $|b(t(\bar{x}))|$. The unary word 1^m makes this encoding (for $k = 1$) p -equivalent to the above-mentioned encoding of polynomials in $\mathbb{Z}[x]$. Recall that the degree $\text{deg}(\alpha)$ of $\alpha \in C_{\text{alg}}$ is the degree of the minimal polynomial p_α for α .

Theorem 2. *There exists an algorithm which, given $k \geq 1$, $\alpha_1, \dots, \alpha_k \in C_{\text{alg}}$ and $t(x_1, \dots, x_k) \in \mathbb{Q}[x_1, \dots, x_k]$, finds $\beta = t(\alpha_1, \dots, \alpha_k) \in C_{\text{alg}}$. More precisely, the algorithm from given words $C_{i \leq k}^{\<} b(\alpha_i)$ and $b(t(x_1, \dots, x_k))$ finds the word $b(\beta)$.*

Let $n_i = \deg[\alpha_i]$ for each $i \leq k$, and let $n = \max_{i \leq k} \{n_i\}$. Then the working time of the algorithm is bounded by $(n_1 n_2 \cdots n_k)^c L^d$, or $n^{ck} L^d$, where c, d are some constants and L is the input length. In particular, for a fixed k we get a p -computable function that evaluates polynomials from $\mathbb{Q}[x_1, \dots, x_k]$. Also, $\deg[\beta] \leq \prod_{i \leq k} \deg[\alpha_i]$.

Proof Sketch. We briefly describe only basic ideas of the proof. The main step is an algorithm that, given $k \geq 1$, non-zero polynomials $p_1(x), \dots, p_k(x) \in \mathbb{Z}[x]$, and $t(x_1, \dots, x_k) \in \mathbb{Q}[x_1, \dots, x_k]$, constructs a non-zero polynomial $q(x) \in \mathbb{Z}[x]$ such that if $\alpha_1, \dots, \alpha_k \in \mathbb{C}$ and $p_i(\alpha_i) = 0$ for $i \leq k$, then $q(t(\alpha_1, \dots, \alpha_k)) = 0$. Its work time can also be estimated as $(n_1 n_2 \cdots n_k)^c L^d$, where $n_i = \deg[p_i(x)]$, L is the input length, and c, d are fixed constants.

The algorithm is based on bulky computations. Let b_i be the leading coefficient of $p_i(x)$, $i \leq k$. Given $s \geq 0$, we find numbers $c_t^{s,i} \in \mathbb{Z}$ such that

$$\alpha_i^s = \frac{1}{b_i^s} \sum_{t < n_i} c_t^{s,i} \alpha_i^t, \tag{1}$$

using pseudo-division of the polynomial x^s by $p_i(x)$.

Next, given $s_1, \dots, s_k \geq 0$, we find numbers $d_{t_1, \dots, t_k}^{s_1, \dots, s_k} \in \mathbb{Z}$ such that

$$\alpha_1^{s_1} \alpha_2^{s_2} \cdots \alpha_k^{s_k} = \frac{1}{b_1^{s_1} b_2^{s_2} \cdots b_k^{s_k}} \sum_{t_i < n_i} d_{t_1, \dots, t_k}^{s_1, \dots, s_k} \alpha_1^{t_1} \alpha_2^{t_2} \cdots \alpha_k^{t_k}, \tag{2}$$

where $s = \max_{i \leq k} \{s_i\}$. This means that every monomial $\alpha_1^{s_1} \cdots \alpha_k^{s_k}$ is expressed as a linear combination of a finite set of base monomials $\{\alpha_1^{t_1} \cdots \alpha_k^{t_k}\}_{t_i < n_i}$, which has $u = n_1 n_2 \cdots n_k$ elements.

Let $\beta = t(\alpha_1, \dots, \alpha_k)$. Having $t(\bar{x})$, we can find a linear combination of the form (2) for β . The same can be done for all β^s , $s \geq 0$, by induction on s . Therefore we obtain $u + 1$ linear combinations of length u for $\beta^0, \beta^1, \dots, \beta^u$, which must be linearly dependent. Applying Gauss algorithm for solving a corresponding system of linear equations in polynomial time [18], we find numbers $\lambda_0, \lambda_1, \dots, \lambda_u \in \mathbb{Q}$ such that $\lambda_0 + \lambda_1 \beta + \dots + \lambda_u \beta^u = 0$. We have constructed required $q(x)$.

The rest of the proof is as follows. The given numbers $\alpha_1, \dots, \alpha_k \in C_{\text{alg}}$ are defined by two polynomials for their real and imaginary parts. First, we pass to one polynomial $p_i(x) \in \mathbb{Z}[x]$ such that $p_i(\alpha_i) = 0$, $i \leq k$. Using the algorithm above, find $q(x)$ having $\beta = t(\alpha_1, \dots, \alpha_k)$ in its roots. Then, we pass to $q_1(x), q_2(x) \in \mathbb{Z}[x]$ whose real roots include the real and imaginary parts of roots of $q(x)$, respectively. If $\beta = \beta_1 + i\beta_2$ then $\beta_j \in \mathbb{R}$ is a root of $q_j(x)$.

Using approximations for $\alpha_1, \dots, \alpha_k$ and standard arithmetic calculations, we construct an approximation sequence for β . Separating roots of $q_j(x)$, we can find ones that correspond to β .

An essential part of the proof are exact estimations for the work time of all steps above, and the sizes of corresponding intermediate objects. Certainly, we cannot designate them here. The theorem is proved.

We note that the upper bound in Theorem 2 is in a sense close to optimal. This is a corollary of the following result in [19]: for all distinct primes p_1, \dots, p_k and all $n_1, \dots, n_k \geq 1$, the extension degree $[\mathbb{Q}(p_1^{1/n_1} + \dots + p_k^{1/n_k}) : \mathbb{Q}]$ is $n_1 n_2 \dots n_k$. If now p_1, \dots, p_k are the first k primes then $p_k \leq 2^k$. Take arbitrary $n_1, \dots, n_k \geq 1$ and let $\alpha_i = p_i^{1/n_i}$ for $i \leq k$. The number α_i is coded by $(x^{n_i} - p_i, m)$ where $m \in \{1, 2\}$, and $|\mathbb{L}(\alpha_i)| \leq 2n_i + k + 5$, $|\mathbb{b}(\alpha_1) \& \dots \& \mathbb{b}(\alpha_k)| \leq 2(n_1 + \dots + n_k) + k^2 + 6k$. If $t(x_1, \dots, x_k) = x_1 + \dots + x_k$ then $\mathbb{L}(t) \leq k^2 + 6k$. Also, $\mathbb{L}(\alpha_1 + \dots + \alpha_k) \geq n_1 n_2 \dots n_k$, because the length of a number cannot be less than the degree of its minimal polynomial. Thus, the algorithm of Theorem 2 cannot work in polynomial time uniformly on k even when evaluating the polynomials $x_1 + \dots + x_k$, and the upper bound from this theorem is bounded by a polynomial from L and the lower bound $n_1 n_2 \dots n_k$. Moreover, if we take $n_i = n$ for $i \leq k$ and $L^* = \mathbb{L}(t(\alpha_1, \dots, \alpha_k))$ then L^*/L^d for $k \rightarrow \infty$ is close to n^k , therefore the bound of Theorem 2 cannot be essentially improved.

Some algorithms for computing the function

$$\mathbb{b}(t(x_1, \dots, x_k)), \mathbb{b}(\alpha_1) * \dots * \mathbb{b}(\alpha_k) \mapsto t(\alpha_1, \dots, \alpha_k)$$

are well known in the literature. For $k = 2$ one can use resultants to get a polynomial time bound, while for $k > 2$ one can use a series of $k - 1$ of resultants. For computing the resultants good enough algorithms are known [20], but the straightforward application of these algorithms gives rather bad estimates.

At the same time, the standard algorithm for $k = 1$ is fast because the arithmetic in the simple algebraic extension $\mathbb{Q}(\alpha_1)$ is very easy. The standard algorithm for $k > 1$ is to reduce the problem to the case $k = 1$ using the primitive element theorem saying that for all $\alpha_1, \dots, \alpha_k \in \mathbb{C}_{\text{alg}}$, there is a “primitive element” $\theta \in \mathbb{C}_{\text{alg}}$ such that $\mathbb{Q}(\alpha_1, \dots, \alpha_k) = \mathbb{Q}(\theta)$. There is a p-time algorithm which for given $\alpha_1, \alpha_2 \in \mathbb{C}_{\text{alg}}$ finds $\theta \in \mathbb{C}_{\text{alg}}$ and polynomials $c_1(x), c_2(x) \in \mathbb{Q}[x]$ such that $\mathbb{Q}(\alpha_1, \alpha_2) = \mathbb{Q}(\theta)$ and $c_i(\theta) = \alpha_i$ for $i = 1, 2$. The algorithm is described in [16] and used in many other texts, see e.g. [21, §5.4]. This algorithm may be applied for $k > 2$ subsequently (in some order) to compute θ with $\mathbb{Q}(\theta) = \mathbb{Q}(\alpha_1, \dots, \alpha_k)$, see e.g. [22, §4.2] or [23]. This method is also used in [24] for describing a relatively fast algorithm of quantifier elimination for $\text{Th}(\mathbb{R})$.

In our situation, this method yields for a fixed k a p-time algorithm for computing $t(\alpha_1, \dots, \alpha_k)$ but the resulting estimate (based on the estimates in [16] for finding θ and $c_1, c_2(x)$) seems to be at least $n^{ck \log(k)} L^d$ where n is the maximal degree of α_i , $i \leq k$, L is the input length and c, d are constants. Our algorithms are based on other ideas (not using the primitive elements) and yield a better estimate for $k \rightarrow \infty$.

4 Complexity of Root Finding

Here we study the complexity of root-finding in \mathbb{C}_{alg} , i.e. of finding all roots of an equation $\alpha_e x^e + \dots + \alpha_1 x + \alpha_0 = 0$ where $\alpha_i \in \mathbb{C}_{\text{alg}}$ for $i \leq e$. Our estimates look better if we consider equations of the form

$$t_e(\alpha_1, \dots, \alpha_k) x^e + \dots + t_1(\alpha_1, \dots, \alpha_k) x + t_0(\alpha_1, \dots, \alpha_k) = 0, \quad (3)$$

where $\alpha_1, \dots, \alpha_k \in C_{\text{alg}}$ and $t_j(\bar{x}) \in \mathbb{Q}[x_1, \dots, x_k]$. The problem is to find a list of (codes of) all roots from given $b(\alpha_1) \& \dots \& b(\alpha_k)$ and $b(t_0(\bar{x})) \& \dots \& b(t_e(\bar{x}))$. The form (3) is convenient since our algorithm remains polynomial for fixed k even if e grows.

Again, there are well known p-time algorithms that solve the problem for $k = 1$, and for $k > 1$ the authors often recommend to go from $\alpha_1, \dots, \alpha_k$ to a primitive element θ [16]. In this way, we meet the same growth of complexity as above. We propose an alternative algorithm (not using the resultants and primitive elements at all) which again yields a better estimate $(n_1 \dots n_k)^c L^d$.

Theorem 3. *There exists an algorithm which, given $k \geq 1$, $\alpha_1, \dots, \alpha_k \in C_{\text{alg}}$ and polynomials $t_0(\bar{x}), \dots, t_e(\bar{x}) \in \mathbb{Q}[x_1, \dots, x_k]$, finds a list $\beta_1, \dots, \beta_g \in C_{\text{alg}}$ of all complex roots of the Eq. (3).*

More precisely, the algorithm sends words $C_{i \leq k}^{\&} b(\alpha_i)$ and $C_{p \leq e}^{\&} b(t_p(\bar{x}))$ to a word $C_{j \leq g}^{\&} b(\beta_j)$. Let $n_i = \deg[\alpha_i]$ for $i \leq k$, $n = \max_{i \leq k} \{n_i\}$, and let L be the input length. Then the working time of the algorithm is estimated as $(n_1 n_2 \dots n_k)^c L^d$, or $n^c L^d$, where c, d are constant. In particular, if k is fixed or $n = 1$, we get a p-time algorithm for root-finding. Furthermore, $\deg[\beta_j] \leq e \prod_{i \leq k} \deg[\alpha_i]$ for $j \leq g$.

Proof Sketch. Again we briefly describe only the basic ideas of the proof. It is close to the proof of Theorem 2 in many aspects. The main part of the proof is devoted to the case where $t_e(\bar{x}) = -1$. Then (3) has the form

$$x^e = t_{e-1}(\alpha_1, \dots, \alpha_k)x^{e-1} + \dots + t_1(\alpha_1, \dots, \alpha_k)x + t_0(\alpha_1, \dots, \alpha_k). \quad (4)$$

Let $\alpha_1, \dots, \alpha_k \in C_{\text{alg}}$ and let $\beta \in C_{\text{alg}}$ be a root of (4). Proceeding as in Theorem 2, we find minimal polynomials $p_i(x) \in \mathbb{Z}[x]$ such that $p_i(\alpha_i) = 0$. Let $n_i = \deg[p_i(x)]$. Then β^e can be expressed as a linear combination of a finite set of base monomials $\{\alpha_1^{t_1} \alpha_2^{t_2} \dots \alpha_k^{t_k} \beta^p\}_{t_i < n_i}^{p < e}$ by (2). The set has eu elements, where $u = n_1 n_2 \dots n_k$. Using induction on s , we can express every β^s , $s \geq e$, by such a combination:

$$\beta^s = \frac{1}{e_s} \sum_{t_i < n_i}^{p < e} e_{t_1, \dots, t_k, p}^s \alpha_1^{t_1} \dots \alpha_k^{t_k} \beta^p, \quad (5)$$

where $e_s, e_{t_1, \dots, t_k, p}^s \in \mathbb{Z}$.

As in Theorem 2, we therefore obtain $eu + 1$ linear combinations of length eu for $\beta^0, \beta^1, \dots, \beta^{eu}$. They are linear dependent. Solving a system of linear equations, we obtain numbers $\lambda_0, \lambda_1, \dots, \lambda_{eu} \in \mathbb{Q}$ such that $\lambda_0 + \lambda_1 \beta + \dots + \lambda_{eu} \beta^{eu} = 0$. Hence, we have constructed a polynomial $q(x) \in \mathbb{Z}[x]$ such that $q(\beta) = 0$.

Now all roots of (4) are among the roots of $q(x)$. We pass to $q_1(x), q_2(x) \in \mathbb{Z}[x]$ whose real roots include the real and imaginary parts of roots of $q(x)$, respectively. Then we can form a finite list of elements of C_{alg} which includes all its roots. Using the algorithm from Theorem 2, we substitute them to (4) and prove which of them are the required roots.

In the general case, where $t_e(\bar{x})$ is an arbitrary polynomial, we pass from k to $k + 1$, defining $\alpha_{k+1} = -1/t_e(\alpha_1, \dots, \alpha_k)$, and obtain the equation

$$x^e = \alpha_{k+1}t_{e-1}(\alpha_1, \dots, \alpha_k)x^{e-1} + \dots + \alpha_{k+1}t_0(\alpha_1, \dots, \alpha_k). \quad (6)$$

The proof is reduced to the previous case. If $t_e(\alpha_1, \dots, \alpha_k) = 0$ then we proceed by induction on e . All needed estimations are omitted here. The theorem is proved.

References

1. Ershov, Y.L., Goncharov, S.S.: Constructive Models. Plenum, New York (1999)
2. Stoltenberg-Hansen, V., Tucker, J.V.: Effective algebras. In: Handbook of Logic in Computer Science: Semantic Modelling, vol. IV, pp. 357–526. Oxford University Press, Oxford (1995)
3. Rabin, M.O.: Computable algebra, general theory and theory of computable fields. Trans. Am. Math. Soc. **95**, 341–360 (1960)
4. Ershov, Y.L.: Theorie der Nummerierungen III. Ziets. Math. Logik Grundl. Math. **23**, 289–371 (1977)
5. Nerode, A., Rummel, J.B.: Complexity theoretic algebra I, vector spaces over finite fields. In: Proceedings of Structure in Complexity, 2nd Annual Conference, pp. 218–239. Computer Sci. Press, New York (1987)
6. Cenzer, D., Rummel, J.B.: Complexity theoretic model theory and algebra. In: Handbook of Recursive Mathematics, vol. 1. Elsevier, New York City (1998)
7. Cenzer, D., Rummel, J.: Polynomial time versus recursive models. Ann. Pure Appl. Log. **54**, 17–58 (1991)
8. Basu, S., Pollack, R., Roy, M.: Algorithms in Real Algebraic Geometry. Springer, Heidelberg (2006). <https://doi.org/10.1007/3-540-33099-2>
9. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley Pub. Co., Boston (1974)
10. Alaev, P.E.: Structures computable in polynomial time I. Algebra Log. **55**(6), 421–435 (2016). <https://doi.org/10.1007/s10469-017-9416-y>
11. Van der Waerden, B.L.: Algebra. Springer, Berlin (1967)
12. Balcázar, J.L., Díaz, J., Gabarró, J., Structural, C.I.: Structural Complexity I. EATCS, vol. 11. Springer, Heidelberg (1988). <https://doi.org/10.1007/978-3-642-97062-7>
13. Akritas, A.G.: Elements of Computer Algebra with Applications. Wiley, New York (1989)
14. Collins, G.E., Loos, R.: Real zeros of polynomials. In: Buchberger, B., Collins, G.E., Loos, R., Albrecht, R. (eds.) Computer Algebra: Symbolic and Algebraic Computations. COMPUTING, vol. 4, pp. 83–94. Springer, Vienna (1982). https://doi.org/10.1007/978-3-7091-7551-4_7
15. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**, 515–534 (1982)
16. Loos, R.: Computing in algebraic extensions. In: Buchberger, B., Collins, G.E., Loos, R. (eds.) Computer Algebra: Symbolic and Algebraic Computations. COMPUTING, vol. 4, pp. 173–187. Springer, Vienna (1982). https://doi.org/10.1007/978-3-7091-3406-1_12

17. Coste, M., Roy, M.F.: Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *J. Symb. Comput.* **5**, 121–129 (1988)
18. Schrijver, A.: *Theory of Linear and Integer Programming*. Wiley, New York (1986)
19. Jian-Ping, Z.: On the degree of extensions generated by finitely many algebraic numbers. *J. Number Theory* **34**, 133–141 (1990)
20. Collins, G.E.: The Calculation of multivariate polynomial resultants. *J. Assoc. Comput. Mach.* **18**(4), 515–532 (1971)
21. Winkler, F.: *Polynomial Algorithms in Computer Algebra*. Springer, Wien (1996). <https://doi.org/10.1007/978-3-7091-6571-3>
22. Cohen, H.: *A Course in Computational Algebraic Number Theory*. Springer, Heidelberg (1996). <https://doi.org/10.1007/978-3-662-02945-9>
23. Yap, C.K.: *Fundamental Problems in Algorithmic Algebra*. Oxford University Press, Oxford (2000)
24. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Brakhage, H. (ed.) *GI-Fachtagung 1975*. LNCS, vol. 33, pp. 134–183. Springer, Heidelberg (1975). https://doi.org/10.1007/3-540-07407-4_17



Multiple Permitting and Array Noncomputability

Klaus Ambos-Spies^(✉)

Institut für Informatik, Universität Heidelberg, 69120 Heidelberg, Germany
ambos@math.uni-heidelberg.de

Abstract. Downey et al. [5] have introduced the array noncomputable (a.n.c.) computably enumerable (c.e.) sets which capture certain multiple permitting arguments. In this way they have classified the c.e. degrees below which certain constructions can be performed. More recently, Downey et al. [3] have introduced the not totally ω -c.a. c.e. degrees allowing some stronger permitting arguments. Here we introduce a formal notion of multiple permitting which captures the permitting strength of the a.n.c. sets. This notion which – in contrast to array noncomputability – is wtt-invariant allows to simplify the proofs of the basic properties of the a.n.c. sets. Moreover, some results on the a.n.c. sets can be naturally extended to the multiply permitting sets hence to the c.e. sets which are wtt-equivalent to a.n.c. sets. We demonstrate this by showing that multiply permitting sets are not dense simple. Finally, in Ambos-Spies and Losert (ta) the multiply permitting notion introduced here is refined in order to get a new formal characterization of the permitting power of the not totally ω -c.a. c.e. degrees too.

1 Introduction

Downey et al. [5] studied a class of permitting arguments where each positive requirement needs multiple permitting to succeed. They introduced the array noncomputable (a.n.c.) computably enumerable (c.e.) sets which “consist of precisely those sets which allow enough permission for these constructions to be performed”. By classifying the degrees of the a.n.c. sets – in particular by showing that first all non-low₂ c.e. degrees are a.n.c., second the a.n.c. degrees are closed upwards in the c.e. degrees and third the a.n.c. degrees nontrivially split both the nonzero low degrees and the c.e. degrees which are low₂ but not low – Downey, Jockusch and Stob describe the degrees in which the multiple permitting constructions can be performed.

Actually, the a.n.c. sets are characterized by a certain similarity property which is exploited in showing that any a.n.c. set A provides the required permitting. Namely, in order to make A change on the first x numbers after a stage s , a number $\leq x$ is enumerated into an auxiliary set B and it is argued that (under certain circumstances) the fact that A locally looks like B forces a number $\leq x$ to enter A after stage s . So the a.n.c. sets provide multiple permittings but they

characterize the sets giving such permittings only up to wtt-equivalence. Here we introduce multiply permitting sets which more directly describe and provide the desired sort of permittings. The multiply permitting sets coincide with the a.n.c. sets up to wtt-equivalence but, in contrast to the latter, the multiply permitting sets are wtt-invariant. Moreover the permittings can be achieved in a more straightforward way which allows to simplify proofs of some results dealing with a.n.c. sets, and, by replacing a.n.c. sets by the newly defined multiply permitting sets, in some cases such results can be extended from the a.n.c. sets to all c.e. sets which are wtt-equivalent to some a.n.c. sets. In this note we demonstrate this by giving some examples.

After introducing the new multiple permitting notion in Sect. 2 we first give some basic properties of the multiply permitting sets (Sect. 3) and of their wtt-degrees (Sect. 4). A comparison of the proofs given here with the corresponding proofs on a.n.c. sets (in cases where parallel results hold) in [5] shows the simplifications provided by our new notion. Then, in Sect. 5, we show the coincidence of the multiple permitting notion and array noncomputability up to wtt-equivalence. For an example of a result which can be extended from the a.n.c. sets to the multiply permitting sets, in Sect. 6 we look at simplicity properties and, by extending a result of Downey et al. [5] on a.n.c. sets, we show that no multiply permitting set is dense simple. Finally, we note that our multiple permitting notion can be extended in order to capture the stronger multiple permitting arguments pertaining to the non-totally- ω -c.a. c.e. degrees introduced in Downey et al. [3], thereby giving a new characterization of the permitting power of these degrees (see Sect. 7).

Our notation is standard. For unexplained notation we refer to the monographs Downey and Hirschfeldt [4] and Soare [9].

2 Multiple Permitting Notions

We first review the array noncomputability notions. Just as the new multiply permitting notions to be introduced here, these notions are based on very strong arrays (see [5] respectively [4] for the variation used here). A *very strong array* (v.s.a.) $\mathcal{F} = \{F_n\}_{n \geq 0}$ is a sequence of mutually disjoint finite sets such that there is a computable function f such that $f(n)$ is the canonical index of F_n and such that $0 < |F_n| < |F_{n+1}|$ ($n \geq 0$). A v.s.a. $\mathcal{F} = \{F_n\}_{n \geq 0}$ is *complete* if $\bigcup_{n \geq 0} F_n = \omega$ and \mathcal{F} is a *very strong array of intervals* (v.s.a.i.) if the members F_n of \mathcal{F} are intervals where $\max F_n < \min F_{n+1}$. Then, given a v.s.a. $\mathcal{F} = \{F_n\}_{n \geq 0}$, a c.e. set A is *\mathcal{F} -array noncomputable* if, for any c.e. set B , there are infinitely many numbers n such that $A \cap F_n = B \cap F_n$, and A is *array noncomputable* (a.n.c.) if A is \mathcal{F} -array noncomputable for some v.s.a. \mathcal{F} . Finally, a c.e. degree is *a.n.c.* if it contains an a.n.c. set.

Definition 1. Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be a v.s.a., let f be a computable function, let A be a c.e. set, and let $\{A_s\}_{s \geq 0}$ be a computable enumeration of A . Then A is \mathcal{F} -permitting via f and $\{A_s\}_{s \geq 0}$ if, for any partial computable function ψ ,

$$\exists^\infty n \forall x \in F_n(\psi(x) \downarrow \Rightarrow A \upharpoonright f(x) + 1 \neq A_{\psi(x)} \upharpoonright f(x) + 1) \quad (1)$$

holds. A is \mathcal{F} -permitting via f if there is a computable enumeration $\{A_s\}_{s \geq 0}$ of A such that A is \mathcal{F} -permitting via f and $\{A_s\}_{s \geq 0}$; A is \mathcal{F} -permitting if A is \mathcal{F} -permitting via some computable f ; and A is multiply permitting if A is \mathcal{F} -permitting for some v.s.a. \mathcal{F} . A c.e. r -degree \mathbf{a} is multiply permitting if there is a c.e. set A in \mathbf{a} such that A is multiply permitting.

Note that, for a given c.e. set A which is \mathcal{F} -permitting via (a nondecreasing function) f , for infinitely many n we can force A to change below $f(\max F_n) + 1$ after up to $|F_n|$ effectively enumerated stages $s_0^n < \dots < s_{m_n}^n$ ($m_n < |F_n|$) by letting $\psi(x_p^n) = s_p^n$ where $x_0^n < x_1^n < \dots$ are the elements of F_n in order.

3 Some Basic Observations

In the following we list some simple but useful facts on the multiply permitting sets which we will tacitly use later. We first observe that the multiple permitting properties do not depend on the chosen enumerations, that the permitting bounds f may be assumed to be strictly increasing, and that the quantifier $\exists^\infty n$ in (1) can be replaced by $\exists n$.

Proposition 1. *Let A be \mathcal{F} -permitting via f and $\{A_s\}_{s \geq 0}$ and let $\{\hat{A}_s\}_{s \geq 0}$ be a computable enumeration of A . Then A is \mathcal{F} -permitting via f and $\{\hat{A}_s\}_{s \geq 0}$.*

Proof. Given a partial computable function $\hat{\psi}$, we have to show

$$\exists^\infty n \forall x \in F_n(\hat{\psi}(x) \downarrow \Rightarrow A \upharpoonright f(x) + 1 \neq \hat{A}_{\hat{\psi}(x)} \upharpoonright f(x) + 1). \quad (2)$$

Define the partial computable function ψ by letting $\psi(x)$ be the least stage $s > \hat{\psi}(x)$ such that $\hat{A}_{\hat{\psi}(x)} \upharpoonright f(x) + 1 \subseteq A_s \upharpoonright f(x) + 1$ if $\hat{\psi}(x)$ is defined and by letting $\psi(x) \uparrow$ otherwise. Then $\hat{\psi}$ and ψ have the same domain and $\hat{A}_{\hat{\psi}(x)} \upharpoonright f(x) + 1 \subseteq A_{\psi(x)} \upharpoonright f(x) + 1$. Since, by choice of A , (1) holds, this implies (2). \square

Proposition 2. *Let A be \mathcal{F} -permitting via f and let \hat{f} be any computable function dominating f . Then A is \mathcal{F} -permitting via \hat{f} . Hence, in particular, for any \mathcal{F} -permitting set A there is a strictly increasing computable function \hat{f} such that A is \mathcal{F} -permitting via \hat{f} .*

Proof. This is immediate by definition and by the fact that any computable function is dominated by a strictly increasing computable function. \square

Proposition 3. *Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be a v.s.a., let f be a computable function, let A be a c.e. set and let $\{A_s\}_{s \geq 0}$ be a computable enumeration of A such that, for any partial computable function ψ ,*

$$\exists n \forall x \in F_n(\psi(x) \downarrow \Rightarrow A \upharpoonright f(x) + 1 \neq A_{\psi(x)} \upharpoonright f(x) + 1) \quad (3)$$

holds. Then A is \mathcal{F} -permitting via f (and $\{A_s\}_{s \geq 0}$).

We omit the straightforward proof but remark that a corresponding observation on a.n.c. sets has been made in [5]. The next proposition shows that if we have to give a set A which is \mathcal{F} -permitting via f then w.l.o.g. we may assume that \mathcal{F} is a complete v.s.a.i. hence, in particular, computable. For this sake we use the following definition. We say that the v.s.a. $\hat{\mathcal{F}} = \{\hat{F}_n\}_{n \geq 0}$ dominates the v.s.a. $\mathcal{F} = \{F_n\}_{n \geq 0}$ if for almost every number n there is a number m such that $F_m \subseteq \hat{F}_n$.

Proposition 4. *Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ and $\hat{\mathcal{F}} = \{\hat{F}_n\}_{n \geq 0}$ be very strong arrays such that $\hat{\mathcal{F}} = \{\hat{F}_n\}_{n \geq 0}$ dominates $\mathcal{F} = \{F_n\}_{n \geq 0}$, and let f be any computable function. Then any c.e. set A which is $\hat{\mathcal{F}}$ -permitting via f is \mathcal{F} -permitting via f . Hence, in particular, for any v.s.a. $\mathcal{F} = \{F_n\}_{n \geq 0}$ there is a complete v.s.a.i. $\hat{\mathcal{F}} = \{\hat{F}_n\}_{n \geq 0}$ such that any c.e. set A which is $\hat{\mathcal{F}}$ -permitting via f is \mathcal{F} -permitting via f .*

Proof. The first part of the proposition is immediate. For the second part it suffices to observe that any v.s.a. \mathcal{F} is dominated by a complete v.s.a.i. $\hat{\mathcal{F}}$. For instance we may let $\hat{F}_n = [x_n, x_{n+1})$ where $x_0 = 0$ and $x_{n+1} = 1 + x_n + \max F_m$ for the least m such that $x_n \leq \min F_m$. \square

Finally, we observe that the multiple permitting property does not depend on the underlying v.s.a. as long as we do not require the same permitting bound f .

Lemma 1 (Array-Invariance Lemma). *Let A be multiply permitting and let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be any v.s.a. Then A is \mathcal{F} -permitting.*

Proof. By Proposition 4, w.l.o.g. we may assume that \mathcal{F} is a complete v.s.a.i. So, for any number x , we can compute the unique number n_x such that $x \in F_{n_x}$. By the assumption that A is multiply permitting, fix a v.s.a. $\hat{\mathcal{F}} = \{\hat{F}_n\}_{n \geq 0}$ and a computable function \hat{f} such that A is $\hat{\mathcal{F}}$ -permitting via \hat{f} where, by Proposition 2, we may assume that \hat{f} is strictly increasing. Finally, fix a strictly increasing computable function g such that, for $n \geq 0$, $|F_n| \leq |\hat{F}_{g(n)}|$, and define the computable function f by letting $f(x) = \hat{f}(\max \bigcup_{m < g(n_x+1)} \hat{F}_m)$. We claim that A is \mathcal{F} -permitting via f .

Fix a partial computable function ψ . By Proposition 3, it suffices to show (3). Define the partial computable function $\hat{\psi}$ by letting $\hat{\psi}(\hat{x}_k^m) = \psi(x_k^n)$ for $g(n) \leq m < g(n+1)$ and $k < |F_n|$ where $\hat{x}_0^m, \dots, \hat{x}_{|\hat{F}_m|-1}^m$ are the elements of \hat{F}_m in increasing order and $x_0^n, \dots, x_{|F_n|-1}^n$ are the elements of F_n in increasing order and by letting $\hat{\psi}(x) \uparrow$ otherwise. (Note that, by definition of g , $|\hat{F}_m| \geq |F_n|$. Hence $\hat{\psi}$ is well defined.)

Now, since A is $\hat{\mathcal{F}}$ -permitting via \hat{f} , there is a number $m \geq g(0)$ such that

$$\forall \hat{x} \in \hat{F}_m (\hat{\psi}(\hat{x}) \downarrow \Rightarrow A_{\hat{\psi}(\hat{x})} \upharpoonright \hat{f}(x) \neq A \upharpoonright f(x)).$$

But, by definition of $\hat{\psi}$ and f this implies that the unique n such that $g(n) \leq m < g(n+1)$ witnesses that (3) holds. \square

4 On the Wtt-Degrees of the Multiply Permitting Sets

Here we first show that the property of being multiply permitting is wtt-degree invariant. In fact, any c.e. set which wtt-computes a multiply permitting set is multiply permitting too. Then we show that no multiply permitting c.e. set can be split into two c.e. sets which are not multiply permitting. We conclude that the class of the c.e. wtt-degrees which are not multiply permitting is an ideal in the partial ordering of the c.e. wtt-degrees.

Lemma 2 (Wtt-Invariance Lemma). *Let A and B be c.e. sets such that A is multiply permitting and $A \leq_{\text{wtt}} B$. Then B is multiply permitting. Hence, in particular, any c.e. set which is wtt-equivalent to a multiply permitting set is multiply permitting.*

Lemma 2 is immediate by the following lemma.

Lemma 3. *Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be a v.s.a., let f and g be strictly increasing computable functions and let A and B be c.e. sets such that A is \mathcal{F} -permitting via f and $A \leq_{g\text{-T}} B$. Then B is \mathcal{F} -permitting via $g(f)$.*

Proof. Given a computable enumeration $\{B_s\}_{s \geq 0}$ of B and a partial computable function ψ , by Proposition 3, it suffices to show that there is a number n such that

$$\forall x \in F_n (\psi(x) \downarrow \Rightarrow B \upharpoonright g(f(x)) + 1 \neq B_{\psi(x)} \upharpoonright g(f(x)) + 1). \quad (4)$$

Fix a computable enumeration $\{A_s\}_{s \geq 0}$ of A such that A is \mathcal{F} -permitting via f and $\{A_s\}_{s \geq 0}$, fix a g -bounded Turing functional Φ such that $A = \Phi^B$, and define the partial computable function $\hat{\psi}$ by letting $\hat{\psi}(x)$ be the least stage $s \geq \psi(x)$ such that $A_s \upharpoonright f(x) + 1 = \Phi_s^B \upharpoonright f(x) + 1$ if $\psi(x)$ is defined and let $\hat{\psi}(x) \uparrow$ otherwise. Note that the domains of ψ and $\hat{\psi}$ agree; $\hat{\psi}$ majorizes ψ where defined; and, for any x such that $\psi(x) \downarrow$ (hence $\hat{\psi}(x) \downarrow$) and $A \upharpoonright f(x) + 1 \neq A_{\hat{\psi}(x)} \upharpoonright f(x) + 1$, it follows that $B \upharpoonright g(f(x)) + 1 \neq B_{\hat{\psi}(x)} \upharpoonright g(f(x)) + 1$ hence $B \upharpoonright g(f(x)) + 1 \neq B_{\psi(x)} \upharpoonright g(f(x)) + 1$. So we obtain a number n satisfying (4), by fixing n such that $A \upharpoonright f(x) + 1 \neq A_{\hat{\psi}(x)} \upharpoonright f(x) + 1$ for all $x \in F_n$ such that $\hat{\psi}(x) \downarrow$. Note that such an n must exist since A is \mathcal{F} -permitting via f and $\{A_s\}_{s \geq 0}$. \square

Lemma 4 (Splitting Lemma). *Let A, A_0, A_1 be c.e. sets such that A_0 and A_1 are disjoint, $A = A_0 \cup A_1$ and A is multiply permitting. Then A_0 or A_1 is multiply permitting.*

Proof. Fix computable enumerations $\{A_s\}_{s \geq 0}$ and $\{A_{i,s}\}_{s \geq 0}$ of A and A_i , respectively, such that $A_s = A_{0,s} \cup A_{1,s}$. Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be a complete v.s.a.i., let $\mathcal{F}_i = \{F_{2n+i}\}_{n \geq 0}$ ($i = 0, 1$), and let $\hat{\mathcal{F}} = \{F_{2n} \cup F_{2n+1}\}_{n \geq 0}$. By Lemma 1 and Proposition 2 fix a strictly increasing function \hat{f} such that A is $\hat{\mathcal{F}}$ -permitting via \hat{f} , and define the computable function f by letting $f(x) = \hat{f}(\max F_{2n} \cup F_{2n+1})$ for $x \in F_{2n} \cup F_{2n+1}$. We claim that, for some $i \leq 1$, A_i is \mathcal{F}_i -permitting via f .

For a contradiction assume that neither A_0 is \mathcal{F}_0 -permitting via f nor A_1 is \mathcal{F}_1 -permitting via f . Then there are partial computable functions ψ_i such that, for any $n \geq 0$ and $i \leq 1$, there is a number $x_{n,i} \in F_{2n+i}$ such that $\psi_i(x_{n,i}) \downarrow$ and

$$A_{i,\psi_i(x_{n,i})} \upharpoonright f(x_{n,i}) + 1 = A_i \upharpoonright f(x_{n,i}) + 1. \quad (5)$$

Fix $i_n \leq 1$ minimal such that $\psi_{i_n}(x_{n,i_n}) \geq \psi_{1-i_n}(x_{n,1-i_n})$. Then, by definition of f , $\hat{f}(x_{n,i_n}) \leq f(x_{n,0}) = f(x_{n,1})$. Since $A_s = A_{0,s} \cup A_{1,s}$, it follows by (5) that

$$A_{\psi_{i_n}(x_{n,i_n})} \upharpoonright \hat{f}(x_{n,i_n}) + 1 = A \upharpoonright \hat{f}(x_{n,i_n}) + 1.$$

So if we define the partial computable function ψ by letting $\psi(x) = \psi_i(x)$ for $x \in \bigcup_{n \geq 0} F_{2n+i}$ ($i \leq 1$) then, for any n and for $x = x_{n,i_n}$, $x \in F_{2n} \cup F_{2n+1}$, $\psi(x) = \psi_{i_n}(x_{n,i_n}) \downarrow$ and

$$A_{\psi(x)} \upharpoonright \hat{f}(x) + 1 = A \upharpoonright \hat{f}(x) + 1.$$

But this contradicts the assumption that A is $\hat{\mathcal{F}}$ -permitting via \hat{f} . \square

Theorem 1. *The class $\overline{\mathbf{MP}}_{\text{wtt}}$ of the c.e. wtt-degrees which do not contain multiply permitting c.e. sets is an ideal in the upper semilattice $(\mathbf{R}_{\text{wtt}}, \leq)$ of the c.e. wtt-degrees.*

Proof. By Lemma 2, the class $\overline{\mathbf{MP}}_{\text{wtt}}$ is closed downwards. So, given degrees \mathbf{a}_0 and \mathbf{a}_1 in $\overline{\mathbf{MP}}_{\text{wtt}}$, it suffices to show that, for $\mathbf{a} = \mathbf{a}_0 \vee \mathbf{a}_1$, \mathbf{a} is in $\overline{\mathbf{MP}}_{\text{wtt}}$. Fix disjoint c.e. sets A_0 and A_1 such that $A_0 \in \mathbf{a}_0$ and $A_1 \in \mathbf{a}_1$, and let $A = A_0 \cup A_1$. Then $A \in \mathbf{a}$. Moreover, since A_0 and A_1 are not multiply permitting, A is not multiply permitting by Lemma 4. It follows, by Lemma 2, that no c.e. set in \mathbf{a} is multiply permitting. Hence \mathbf{a} is in $\overline{\mathbf{MP}}_{\text{wtt}}$. \square

The reader should note that, in contrast to Lemma 3, Downey et al. [5] have shown that array noncomputability is not wtt-invariant. Downey et al. [5] proved the analog of Lemma 4 for a.n.c. sets, however, and they showed that any c.e. set wtt-above an a.n.c. set is wtt-equivalent to an a.n.c. set. Since we show next that the wtt-degrees of the multiply permitting sets coincide with the wtt-degrees of the a.n.c. sets, this gives an alternative proof of Theorem 1.

5 Multiple Permitting vs. Array Noncomputability

We now show that a set A is multiply permitting iff it is wtt-equivalent to an a.n.c. set. For this sake we first show that \mathcal{F} -a.n.c. sets are \mathcal{F} -permitting via the identity function. Then we show that, for any v.s.a. \mathcal{F} and for any multiply permitting set A there is an \mathcal{F} -a.n.c. set which is wtt-equivalent to A .

Lemma 5 (Permitting Lemma for A.N.C. Sets). *Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be a v.s.a., let A be an \mathcal{F} -a.n.c. set, let $\{A_s\}_{s \geq 0}$ be a computable enumeration of A , and let ψ be a partial computable function. Then*

$$\exists^\infty n \forall x \in F_n (\psi(x) \downarrow \Rightarrow x \in A \setminus A_{\psi(x)}) \quad (6)$$

holds. Hence, in particular, A is \mathcal{F} -permitting via $f(x) = x$.

Proof. Let $V = \{x : \psi(x) \downarrow \ \& \ x \notin A_{\psi(x)}\}$. Then V is c.e. So, by \mathcal{F} -array noncomputability of A , there are infinitely many n such that $A \cap F_n = V \cap F_n$. But any such n is a witness for (6). \square

Lemma 6. *Let A be multiply permitting and let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be a v.s.a. There is an \mathcal{F} -a.n.c. set B such that $B =_{\text{wtt}} A$.*

Proof. Since the analog of Proposition 4 holds for the array noncomputable sets too, w.l.o.g. we may assume that \mathcal{F} is computable. For any $e \geq 0$, let $\mathcal{F}_e = \{F_{\langle e+1, n \rangle}\}_{n \geq 0}$. Then \mathcal{F}_e is a v.s.a. Moreover, there is a computable function f such that A is \mathcal{F}_e -permitting via f for all e . Note that the latter follows from Lemma 1 and Proposition 4 since the v.s.a. $\hat{\mathcal{F}} = \{\hat{F}_n\}_{n \geq 0}$ where $\hat{F}_n = \bigcup_{\langle n, n \rangle \leq m < \langle n+1, n+1 \rangle} F_m$ dominates the very strong arrays \mathcal{F}_e for all e .

Now define B as follows. For $x \in F_{\langle 0, n \rangle}$ let $B(x) = A(n)$. This ensures that $A =_{\text{wtt}} B \cap \bigcup_{n \geq 0} F_{\langle 0, n \rangle}$. So, in order to guarantee that $B =_{\text{wtt}} A$, it suffices to define B on $\bigcup_{e \geq 0, n \geq 0} F_{\langle e+1, n \rangle}$ so that

$$B \cap \bigcup_{e \geq 0, n \geq 0} F_{\langle e+1, n \rangle} \leq_{\text{wtt}} A. \quad (7)$$

On the other hand, in order to make B \mathcal{F} -a.n.c., it suffices to ensure

$$\exists^\infty n (B \cap F_{\langle e+1, n \rangle} = W_e \cap F_{\langle e+1, n \rangle}) \quad (8)$$

for $e \geq 0$. We achieve (7) and (8) by letting

$$B \cap F_{\langle e+1, n \rangle} = \{x \in F_{\langle e+1, n \rangle} : \exists s (x \in W_{e,s} \ \& \ A_{s+1} \upharpoonright f(x)+1 \neq A_s \upharpoonright f(x)+1)\}$$

for $e, n \geq 0$. It is obvious that this ensures (7). Satisfaction of (8) follows from the fact that A is \mathcal{F}_e -permitting via f by considering the partial computable function $\psi(x) = \mu s (x \in W_{e,s})$. \square

Theorem 2. *Let \mathbf{a} be a c.e. wtt-degree, let \mathcal{F} be a very strong array and let f be a strictly increasing computable function. The following are equivalent.*

- (i) \mathbf{a} is array noncomputable.
- (ii) \mathbf{a} is multiply permitting.
- (iii) There is an \mathcal{F} -a.n.c. set $A \in \mathbf{a}$.
- (iv) There is a c.e. set $A \in \mathbf{a}$ such that A is \mathcal{F} -permitting via f .

Note that the equivalence (i) \Leftrightarrow (iii) is already in [5].

Proof. The implications (iii) \Rightarrow (i) and (iv) \Rightarrow (ii) are immediate by definition, and the implication (i) \Rightarrow (ii) follows from the implication (iii) \Rightarrow (iv). So it suffices to prove the implications (ii) \Rightarrow (iii) and (iii) \Rightarrow (iv). But the former implication holds by Lemma 6 while the latter implication holds by Lemma 5 and Proposition 2. \square

Corollary 1 (Downey et al. [5]). *For any a.n.c. set A and any v.s.a. \mathcal{F} there is an \mathcal{F} -a.n.c. set which is wtt-equivalent to A .*

Proof. This is immediate by the implication (i) \Rightarrow (iii) in Theorem 2. \square

Corollary 2 (Downey et al. [5]). *The class $\overline{\text{ANC}}_{\text{wtt}}$ of the c.e. wtt-degrees which do not contain array noncomputable c.e. sets is an ideal in the upper semilattice $(\mathbf{R}_{\text{wtt}}, \leq)$ of the c.e. wtt-degrees.*

Proof. This is immediate by the equivalence (i) \Leftrightarrow (ii) in Theorem 2 and by Theorem 1. \square

Downey et al. [5] have shown that there are c.e. sets A_0 and A_1 such that $\text{deg}_{\text{T}}(A_0)$ and $\text{deg}_{\text{T}}(A_1)$ are not a.n.c. and $\text{deg}_{\text{T}}(A_0 \oplus A_1) = \mathbf{0}'$ hence $\text{deg}_{\text{T}}(A_0 \oplus A_1)$ is a.n.c. ([5], Theorem 3.15). So the analog of Corollary 2 for Turing degrees fails. Moreover, by Corollary 2, for sets A_0 and A_1 as above, the wtt-degree of the Turing complete set $A_0 \oplus A_1$ is not a.n.c. So, in particular, there is a c.e. set A such that $\text{deg}_{\text{T}}(A)$ is a.n.c. whereas $\text{deg}_{\text{wtt}}(A)$ is not a.n.c. By Theorem 2, these results imply the following corresponding results on multiply permitting sets: the Wtt-Invariance Lemma (Lemma 2) fails if we replace wtt-reducibility by Turing reducibility and so does Theorem 1. Moreover, the complete Turing degree $\mathbf{0}'$ contains c.e. sets with the multiple permitting property and c.e. sets without the multiple permitting property. Ambos-Spies and Monath [2] have extended the latter result by showing that any high c.e. Turing degree has this property. (In Sect. 6 below we obtain an alternative proof of this fact.) They have also shown, however, that there is a completely multiply permitting Turing degree, i.e., a c.e. Turing degree \mathbf{a} such that all c.e. sets $A \in \mathbf{a}$ are multiply permitting (or, equivalently, all c.e. wtt-degrees inside \mathbf{a} are a.n.c.).

6 Array Noncomputability and Simplicity Properties

Downey et al. [5] have shown that a.n.c. sets are not dense simple hence neither hyperhypersimple nor maximal ([5], Theorem 3.1). Here we extend this result on a.n.c. sets to multiply permitting sets (hence to a.n.c. wtt-degrees).

Theorem 3. *No multiply permitting set is dense simple.*

Recall that a c.e. set A is *dense simple* if the complement $\overline{A} = \{\bar{a}_0 < \bar{a}_1 < \dots\}$ of A is infinite and the principal function $p_{\overline{A}}(n) = \bar{a}_n$ of \overline{A} dominates every computable function. Just as in [5] we use the following equivalent characterization of dense simplicity due to Robinson [8]: a coinfinite c.e. set A is dense simple if and only if, for every strong array $\{F_n\}_{n \geq 0}$ of mutually disjoint sets, there is a number m such that

$$\forall n \geq m (|F_n \cap \overline{A}| < n). \quad (9)$$

Proof. For a contradiction assume that A is multiply permitting and dense simple. Let $\mathcal{I} = \{I_n\}_{n \geq 0}$ be the unique complete v.s.a.i. such that $|I_n| = 1 + \sum_{n' \leq n} n'$, let $x_0^n < \dots < x_{k_n}^n$ ($k_n = 0 + 1 + \dots + n$) be the elements of I_n in increasing order, and, by the Array-Invariance Lemma, fix a strictly increasing computable function f such that A is \mathcal{I} -permitting via f .

Define the strong array $\{F_n\}_{n \geq 0}$ of mutually disjoint sets by letting $F_0 = [0, f(x_{k_0}^0)]$ and $F_{n+1} = (f(x_{k_n}^n), f(x_{k_{n+1}}^{n+1})]$. By dense simplicity of A fix m such that (9) holds, fix a computable enumeration $\{A_s\}_{s \geq 0}$ of A , and fix s_0 such that $A_{s_0} \cap F_n = A \cap F_n$ for all $n < m$. Then the function $s(n)$ defined by

$$s(n) = \mu s \geq s_0 [\forall n' (m \leq n' \leq n \Rightarrow |F_{n'} \cap A_s| > |F_{n'}| - n')]$$

is partial computable and $s(n) \downarrow$ for $n \geq m$. Finally, define the partial computable function ψ on $\bigcup_{n \geq m} I_n$ by letting $\psi(x_0^n) = s(n)$ and, for $k < k_n$,

$$\psi(x_{k+1}^n) = \mu s > \psi(x_k^n) [A_s \upharpoonright f(x_{k_n}^n) + 1 \neq A_{\psi(x_k^n)} \upharpoonright f(x_{k_n}^n) + 1]$$

provided that $\psi(x_k^n) \downarrow$ (and by letting $\psi(x_{k+1}^n) \uparrow$ otherwise).

Now, since A is \mathcal{I} -permitting via f , we may fix $n \geq m$ such that

$$\forall x \in I_n (\psi(x) \downarrow \Rightarrow A \upharpoonright f(x) + 1 \neq A_{\psi(x)} \upharpoonright f(x) + 1).$$

It follows by definition of ψ that $\psi(x) \downarrow$ for all $x \in I_n$ and that, for $k < k_n$, $\psi(x_k^n) < \psi(x_{k+1}^n)$ and $A_{\psi(x_{k+1}^n)} \upharpoonright f(x_{k_n}^n) + 1 \neq A_{\psi(x_k^n)} \upharpoonright f(x_{k_n}^n) + 1$ whence

$$|(A \upharpoonright f(x_{k_n}^n) + 1) \setminus (A_{s(n)} \upharpoonright f(x_{k_n}^n) + 1)| \geq k_n = 0 + 1 + \dots + n. \quad (10)$$

On the other hand, by definition of the strong array $\{F_n\}_{n \geq 0}$, $\omega \upharpoonright f(x_{k_n}^n) + 1$ is the disjoint union of the sets $F_{n'}$ for $n' \leq n$. So, by definition of $s(n)$,

$$|(A \upharpoonright f(x_{k_n}^n) + 1) \setminus (A_{s(n)} \upharpoonright f(x_{k_n}^n) + 1)| < m + (m + 1) + \dots + n$$

contrary to (10). □

Note that, by the Wtt-Invariance Lemma and by Theorem 2, Theorem 3 can be rephrased as follows.

Corollary 3. *Let A be a.n.c. Then A is not wtt-reducible to any dense simple set.*

Moreover, since any non-low₂ c.e. degree hence any high c.e. degree contains an a.n.c. set (Downey et al. [5]) and since any high c.e. degree contains a maximal set hence a dense simple set (Martin [6]), it follows by Theorems 3 and 2 that any high c.e. degree contains a c.e. set which is multiply permitting and a c.e. set which is not multiply permitting (or, equivalently, a c.e. wtt-degree which is a.n.c. and a c.e. wtt-degree which is not a.n.c.). As mentioned at the end of Sect. 5 already, the latter was previously shown by Ambos-Spies and Monath [2] using some other arguments.

Finally we should mention, that Monath [7] further explores the question which of the most common simplicity notions are compatible with the multiply permitting property. By showing that there is an r -maximal multiply permitting set (and by Theorem 3 above) he obtains a complete answer for the simplicity notions considered in Soare ([9], Diagram 1.1, page 211). Monath's result contrasts a result of Downey et al. [5] showing that no a.n.c. set is r -maximal (in fact no a.n.c. set is strongly hypersimple).

7 Further Results

By the Array-Invariance Lemma (Lemma 1) any multiply permitting set A is \mathcal{F} -permitting for all very strong arrays \mathcal{F} . The permitting bound $f_{\mathcal{F}}$ such that A is \mathcal{F} -permitting via $f_{\mathcal{F}}$, however, in general depends on the array \mathcal{F} . It is natural to ask whether this bound can be fixed, i.e., whether there are *uniformly* multiply permitting sets which are \mathcal{F} -permitting via f for all very strong arrays \mathcal{F} and fixed f . In [1] Ambos-Spies and Losert show that uniformly multiply permitting sets exist and that their Turing degrees are just the c.e. not totally ω -c.a. degrees. This gives a new way to formally describe the permitting properties of these degrees. In [1] this is exploited in order to give a finite lattice such that the c.e. degrees which bound embeddings of this lattice in the c.e. Turing degrees are just the not totally ω -c.a. degrees.

References

1. Ambos-Spies, K., Losert, N.: Universally array noncomputable sets. Submitted
2. Ambos-Spies, K., Monath, M.: Completely array noncomputable degrees. In Preparation
3. Downey, R., Greenberg, N., Weber, R.: Totally ω -computably enumerable degrees and bounding critical triples. *J. Math. Log.* **7**, 145–171 (2007)
4. Downey, R.G., Hirschfeldt, D.R.: *Algorithmic Randomness and Complexity. Theory and Applications of Computability.* Springer, New York (2010). <https://doi.org/10.1007/978-0-387-68441-3>
5. Downey, R., Jockusch, C., Stob, M.: Array nonrecursive sets and multiple permitting arguments. In: Ambos-Spies, K., Müller, G.H., Sacks, G.E. (eds.) *Recursion Theory Week. LNM*, vol. 1432, pp. 141–173. Springer, Heidelberg (1990). <https://doi.org/10.1007/BFb0086116>
6. Martin, D.A.: Classes of recursively enumerable sets and degrees of unsolvability. *Z. Math. Log. Grundle. Math.* **12**, 295–310 (1966)
7. Monath, M.: A c.e. weak truth table degree which is array noncomputable and r -maximal. In: Manea, F., Miller, R.G., Nowotka, D. (eds.) *CiE 2018. LNCS*, vol. 10936, pp. 297–306. Springer, Cham (2018)
8. Robinson, R.W.: Simplicity of recursively enumerable sets. *J. Symb. Log.* **32**, 162–172 (1967)
9. Soare, R.I.: *Recursively Enumerable sets and Degrees. A Study of Computable Functions and Computably Generated Sets. Perspectives in Mathematical Logic.* Springer, Berlin (1987). xviii+437 pp



Degrees of Categoricity for Prime and Homogeneous Models

Nikolay Bazhenov^{1,2}  and Margarita Marchuk^{1,2}

¹ Sobolev Institute of Mathematics, Novosibirsk, Russia
bazhenov@math.nsc.ru, margaretmarchuk@gmail.com

² Novosibirsk State University, Novosibirsk, Russia

Abstract. We study effective categoricity for homogeneous and prime models of a complete theory. For a computable structure \mathcal{S} , the degree of categoricity of \mathcal{S} is the least Turing degree which can compute isomorphisms among arbitrary computable copies of \mathcal{S} . We build new examples of degrees of categoricity for homogeneous models and for prime Heyting algebras, i.e. prime models of a complete extension of the theory of Heyting algebras. We show that $\mathbf{0}^{(\omega+1)}$ is the degree of categoricity for a homogeneous model. We prove that any Turing degree which is d.c.e. in and above $\mathbf{0}^{(n)}$, where $3 \leq n < \omega$, is the degree of categoricity for a prime Heyting algebra.

Keywords: Computable categoricity · Categoricity spectrum
Degree of categoricity · Computable structure · Autostability spectrum
Homogeneous model · Prime model · Heyting algebra

1 Introduction

In this paper we study algorithmic complexity of isomorphisms between computable copies for prime and homogeneous models of a complete theory.

Let \mathbf{d} be a Turing degree. A computable structure \mathcal{M} is \mathbf{d} -*computably categorical* if for any computable copy \mathcal{N} of \mathcal{M} , there is a \mathbf{d} -computable isomorphism $f: \mathcal{M} \rightarrow \mathcal{N}$. The *categoricity spectrum* of \mathcal{M} is the set

$$CatSpec(\mathcal{M}) = \{\mathbf{d} : \mathcal{M} \text{ is } \mathbf{d}\text{-computably categorical}\}.$$

A degree \mathbf{c} is the *degree of categoricity* of \mathcal{M} if \mathbf{c} is the least degree in the spectrum $CatSpec(\mathcal{M})$.

The research of \mathbf{d} -computably categorical structures goes back to the works of Fröhlich and Shepherdson [15], and Mal'tsev [18, 19]. Categoricity spectra and degrees of categoricity were introduced by Fokina et al. [14]. They showed [14] that for a natural number n , any Turing degree \mathbf{d} which is d.c.e. in and above $\mathbf{0}^{(n)}$

N. Bazhenov was supported by RFBR project No. 16-31-60058 mol_a_dk.

M. Marchuk was supported by RFBR project No. 17-01-00247.

is the degree of categoricity for a computable structure. Csima et al. [11] proved that a similar result holds for any degree d.c.e. in and above $\mathbf{0}^{(\alpha+1)}$, where α is an infinite computable ordinal. Anderson and Csima [1] constructed a Σ_2^0 set whose Turing degree cannot be a degree of categoricity. Miller [20] built the first example of a structure with no degree of categoricity. For further results on categoricity spectra, the reader is referred to, e.g., [5, 13].

Our motivation for investigating categoricity spectra of prime models is twofold. First, much work has been done of late in studying categoricity spectra of structures in familiar classes: algebraic fields [20, 21], linear orders [6, 16], rigid structures [9, 12], etc. Following this line of research, we want to investigate \mathbf{d} -computable categoricity for structures with some natural model-theoretic properties.

Our second source of motivation has an origin in the following results. Suppose that \mathbf{d} is a degree which is d.c.e. in and above $\mathbf{0}^{(n)}$, and \mathcal{M} is the structure with degree of categoricity \mathbf{d} , built in [14]. A careful analysis of the construction reveals that \mathcal{M} is a prime model of its first-order theory. On the other hand, the authors [8] proved the following: Suppose that T is a complete extension of the theory of Boolean algebras. If \mathcal{B} is a prime model of T , then \mathcal{B} has degree of categoricity $\mathbf{d} \in \{\mathbf{0}^{(\alpha)} : \alpha \leq \omega\}$. Therefore, it is natural to state the following problem.

Problem. What degrees can be degrees of categoricity for prime models? In particular, if \mathcal{M} is a prime model which belongs to a familiar algebraic class, then what can one say about the categoricity spectrum of \mathcal{M} ?

Furthermore, it is natural to consider a more general class of structures, the class of homogeneous models of a complete theory. This can be illustrated by the following fact. It is not difficult to prove that for any computable homogeneous model \mathcal{M} , there is an “upper bound” on its categoricity spectrum: \mathcal{M} is always $\mathbf{0}^{(\omega+1)}$ -computably categorical (see Sect. 3 for details).

In the paper we study degrees of categoricity for prime and homogeneous models of a complete theory. The outline of the paper is as follows. Section 2 contains the necessary preliminaries. In Sect. 3 we build a computable homogeneous model (of a complete theory) with degree of categoricity $\mathbf{0}^{(\omega+1)}$. This shows that the upper bound given above is sharp. In Sect. 4, for $n \geq 3$ and a Turing degree \mathbf{d} d.c.e. in and above $\mathbf{0}^{(n)}$, we construct a prime model \mathcal{M} of a complete extension of the theory of Heyting algebras such that \mathbf{d} is the degree of categoricity for \mathcal{M} . We leave open whether this can be extended for $n \leq 2$. Note that this theorem contrasts with the result of [8] mentioned above: a similar result cannot be obtained for Boolean algebras. Section 5 contains further discussion.

2 Preliminaries

We consider only computable languages. For any considered countable structure, its domain is contained in ω .

For a structure \mathcal{M} , $Th(\mathcal{M})$ denotes the first-order theory of \mathcal{M} . Suppose that \mathcal{M} and \mathcal{N} are L -structures, $\bar{a} \in \mathcal{M}$ and $\bar{b} \in \mathcal{N}$ are tuples of the same finite length. Then the formula $(\mathcal{M}, \bar{a}) \equiv (\mathcal{N}, \bar{b})$ means that the structures (\mathcal{M}, \bar{a}) and (\mathcal{N}, \bar{b}) are elementarily equivalent.

2.1 Prime and Homogeneous Models

Suppose that T is a complete first-order theory in a language L . An L -structure \mathcal{M} is a *prime model* of T if $\mathcal{M} \models T$ and for any structure $\mathcal{N} \models T$, there is an elementary embedding from \mathcal{M} into \mathcal{N} . An L -structure \mathcal{M} is an *atomic model* if for any tuple $\bar{a} = a_0, \dots, a_n$ from \mathcal{M} , there is an L -formula $\varphi(x_0, \dots, x_n)$ such that $\mathcal{M} \models \varphi(\bar{a})$, and every L -formula $\psi(x_0, \dots, x_n)$ satisfies the following: if $\mathcal{M} \models \psi(\bar{a})$, then $\mathcal{M} \models \forall \bar{x}[\varphi(\bar{x}) \rightarrow \psi(\bar{x})]$. Such a formula $\varphi(\bar{x})$ is called a *complete formula* of the theory $Th(\mathcal{M})$.

Vaught (see [10]) proved that an L -structure \mathcal{M} is a prime model of a theory T if and only if $\mathcal{M} \models T$ and \mathcal{M} is a countable atomic model.

An L -structure \mathcal{M} is a *homogeneous model* of T if $\mathcal{M} \models T$ and for any tuples \bar{a} and \bar{b} of the same finite length and any element c from \mathcal{M} , the following holds:

$$(\mathcal{M}, \bar{a}) \equiv (\mathcal{M}, \bar{b}) \Rightarrow \exists d[(\mathcal{M}, \bar{a}, c) \equiv (\mathcal{M}, \bar{b}, d)].$$

Note that a prime model of T is homogeneous.

2.2 Pairs of Computable Structures

Our proofs heavily use the technique of pairs of computable structures developed by Ash and Knight [3, 4]. Here we give necessary preliminaries on the technique.

For a language L , *infinitary formulas* of L are formulas of the logic $L_{\omega_1, \omega}$. For a countable ordinal α , infinitary Σ_α and Π_α formulas are defined in a standard way (see, e.g., [4, Chap. 6]). Suppose that \mathcal{A} is an L -structure and K is a family of L -structures. We say that $K \leq_\alpha \mathcal{A}$ if every infinitary Π_α sentence true in all structures from K is also true in \mathcal{A} . We write $\mathcal{B} \leq_\alpha \mathcal{A}$ if $\{\mathcal{B}\} \leq_\alpha \mathcal{A}$.

Let α be a computable ordinal. A family $K = \{\mathcal{A}_i : i \in I\}$ of L -structures is α -friendly if the structures \mathcal{A}_i are uniformly computable in $i \in I$, and the relations

$$B_\beta := \{(i, \bar{a}, j, \bar{b}) : i, j \in I, \bar{a} \in \mathcal{A}_i, \bar{b} \in \mathcal{A}_j, (\mathcal{A}_i, \bar{a}) \leq_\beta (\mathcal{A}_j, \bar{b})\}$$

are computably enumerable, uniformly in $\beta < \alpha$.

Theorem 2.1 ([3, Theorem 3.1]). *Suppose that α is a non-zero computable ordinal, \mathcal{A} and \mathcal{B} are L -structures. If $\mathcal{B} \leq_\alpha \mathcal{A}$ and the family $\{\mathcal{A}, \mathcal{B}\}$ is α -friendly, then for any Π_α^0 set X , there is a uniformly computable sequence of L -structures $\{\mathcal{C}_n\}_{n \in \omega}$ such that*

$$\mathcal{C}_n \cong \begin{cases} \mathcal{A}, & \text{if } n \in X; \\ \mathcal{B}, & \text{if } n \notin X. \end{cases}$$

Theorem 2.2 ([3, Theorem 4.2]). *Suppose that α is a non-zero computable ordinal, \mathcal{A} is an L -structure, $K = \{\mathcal{B}_i : i \in \omega\}$ is a family of L -structures. If $K \leq_\alpha \mathcal{A}$ and the family $\{\mathcal{A}\} \cup K$ is α -friendly, then for any Π_α^0 set X , there is a uniformly computable sequence of L -structures $\{\mathcal{C}_n\}_{n \in \omega}$ such that*

$$\mathcal{C}_n \cong \begin{cases} \mathcal{A}, & \text{if } n \in X; \\ \text{a structure from } K, & \text{if } n \notin X. \end{cases}$$

2.3 Ordinals

We treat ordinals as structures in the language $\{\leq\}$. First, we introduce some auxiliary first-order formulas. Using the proof of [4, Proposition 7.2] or [17, Sect. 2], for an ordinal $\alpha < \omega^\omega$, one can define finitary formulas $\lambda_{\alpha+1}(x)$ and $\mu_{\alpha+1}(x, y)$ such that for any partial order \mathcal{P} , the following holds:

- $\mathcal{P} \models \lambda_{\alpha+1}(a)$ if and only if the set $\mathcal{P}[a] := \{b \in \mathcal{P} : b \text{ is comparable with } a\}$ is linearly ordered and isomorphic to the ordinal $\alpha + 1$.
- $\mathcal{P} \models \mu_{\alpha+1}(a, b)$ if and only if $a \leq b$, the closed interval $[a; b]_{\mathcal{P}} := \{c \in \mathcal{P} : a \leq c \leq b\}$ is linearly ordered, and $[a; b]_{\mathcal{P}}$ is isomorphic to $\alpha + 1$.

Ash [2] obtained a complete description of the relations \leq_α for ordinals. Here we give only an excerpt from the description.

Lemma 2.1 ([2], see also [3, p. 224] and [4, Sect. 15.3.3]). *For a countable ordinal β , we have $\omega^\beta \cdot 2 \leq_{2\beta+1} \omega^\beta$ and $\omega^{\beta+1} + \omega^\beta \leq_{2\beta+2} \omega^{\beta+1}$. Moreover, $\{\gamma : \gamma < \omega^\beta\} \leq_{2\beta} \omega^\beta$.*

It is well-known that ordinals behave nicely with respect to α -friendliness:

Lemma 2.2 ([4, Proposition 15.11]). *Suppose that $\alpha, \beta_0, \beta_1, \dots, \beta_n, \gamma$ are computable ordinals, and γ is infinite. There exist α -friendly families $\{\mathcal{R}_0, \mathcal{R}_1, \dots, \mathcal{R}_n\}$ and $\{\mathcal{S}_j : j \in \omega\}$ with the following properties:*

- every \mathcal{R}_i is isomorphic to β_i ;
- for any non-zero ordinal $\xi < \gamma$, there is a structure \mathcal{S}_j isomorphic to ξ ;
- every \mathcal{S}_j is isomorphic to some $\xi_j < \gamma$.

2.4 Heyting Algebras

We treat Heyting algebras as structures in the language $L_{HA} := \{\vee^2, \wedge^2, \rightarrow^2; 0, 1\}$. An L_{HA} -structure \mathcal{H} is a *Heyting algebra* if the $\{\vee, \wedge\}$ -reduct of \mathcal{H} is a distributive lattice, 0 is the least element in \mathcal{H} , 1 is the greatest element, and \mathcal{H} satisfies the following three axioms:

- (a) $\forall x \forall y [x \wedge (x \rightarrow y) = x \wedge y]$;
- (b) $\forall x \forall y \forall z [x \wedge (y \rightarrow z) = x \wedge ((x \wedge y) \rightarrow (x \wedge z))]$;
- (c) $\forall x \forall y \forall z [z \wedge ((x \wedge y) \rightarrow x) = z]$.

A partial order \leq in \mathcal{H} is recovered in a standard lattice-theoretical way: $x \leq y$ if and only if $x \vee y = y$.

An element y is a *complement* of an element x if $x \vee y = 1$ and $x \wedge y = 0$. Any element x from a Heyting algebra has at most one complement.

Suppose that $\{\mathcal{A}_n\}_{n \in \omega}$ is a sequence of Heyting algebras. The *direct sum* of the sequence $\{\mathcal{A}_n\}_{n \in \omega}$ (we denote it by $\sum_{n \in \omega} \mathcal{A}_n$) is the subalgebra of the product $\prod_{n \in \omega} \mathcal{A}_n$ on the domain

$$\left\{ f \in \prod_{n \in \omega} \mathcal{A}_n : (\exists c \in \{0, 1\}) \exists m (\forall k \geq m) (f(k) = c^{\mathcal{A}_k}) \right\}.$$

It is not hard to show that $\sum_{n \in \omega} \mathcal{A}_n$ is a Heyting algebra. Furthermore, if a sequence $\{\mathcal{A}_n\}_{n \in \omega}$ is computable, then one can build a computable presentation of the sum $\sum_{n \in \omega} \mathcal{A}_n$, in a standard way (see, e.g., [7, sect. 2.1] for details). Hence, in this case, we will identify the direct sum with its standard computable copy.

If $a_i \in \mathcal{A}_i$, $i \leq n$, and $a_n \neq 0^{\mathcal{A}_n}$, then $(a_0, a_1, \dots, a_n, \perp_{n+1})$ denotes the element $(a_0, a_1, \dots, a_n, 0, 0, \dots)$ from $\sum_{n \in \omega} \mathcal{A}_n$. If $a_n \neq 1^{\mathcal{A}_n}$, then by $(a_0, a_1, \dots, a_n, \top_{n+1})$ we denote the element $(a_0, a_1, \dots, a_n, 1, 1, \dots)$.

If \mathcal{L} is a linear order with the least element 0 and the greatest element 1, then \mathcal{L} can be treated as a Heyting algebra $\mathcal{H}(\mathcal{L})$ with the following operations: $x \vee y := \max(x, y)$, $x \wedge y := \min(x, y)$, and

$$x \rightarrow y := \begin{cases} 1, & \text{if } x \leq y; \\ y, & \text{if } x > y. \end{cases}$$

3 Homogeneous Models

In this section, we show that for any computable homogeneous model \mathcal{M} , $\mathbf{0}^{(\omega+1)}$ belongs to the categoricity spectrum of \mathcal{M} . Furthermore, we construct a homogeneous model with degree of categoricity $\mathbf{0}^{(\omega+1)}$.

Proposition 3.1. *Let \mathcal{M} be a computable homogeneous model of a complete theory T .*

- (i) \mathcal{M} is $\mathbf{0}^{(\omega+1)}$ -computably categorical.
- (ii) If T is arithmetical (i.e. $T \leq_T \mathbf{0}^{(n)}$ for some $n < \omega$) and \mathcal{M} is a prime model of T , then \mathcal{M} is $\mathbf{0}^{(\omega)}$ -computably categorical.

For reasons of space, the proof of Proposition 3.1 is omitted.

Theorem 3.1. *There exists a computable homogeneous model \mathcal{M} with degree of categoricity $\mathbf{0}^{(\omega+1)}$.*

Proof. Let \mathcal{M} be a partial order which is arranged as follows. For every ordinal $\alpha \leq \omega^\omega$, \mathcal{M} contains infinitely many copies of $\alpha + 1$. It is not difficult to build a standard computable presentation \mathcal{M}^{st} of \mathcal{M} with the following property: A number x belongs to a copy of $\omega^\omega + 1$ inside \mathcal{M}^{st} if and only if x is even.

For an ordinal $\beta < \omega^\omega$, we define the formula

$$\xi_{\beta+1}(x) := \exists c[\forall y(y \leq x \rightarrow c \leq y) \& \mu_{\beta+1}(c, x)].$$

It is straightforward to show that elements a and b from \mathcal{M} are automorphic if and only if they satisfy one of the following three cases:

1. There are ordinals $\beta \leq \alpha < \omega^\omega$ such that $\mathcal{M} \models \lambda_{\alpha+1}(a) \& \lambda_{\alpha+1}(b) \& \xi_{\beta+1}(a) \& \xi_{\beta+1}(b)$ (this means that each $x \in \{a, b\}$ lies in a copy of $\alpha + 1$, and the closed interval between the least element (in this copy) and x is isomorphic to $\beta + 1$).
2. There is an ordinal $\beta < \omega^\omega$ such that $\xi_{\beta+1}(a) \& \xi_{\beta+1}(b)$ holds, but for any $\alpha < \omega^\omega$, we have $\neg \lambda_\alpha(a) \& \neg \lambda_\alpha(b)$ (in particular, this implies that each of a and b lies in a copy of $\omega^\omega + 1$, and it is not a maximal element in the copy).
3. For any $\alpha, \beta < \omega^\omega$, we have $\neg \lambda_{\alpha+1}(a) \& \neg \lambda_{\alpha+1}(b) \& \neg \xi_{\beta+1}(a) \& \neg \xi_{\beta+1}(b)$ (this means that each of a and b is a maximal element in a copy of $\omega^\omega + 1$).

Using this observation, it is not hard to prove that for any tuples \bar{a} and \bar{b} , the condition $(\mathcal{M}, \bar{a}) \equiv (\mathcal{M}, \bar{b})$ implies that $(\mathcal{M}, \bar{a}) \cong (\mathcal{M}, \bar{b})$. Hence, a standard argument shows that \mathcal{M} is a homogeneous model of $Th(\mathcal{M})$.

Fix an ω -friendly family $K = \{\mathcal{S}_j : j \in \omega\}$ such that K contains a copy of every non-zero ordinal $\beta < \omega^\omega$ and nothing else. An analysis of the proof of Lemma 2.2 shows that one can build a computable copy \mathcal{R} of ω^ω such that the family $\{\mathcal{R}\} \cup K$ is also ω -friendly. By Lemma 2.1, we have $K \leq_\omega \mathcal{R}$. Hence, one can apply Theorem 2.2 and produce a computable sequence of linear orders $\{\mathcal{C}_n\}_{n \in \omega}$ such that for any n ,

$$\mathcal{C}_n \cong \begin{cases} \omega^\omega, & \text{if } n \notin \emptyset^{(\omega+1)}; \\ \text{an ordinal } < \omega^\omega, & \text{if } n \in \emptyset^{(\omega+1)}. \end{cases}$$

Consider a computable copy \mathcal{N} of \mathcal{M} which is built as follows. The partial order \mathcal{N} is a disjoint union of \mathcal{M}^{st} and a computable partial order \mathcal{N}_0 which is arranged as a computable disjoint union of the sequence $\{\mathcal{C}_n + 1\}_{n \in \omega}$. Suppose that c_n is the greatest element in the order $(\mathcal{C}_n + 1)$. Let f be an isomorphism from \mathcal{N} onto \mathcal{M}^{st} . Then the following conditions are equivalent: $n \notin \emptyset^{(\omega+1)}$ iff $(c_n$ lies in a copy of $\omega^\omega + 1$) iff $(f(c_n)$ is an even number). Therefore, $f \geq_T \mathbf{0}^{(\omega+1)}$. This fact and Proposition 3.1 together imply that $\mathbf{0}^{(\omega+1)}$ is the degree of categoricity for \mathcal{M} . \square

4 Prime Heyting Algebras

A *prime Heyting algebra* is a prime model of a complete extension of the theory of Heyting algebras. In this section, we build new examples of degrees of categoricity for prime Heyting algebras.

Theorem 4.1. *Suppose that $n \geq 3$ is a natural number, and \mathbf{d} is a Turing degree which is d.c.e. in and above $\mathbf{0}^{(n)}$. Then there exists a computable prime Heyting algebra with degree of categoricity \mathbf{d} .*

Proof. Here we give a proof for the case when n is even. The case of odd n is similar, and it will be discussed at the end of this section. We emphasize that all omitted technical details can be recovered from the proof of [7, Theorem 4].

A relativization of the argument from [14, Theorem 3.1] shows that one can choose a set $D \in \mathbf{d}$ such that D is d.c.e. in $\mathbf{0}^{(n)}$ and for any $X \subseteq \omega$, we have: If \overline{D} is c.e. in X , then $D \leq_T X \oplus \mathbf{0}^{(n)}$.

Suppose that $n = 2m$, where $m \geq 2$. We fix Σ_{2m+1}^0 sets A and B such that $B \subset A$ and $D = A \setminus B$.

Using Lemma 2.2, we obtain a $(2m + 1)$ -friendly family $\{\mathcal{S}_1, \mathcal{S}_2\}$ such that $\mathcal{S}_1 \cong \omega^m$ and $\mathcal{S}_2 \cong \omega^m \cdot 2$. Since $\omega^m \cdot 2 \leq_{2m+1} \omega^m$, we apply Theorem 2.1 and produce three computable sequences $\{\mathcal{C}_e\}_{e \in \omega}$, $\{\mathcal{D}_e\}_{e \in \omega}$, and $\{\mathcal{E}_e\}_{e \in \omega}$ such that for any $e \in \omega$,

$$\begin{aligned} \mathcal{C}_e &\cong \begin{cases} \omega^m, & \text{if } e \notin A, \\ \omega^m \cdot 2, & \text{if } e \in A; \end{cases} & \mathcal{D}_e &\cong \begin{cases} \omega^m, & \text{if } e \notin B, \\ \omega^m \cdot 2, & \text{if } e \in B; \end{cases} \\ \mathcal{E}_e &\cong \begin{cases} \omega^m, & \text{if } e \notin \emptyset^{(2m)}, \\ \omega^m \cdot 2, & \text{if } e \in \emptyset^{(2m)}. \end{cases} \end{aligned}$$

For every e , we define a computable linear order \mathcal{L}_e :

$$\begin{aligned} \mathcal{L}_{4k} &:= \mathcal{C}_k + 1 + \eta + \mathcal{C}_k + (2k + 1), \\ \mathcal{L}_{4k+1} &:= \mathcal{D}_k + 1 + \eta + \omega^m \cdot 2 + (2k + 1), \\ \mathcal{L}_{4k+2} &:= \mathcal{E}_k + 1 + \eta + \mathcal{E}_k + (2k + 2), \\ \mathcal{L}_{4k+3} &:= \omega^m + 1 + \eta + \omega^m \cdot 2 + (2k + 2). \end{aligned}$$

It is easy to effectively construct the sequence $\{\mathcal{L}_e\}_{e \in \omega}$ in a uniform way.

Consider a computable Heyting algebra $\mathcal{M} := \sum_{e \in \omega} \mathcal{H}(\mathcal{L}_e)$. We prove that \mathcal{M} satisfies our theorem.

For $k \in \omega$, let $e_k := (0, 0, \dots, 0, c_k, \perp_{k+1})$, where c_k is the greatest element in the algebra $\mathcal{H}(\mathcal{L}_k)$. We define auxiliary finitary formulas

$$\begin{aligned} Lin(x) &:= \forall y \forall z [(y \leq x) \& (z \leq x) \rightarrow (y \leq z) \vee (z \leq y)], \\ MaxLin(x) &:= Lin(x) \& \forall y [(x \leq y) \& Lin(y) \rightarrow (y = x)]. \end{aligned}$$

The $\forall \exists$ -formula $MaxLin(x)$ says that an element x is maximal such that the interval $[0; x]_{\mathcal{M}}$ is linearly ordered. It is not hard to establish the following algebraic fact (see [7, Lemma 3] for a similar argument).

Claim. (1) The set $MaxLin(\mathcal{M})$ contains precisely the elements e_k , $k \in \omega$.

(2) If a is an element such that $a \notin \{0^{\mathcal{M}}, 1^{\mathcal{M}}\}$, then it satisfies exactly one of the following two conditions:

(2a) There are elements p_0, p_1, \dots, p_k such that $a = p_0 \vee p_1 \vee \dots \vee p_k$, and for any $i \leq k$, we have $0 < p_i \leq e_{l_i}$, where $l_i \neq l_j$ for $i \neq j$.

(2b) There are elements q_0, q_1, \dots, q_k such that for any $i \leq k$, we have $0 \leq q_i < e_{l_i}$, where $l_i \neq l_j$ for $i \neq j$. These elements have the following property: the union $(e_{l_0} \vee e_{l_1} \vee \dots \vee e_{l_k})$ has a complement r in \mathcal{M} , and $a = (q_0 \vee q_1 \vee \dots \vee q_k) \vee r$.

Lemma 4.1. *The structure \mathcal{M} is a prime Heyting algebra.*

Proof (sketch). It is sufficient to show that \mathcal{M} is an atomic model. First, we demonstrate how to obtain a complete formula $\psi_a(x)$ for an element a from \mathcal{M} . We will use the following remark (its proof is omitted for reasons of space).

Remark 4.1. For any $k \in \omega$, the order \mathcal{L}_k is an atomic model.

Suppose that $0 < a \leq e_k$ for some k . We find ordinals $\alpha, \beta < \omega^{m+1}$ such that the order \mathcal{L}_k is isomorphic to $(\alpha + 1) + \eta + (\beta + 1)$. Let $\xi_a(x)$ be a complete formula of the theory $Th(\mathcal{L}_k)$ such that $\xi_a(a)$ holds in $[0; e_k]_{\mathcal{M}}$. We set

$$\begin{aligned} \psi_a^0(x, e) &:= \text{MaxLin}(e) \& \exists y \exists z [\mu_{\alpha+1}(0, y) \& (\text{the interval } [y; z] \text{ is dense}) \& \\ &\quad \mu_{\beta+1}(z, e)] \& (x \leq e) \& \xi_a(x), \\ \psi_a(x) &:= \exists e \psi_a^0(x, e). \end{aligned}$$

If $a \not\leq e_k$ for any k , then either $a = 1$, or a satisfies one of the conditions (2a) or (2b) from the claim above. For example, if a satisfies (2b), then one can set

$$\begin{aligned} \psi_a(x) &:= \exists u_0 \dots \exists u_k \exists v_0 \dots \exists v_k \exists r [(v_i \text{ are pairwise distinct}) \& \bigwedge_i \psi_{q_i}^0(u_i, v_i) \& \\ &\quad (r \text{ is a complement of } v_0 \vee \dots \vee v_k) \& (x = u_0 \vee \dots \vee u_k \vee r)]. \end{aligned}$$

If (2a) holds for a , then $\psi_a(x)$ is defined in a similar way. It is not hard to show that the condition $\mathcal{M} \models \psi_a(b)$ implies that $(\mathcal{M}, a) \cong (\mathcal{M}, b)$. Hence, ψ_a is a complete formula of the theory $Th(\mathcal{M})$.

Construction of a complete formula for a tuple $\bar{a} = a_0, \dots, a_k$, where $k \geq 1$, is a little bit more tricky. Nevertheless, this can be done using a more careful analysis. A typical example looks like follows: Suppose that $a_0 = p_0 \vee p_1$ and $a_1 = q_0 \vee q_1 \vee q_2 \vee (0, 0, 0, \top_3)$, where $0 < p_i < e_i$ and $0 < q_i < e_i$. Then a complete formula for the tuple $\bar{a} = a_0, a_1$ is defined as follows:

$$\begin{aligned} \psi_{\bar{a}}(x_0, x_1) &:= \exists u_0 \exists u_1 \exists v_0 \exists v_1 \exists v_2 \exists w_0 \exists w_1 \exists w_2 \exists r [(w_i \text{ are pairwise distinct}) \& \\ &\quad \bigwedge_{i \leq 1} \psi_{p_i}^0(u_i, w_i) \& \bigwedge_{j \leq 2} \psi_{q_j}^0(v_j, w_j) \& (r \text{ is a complement of } (w_0 \vee w_1 \vee w_2)) \& \\ &\quad (x_0 = u_0 \vee u_1) \& (x_1 = v_0 \vee v_1 \vee v_2 \vee r)]. \end{aligned}$$

Lemma 4.1 is proved. □

The next two lemmas show that \mathbf{d} is the degree of categoricity for \mathcal{M} .

Lemma 4.2. *The structure \mathcal{M} is \mathbf{d} -computably categorical.*

For reasons of space, the proof of Lemma 4.2 is omitted.

Lemma 4.3. *There is a computable copy \mathcal{N} of \mathcal{M} such that for any isomorphism f from \mathcal{M} onto \mathcal{N} , we have $f \geq_T \mathbf{d}$.*

Proof. For $e \in \omega$, we define a computable linear order \mathcal{R}_e :

$$\begin{aligned} \mathcal{R}_{4k} &:= \mathcal{D}_k + 1 + \eta + \mathcal{C}_k + (2k + 1), \\ \mathcal{R}_{4k+1} &:= \mathcal{C}_k + 1 + \eta + \omega^m \cdot 2 + (2k + 1), \\ \mathcal{R}_{4k+2} &:= \omega^m + 1 + \eta + \mathcal{E}_k + (2k + 2), \\ \mathcal{R}_{4k+3} &:= \mathcal{E}_k + 1 + \eta + \omega^m \cdot 2 + (2k + 2). \end{aligned}$$

It is easy to show that the structure $\mathcal{N} := \sum_{e \in \omega} \mathcal{H}(\mathcal{R}_e)$ is a computable copy of \mathcal{M} . Let u_k denote the element $(0, 0, \dots, 0, d_k, \perp_{k+1})$ from the algebra \mathcal{N} , where d_k is the greatest element in $\mathcal{H}(\mathcal{R}_k)$.

Suppose that f is an isomorphism from \mathcal{M} onto \mathcal{N} . Note the following: If $k \notin \emptyset^{(2m)}$, then $f(e_{4k+2}) = u_{4k+2}$; otherwise, $f(e_{4k+2}) = u_{4k+3}$. Therefore, we have $f \geq_T \mathbf{0}^{(2m)}$.

Furthermore, $k \in \bar{D}$ if and only if $k \in B$ or $f(e_{4k}) = u_{4k}$. Hence, \bar{D} is c.e. in $(f \oplus \mathbf{0}^{(2m)})$. The choice of D implies that $D \leq_T (f \oplus \mathbf{0}^{(2m)}) \equiv_T f$. Lemma 4.3 is proved. \square

The proof for the case when $n = 2m + 1$, $m \geq 1$, is essentially the same modulo the following key modification: we use ordinals ω^{m+1} and $\omega^{m+1} + \omega^m$ in place of ω^m and $\omega^m \cdot 2$, respectively. More details on this case can be recovered from the discussion in [7, p. 610]. This concludes the proof of Theorem 4.1. \square

5 Further Discussion

First, we note that the proof of Theorem 4.1 can be extended in a natural way (see, e.g., [7, Theorem 4] for a similar situation) to obtain.

Corollary 5.1. *Suppose that α is an infinite computable ordinal, and \mathbf{d} is a Turing degree which is d.c.e. in and above $\mathbf{0}^{(\alpha+1)}$. Then there exists a computable Heyting algebra with degree of categoricity \mathbf{d} .*

Notice that Proposition 3.1 implies the following: If $\mathbf{d} \neq \mathbf{0}^{(\omega+1)}$, then the algebra from the corollary cannot be a homogeneous model.

Second, we provide examples of prime models with no degree of categoricity. In [7, Corollary 3], it was proved that for any computable successor ordinal $\alpha \geq 2$, there is a computable Heyting algebra such that its categoricity spectrum is equal to the set of all PA degrees over $\mathbf{0}^{(\alpha)}$. A not difficult analysis of the proof reveals

Corollary 5.2. *Suppose that $n \geq 2$ is a natural number. Then there exists a computable prime Heyting algebra \mathcal{M} such that the categoricity spectrum of \mathcal{M} contains precisely the PA degrees over $\mathbf{0}^{(n)}$.*

References

1. Anderson, B.A., Csima, B.F.: Degrees that are not degrees of categoricity. *Notre Dame J. Form. Log.* **57**(3), 389–398 (2016). <https://doi.org/10.1215/00294527-3496154>

2. Ash, C.J.: Recursive labelling systems and stability of recursive structures in hyperarithmetical degrees. *Trans. Am. Math. Soc.* **298**(2), 497–514 (1986). <https://doi.org/10.1090/S0002-9947-1986-0860377-7>
3. Ash, C.J., Knight, J.F.: Pairs of recursive structures. *Ann. Pure Appl. Log.* **46**(3), 211–234 (1990). [https://doi.org/10.1016/0168-0072\(90\)90004-L](https://doi.org/10.1016/0168-0072(90)90004-L)
4. Ash, C.J., Knight, J.F.: *Computable Structures and the Hyperarithmetical Hierarchy*. *Studies in Logic and the Foundations of Mathematics*, vol. 144. Elsevier Science B.V., Amsterdam (2000)
5. Bazhenov, N.: Autostability spectra for decidable structures. *Math. Struct. Comput. Sci.* **28**(3), 392–411 (2018). <https://doi.org/10.1017/S096012951600030X>
6. Bazhenov, N.A.: Degrees of autostability for linear orders and linearly ordered abelian groups. *Algebra Log.* **55**(4), 257–273 (2016). <https://doi.org/10.1007/s10469-016-9395-4>
7. Bazhenov, N.A.: Effective categoricity for distributive lattices and Heyting algebras. *Lobachevskii J. Math.* **38**(4), 600–614 (2017). <https://doi.org/10.1134/S1995080217040035>
8. Bazhenov, N.A., Marchuk, M.I.: Degrees of autostability for prime Boolean algebras. *Algebra Logic*. To appear
9. Bazhenov, N.A., Yamaleev, M.M.: Degrees of categoricity of rigid structures. In: Kari, J., Manea, F., Petre, I. (eds.) *CiE 2017*. LNCS, vol. 10307, pp. 152–161. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58741-7_16
10. Chang, C.C., Keisler, H.J.: *Model Theory*. North-Holland, Amsterdam (1973)
11. Csima, B.F., Franklin, J.N.Y., Shore, R.A.: Degrees of categoricity and the hyperarithmetic hierarchy. *Notre Dame J. Form. Log.* **54**(2), 215–231 (2013). <https://doi.org/10.1215/00294527-1960479>
12. Fokina, E., Frolov, A., Kalimullin, I.: Categoricity spectra for rigid structures. *Notre Dame J. Form. Log.* **57**(1), 45–57 (2016). <https://doi.org/10.1215/00294527-3322017>
13. Fokina, E.B., Harizanov, V., Melnikov, A.: Computable model theory. In: Downey, R. (ed.) *Turing’s Legacy: Developments from Turing’s Ideas in Logic*, *Lect. Notes Logic*, vol. 42, pp. 124–194. Cambridge University Press, Cambridge (2014). <https://doi.org/10.1017/CBO9781107338579.006>
14. Fokina, E.B., Kalimullin, I., Miller, R.: Degrees of categoricity of computable structures. *Arch. Math. Log.* **49**(1), 51–67 (2010). <https://doi.org/10.1007/s00153-009-0160-4>
15. Fröhlich, A., Shepherdson, J.C.: Effective procedures in field theory. *Philos. Trans. Roy. Soc. Lond. Ser. A* **248**, 407–432 (1956). <https://doi.org/10.1098/rsta.1956.0003>
16. Frolov, A.N.: Effective categoricity of computable linear orderings. *Algebra Log.* **54**(5), 415–417 (2015). <https://doi.org/10.1007/s10469-015-9362-5>
17. Goncharov, S.S., Bazhenov, N.A., Marchuk, M.I.: Index set of linear orderings that are autostable relative to strong constructivizations. *J. Math. Sci.* **221**(6), 840–848 (2017). <https://doi.org/10.1007/s10958-017-3272-0>
18. Mal’tsev, A.I.: Constructive algebras I. *Russ. Math. Surv.* **16**, 77–129 (1961). <https://doi.org/10.1070/RM1961v016n03ABEH001120>
19. Mal’tsev, A.I.: On recursive abelian groups. *Sov. Math. Dokl.* **32**, 1431–1434 (1962)
20. Miller, R.: d-computable categoricity for algebraic fields. *J. Symb. Log.* **74**(4), 1325–1351 (2009). <https://doi.org/10.2178/jsl/1254748694>
21. Miller, R., Shlapentokh, A.: Computable categoricity for algebraic fields with splitting algorithms. *Trans. Am. Math. Soc.* **367**(6), 3955–3980 (2015). <https://doi.org/10.1090/S0002-9947-2014-06093-5>



Universality in Freezing Cellular Automata

Florent Becker¹, Diego Maldonado¹, Nicolas Ollinger¹,
and Guillaume Theyssier²(✉)

¹ Université Orléans, LIFO EA 4022, 45067 Orléans, France

² Institut de Mathématiques de Marseille (Université Aix Marseille,
CNRS, Centrale Marseille), Marseille, France
guillaume.theyssier@cnrs.fr

Abstract. Cellular Automata have been used since their introduction as a discrete tool of modelization. In many of the physical processes one may modelize thus (such as bootstrap percolation, forest fire or epidemic propagation models, life without death, etc), each local change is irreversible. The class of *freezing* Cellular Automata (FCA) captures this feature. In a freezing cellular automaton the states are ordered and the cells can only decrease their state according to this “freezing-order”.

We investigate the dynamics of such systems through the questions of simulation and universality in this class: is there a Freezing Cellular Automaton (FCA) that is able to simulate any Freezing Cellular Automata, i.e. an intrinsically universal FCA? We show that the answer to that question is sensitive to both the number of changes cells are allowed to make, and geometric features of the space. In dimension 1, there is no universal FCA. In dimension 2, if either the number of changes is at least 2, or the neighborhood is Moore, then there are universal FCA. On the other hand, there is no universal FCA with one change and Von Neumann neighborhood. We also show that monotonicity of the local rule with respect to the freezing-order (a common feature of bootstrap percolation) is also an obstacle to universality.

1 Introduction

Cellular automata (CA for short) are discrete dynamical systems at the crossroad of several research fields and points of view (dynamical systems theory, computer science, physical modeling, etc). In the pioneering works impulsed by von Neumann and Ulam in the 50–60s, when cellular automata were formally defined for the first time, two important themes were already present: universality [12, 15, 18] and growth dynamics [19]. Since then, these themes have received a considerable attention in the literature. Concerning universality, production of examples [1, 17] was accompanied by progresses on the formalization and the theoretical analysis of the concept [16], in particular with the emergence of intrinsic simulations and universality [4, 5]. Growing dynamics in cellular automata were also much studied, mostly through (classes of) examples with different points of view

[2, 7, 10, 11]. More recently, substantial work have been published on models of self-assembly tilings, most of which can be seen as a particular non-deterministic 2D CA where structures grow from a seed. Interestingly, the question of intrinsic universality was particularly studied in that case [6, 14].

A common feature of all these examples is that only a bounded number of changes per cell can occur during the evolution. To our knowledge, the first time that the class of CAs with that feature was considered as a whole is in [20] with a point of view of language recognition. More recently the notion of *freezing CA* was introduced in [9], which captures essentially the same idea with an explicit order on states, and a systematic study of this class (dynamics, predictability, complexity) was started. In particular it was established that the class is Turing universal (even in dimension 1).

In this paper, we study intrinsic universality in freezing CA as a first step to understand universality in growth dynamics in general. Our central result is the construction of such intrinsically universal freezing CA: it shows that the class of freezing CA is a natural computational model with maximally complex elements which can be thought of as machines that can be ‘programmed’ to produce any behavior of the class. Moreover, the universal CA that we construct are surprisingly small (5 states, see Sect. 3) which is in strong contrast with the complicated construction known to obtain intrinsic universality for the classical self-assembly aTAM model [6]. Our contribution also lays in the negative results we prove (Theorems 5, 6 and 7): interpreting them as necessary conditions to achieve universality for freezing CA, we obtain a clear landscape of the fundamental computational or dynamical features of this class.

The paper is organized as follows. In Sect. 2 we define the main concepts (freezing CA, and intrinsic simulation) and prove that the use of context-free simulation cannot lead to universality. Section 3 gives a general construction scheme to obtain universal freezing CA giving three positive results in three different settings depending on the dimension, the neighborhood and the maximum number of state changes per cell. In Sect. 4, we show several obstacles to the existence of universal freezing CA: dimension 1, 1 change per cell with von Neumann neighborhood in 2D, and monotonicity.

2 Definitions

Definition 1. *A cellular automaton F of dimension d and state set Q is a tuple $F = (d, Q, N, f)$, where d , its dimension is an integer, Q , its set of states is a finite set, $N \subset \mathbb{Z}^d$ is its finite neighborhood, and $f : Q^N \rightarrow Q$ is its local function.*

It induces a global function, which we also note F , acting on the set of configurations $Q^{\mathbb{Z}^d}$ as follows:

$$\forall c \in Q^{\mathbb{Z}^d}, \forall z \in \mathbb{Z}^d, F(c)_z = f(c|_{z+N})$$

Let e_1, \dots, e_d be the canonical basis of \mathbb{Z}^d , $VN_d = \{\mathbf{0}, e_1, \dots, e_d\}$ is the *von Neumann neighborhood*. We also use the following neighborhoods in dimension

2: $MN = \{(0, 0), (\pm 1, 0), (0, \pm 1), (\pm 1, \pm 1), (\mp 1, \pm 1)\}$ is the *Moore Neighborhood*; $LN = \{(0, 0), (1, 0), (0, 1)\}$ is the *L-neighborhood*.

In many cellular automata from the literature, there is a global bound on the number of times a cell can change: they are *bounded-change* cellular automata. This property is found in most of the Cellular Automata considered in bootstrap percolation, as well as in other well-known examples such as ‘Life without death’ [11] and various models of propagation phenomena like in [7]. We say that a CA is *k-change* if any cell in any orbit changes at most k times of state.

Moreover, in all those examples, the bound is defined through an explicit order on states. Such an automaton is a *freezing cellular automaton*; they were introduced in [9]. This *freezing-order* on state can also be used to define interesting subclasses (see Sect. 4).

Definition 2 (Freezing Cellular Automaton). *A CA F is a \prec -freezing CA, for some (partial) order \prec on states, if $F(c)_z \prec c_z$ for any configuration c and any cell z . A CA is freezing if it is \prec -freezing for some order.*

Any freezing CA is k -change for some k (at most the depth of its freezing order, but possibly less). For $V \subset \mathbb{Z}^d$, we note \mathbf{FCA}_V for the class of d -dimensional freezing cellular automata with neighborhood V . Finally, we set $\mathbf{FCA}^d = \bigcup_{V \subset \mathbb{Z}^d} \mathbf{FCA}_V$ and omit d when context makes it clear.

Intrinsic universality is defined through a notion of simulation between CA. Roughly speaking, F simulates G if there is an encoding of configurations of G into configurations of F such that one step of G is simulated through this encoding by a fixed number of steps of F .

Definition 3 (Simulation). *Let $T > 0$, and $B \subseteq \mathbb{Z}^d$ be a d -dimensional rectangular block, with size-vector $b \in \mathbb{Z}^d$. Let $C \subset \mathbb{Z}^d$ be a finite set, with $\mathbf{0} \in C$. Let $F = (d, Q, N, f)$ and $G = (d, Q', N', g)$ be two d -dimensional cellular automata. F simulates G with slowdown T , block B and context C if there is a coding map $\phi : Q_G^C \rightarrow Q_F^B$ such that the global map $\bar{\phi} : Q_G^{\mathbb{Z}^d} \rightarrow Q_F^{\mathbb{Z}^d}$ verifies:*

- $\bar{\phi}$ is injective;
- $\forall c \in Q_G^{\mathbb{Z}^d} : \bar{\phi}(G(c)) = F^T(\bar{\phi}(c))$.

where $\bar{\phi}$ is defined by: for $z \in \mathbb{Z}^d, r \in B, \bar{\phi}(c)_{bz+r} = \phi(c|_{z+C})_r$

When $C = \{\mathbf{0}\}$, this definition corresponds with the classical notion of ‘injective simulation’, as in [4, 5] and we call it *context-free*.

In the context of freezing CAs, context-free universality is prevented by the irreversibility of any computation performed by a putative U combined with the injectivity of the coding map, as witnessed by the following theorem.

Theorem 1 (No freezing context-free universality). *Let $d \in \mathbb{N}$, there is no $F \in \mathbf{FCA}^d$ which is context-free \mathbf{FCA}_{VN_d} -universal.*

Context-sensitive simulation can get us over this hurdle as we show below; it is akin to the notion of conjugacy in symbolic dynamics [13].

3 Constructing Intrinsically Universal FCA

We give a number of constructions for intrinsically universal freezing cellular automata. All of these exhibit the same running theme: if there is a means of crossing information asynchronously, then universality can be reached. This insight yields three constructions which are concrete implementations under various technical constraints of a common abstract construction. The abstract construction can be described by: the structure of macro-cells, the mechanism to trigger state change in each macro-cell, and the wiring between neighboring macro-cells to ensure communication. At this abstract level we assume that there is a mean to cross wires without interference. Another aspect of wiring is the necessity to put delays on some wires in order to keep synchronicity of information: it is a standard aspect of circuit encoding in CAs [1, 16], which we won't address in detail here but which can be dealt with by having wires make zigzag to adjust their length as desired. The freezing condition imposes strong restrictions on the way we can code, transport and process information. We focus below on where our construction differs from the classical approach in general CAs.

Wires are Fuses. It is not possible to implement classical wires where bits of information travel freely without violating the freezing condition. In all of our constructions wires are actually fuses that can be used only once and they are usually implemented with two states: 1 stays stable without presence of neighboring 0s and 0 propagates over neighboring 1s. With that behavior our wires can be trees connecting various positions in such a way that a 0 appearing at any position is broadcasted to the whole tree. A finite wire can either be uniformly in state $b \in \{0, 1\}$ in which case all leaves 'agree' on the bit of information transported by it, or not uniform in which case information is incoherent between leaves. As it will become clear later, our constructions will use wires between adjacent blocks (or macro-cells) in the simulator CA and our encodings require that those wires are in a coherent state (uniformly $b \in \{0, 1\}$): it is precisely in this aspect that we use the power of context sensitive simulations, because the

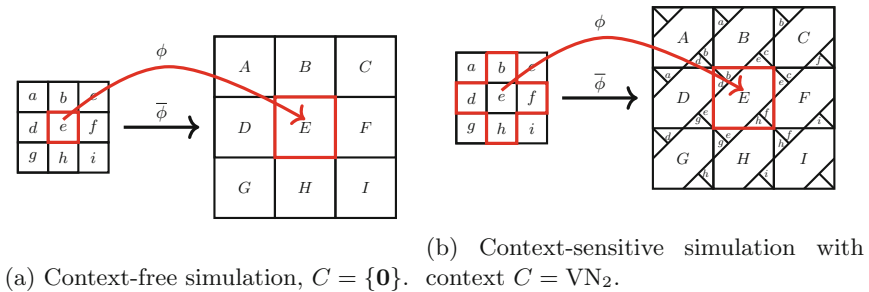


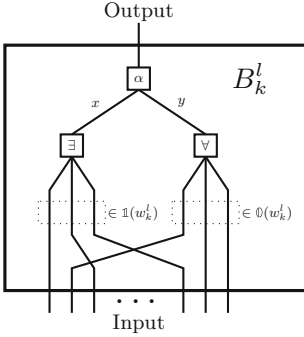
Fig. 1. Coding of states into blocks for the two modes of simulation

content of a block (or macro-cell) cannot be fixed independently of its neighbors in that case (Fig. 1).

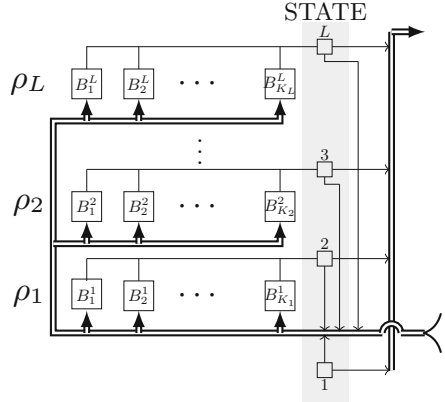
State Codification. In each macro-cell we must code in some way a (possibly very big) state that can change a (possibly very big) number of times: a classical binary encoding would violate the freezing condition so we actually use a unary coding. Given a finite set S and a quasi order (Q, \preceq) , let $q_0 \preceq \dots \preceq q_{|Q|-1}$ be a linearization of \preceq , and let $\iota(q_i) = i$. Then let $Q_u = 1^*0^+ \cap \{0, 1\}^{|Q|}$, and $\phi \in Q \rightarrow Q_u : q \mapsto 1^{\iota(q)}0^{|Q|-\iota(q)}$. Note that for any $i < |Q|$ we have $q \preceq q' \Leftrightarrow \phi(q)_i \leq \phi(q')_i$ (where \leq is the lexicographic order). Since ϕ is a bijection, for any cellular automaton F with state set Q , $\bar{\phi} \circ F \circ \bar{\phi}^{-1}$ is a cellular automaton isomorphic to F , with state set Q_u , which we call the *unary representation* of F . If F is \preceq -freezing, then its unary representation is \leq -freezing. We will use this unary encoding everywhere in the structure of our macro-cells: each state of a simulated CA F will be represented by a collection of wires representing the bits of an element of Q_u defined above. This encoding is coherent with the freezing property of the simulated CA because the fact that states can only decrease corresponds to the fact that the number of wires uniformly equal to 0 increases.

Neighborhood Matchers. The fundamental basic block of our construction is a circuit that detects a fixed pattern in the neighborhood and outputs a bit of information saying: “given this particular neighborhood pattern w , the new state of the macro-cell must be smaller than l ”. Our unary encoding is adapted for this because the predicate “smaller than l ” for a state translates into a condition on a single bit of an element of Q_u , that is to say a single wire in our concrete representation of states. Without loss of generality we assume that $F = (\mathbb{Z}^2, Q_u, f, N)$ is a FCA with state in unary representation. Take $L = |Q_u|$, and $m = |N|$. For $l \in Q_u$, let $\{w_1^l, \dots, w_{K_l}^l\} = \bigcup_{s' \preceq l} f^{-1}(s') = \{n \in (Q_u)^N \mid f(n) \leq l\}$. Take, for some state l , $w_k^l = (q_1, \dots, q_m) \in \bigcup_{s' \preceq l} f^{-1}(s')$ a fixed neighborhood with output smaller than l ; each state q_i is in $\{0, 1\}^L$, so w_k^l is a binary word in $\{0, 1\}^{mL}$. Given l and k , the logic gate diagram of Fig. 2a, the *Neighborhood Matcher*, called B_k^l , outputs 0 if and only if in the input in the wires is exactly w_k^l or the output cable was already in state 0. The i -th wire joins the i -th letter in w with either the \exists gate on the left if the i -th letter of w_k^l is 1 or the \forall gate on the right if the i -th letter of w_k^l is 0. Gate \exists triggers a 0 on wire x if at least on of its incoming wire is 0, while gate \forall triggers a 0 on wire y if all incoming wires are 0. Note that both behaviors are compatible with the freezing conditions since the set of wires in state 0 can only grow during evolution. The gate α at the top triggers a 0 on the output wire if wire y is in state 0 and wire x is in state 1 (see Fig. 2a). It is also a freezing gate, meaning that once it has triggered a 0 it will never change its state again, even if the wire x turns to state 0. Moreover this gate also turns into “trigger” state as soon as the output wire is 0.

Local Function Computation. Now we can compute the local function of F through a *macro-cell* \mathbb{C}_F , receiving the states $x = (x_n)_{n \in N}$ of the neighborhood as input, and yielding the next state $f(x)$ as output. For this we will divide the space into rows ρ_l for $l \in Q_u$, and some number of columns. Intuitively, the role



(a) Neighborhood matcher B_k^l . Notations $\mathcal{O}(w)$ and $\mathbb{1}(w)$ stand for the set of all indexes i s.t. $w_i = 0$ and $w_i = 1$ respectively. This block triggers a 0 on the output wire exactly when the input is w_k^l .



(b) Construction of a macro-cell. Single line represent wires transporting one bit and double lines represent multi-bit wires (representing a state).

Fig. 2. Recognizing one neighborhood (left), and wiring these *neighborhood matchers* into a *macro-cell* which computes the local function of F (right).

of row ρ_l is to maintain the information “the current state of the macro-cell is less than l ”. For a given $l \in Q_u$, ρ_l contains all block B_k^l for $k \in \{1, \dots, K_l\}$. The inputs are distributed to each block, and the outputs of all blocks in ρ_l are connected together by a broadcast wire. Thus, the final output in ρ_l is 0 as soon as one block K_k^l triggers, *i.e.* as soon as $f(x) \leq l$, see Fig. 2b. Notice that once a neighborhood matcher B_k^l in row l has output 0, the output of the macro-cell it belongs to must be less than l for ever: indeed, at the time when the B_k^l was triggered to output 0 the output value of the Macro-Cell must be less than l by definition of B_k^l , after that time the output is always less than l thanks to the freezing condition on the CA being simulated. Concatenating rows in the right order, we obtain as output of the gate the correct state codification $f(x)$ for any state of the neighborhood x received as input.

Information Exchanging. Given these basic blocks, one needs to embed one *macro-cell* per simulated cell on the simulator CA, and wire the inputs and outputs of neighboring macro-cells, as in Fig. 3. The wiring between *macro-cells* depends on the neighborhood of the simulated CA. In order to clarify the presentation we will always assume that the simulated CA has a von Neumann neighborhood which is enough to achieve universality thanks to the following lemma.

Lemma 1. *For any dimension d and any $F \in \mathbf{FCA}^d$ there is $G \in \mathbf{FCA}^d$ with von Neumann neighborhood that simulates F .*

The von Neumann wiring between *macro-cells* in dimension 2 is shown on Fig. 3. It is straightforward to generalize it to any dimension. Technically, thanks to Lemma 1, all the encoding map ϕ we use later have a von Neumann neighborhood context (C in Definition 3).

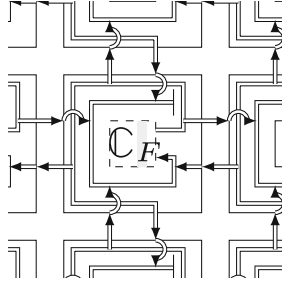


Fig. 3. The dashed block in the middle is a macro cell, as in Fig. 2b. The fat arrows exchange the state of each macro-cell with its neighbors.

Context-Sensitive Encoding. Given a configuration c of the simulated CA, the encoding is defined as follows. All wires of the construction are in a coherent state (same state along the wire). Each macro-cell holds a state of the configuration c represented in unary by the rows ρ_l described above. Wires incoming from neighboring macro-cells hold the information about neighboring states (hence the context-free encoding) which is transmitted to each block B_k^l . Inside each of this blocks the inputs arrive at gates “ \exists ” and “ \forall ” and these gates have eventually triggered a 0 on wires x and y . However, gate “ α ” has not yet triggered to preserve the property that the main wire of row ρ_l is coherent and represents the information on the *current* state of the macro-cell. Starting from that well encoded configuration, a simulation cycle begins by the possible triggering of “ α ” gates. After some time a well encoded configuration is reached again because changes coming from triggerings of α gates are broadcasted on each row, and, in each block B_k^l , the content up to the α gate is determined by the inputs.

We can now state three variants of the construction which differ essentially in the way crossing of information is implemented.

Theorem 2. $\exists U \in \mathbf{FCA}_{\text{VN}_2}$ with 5 states which is 2-change and \mathbf{FCA}^2 -universal.

Theorem 3. $\exists U \in \mathbf{FCA}_{\text{VN}_3}$ with 2 states which is 1-change and \mathbf{FCA}^3 -universal.

Theorem 4. $\exists U \in \mathbf{FCA}_{\text{MN}_2}$ with 4 states which is 1-change and \mathbf{FCA}^2 -universal.

4 Obstacles to FCA-Universality

The One-Dimensional Case. Although one-dimensional freezing CA can be computationally universal [9], they cannot be \mathbf{FCA}^1 -universal. This is a major difference with CA in general. The intuition behind this limitation is the following: in any given 1D freezing CA, there is a bound on the number of times a zone of consecutive cells can be crossed by a signal; and above this bound, the zone becomes a blocking word preventing any information flow between left and right halves around it.

Theorem 5 (Dimension 1). *There is no $F \in \mathbf{FCA}^1$ which is $\mathbf{FCA}_{\text{VN}_1}$ -universal, even with context-sensitive simulation.*

2D von Neumann 1-Change FCA: Information Crossing. We will show that there is no freezing universal FCA which is 1-change and has the von Neumann neighborhood. This result is to be contrasted with the case of self-assembly tiling where an intrinsically universal system exists [6] (although with an unavoidably more technical definition of simulation). The intuition is that the propagation of state changes in such FCA produces 4-connected paths that can not be crossed in the future of the evolution because only 1 state change per cell is possible. As shown in the construction of Theorem 2, two changes per cell are enough to get rid of this limitation, even with the von Neumann neighborhood.

We will show that no 1-change von Neumann FCA can simulate the following 2-change FCA $(\mathbb{Z}^2, Q_F, \text{LN}, f)$, with $Q_F = \{0, \leftarrow, \downarrow, \leftarrow\downarrow\}$ and where f is defined by

$$f(0, \downarrow, 0) = \downarrow \quad f(0, 0, \leftarrow) = \leftarrow \quad f(0, *, \leftarrow\downarrow) = \leftarrow \quad f(\downarrow, *, \leftarrow) = \leftarrow\downarrow$$

and $f(a, *, *) = a$ else, where $*$ stands for any state and the arguments of f correspond to neighborhood LN in the following order: center, north, east.

Theorem 6. *There is no automaton in $\mathbf{FCA}_{\text{VN}_2}$ which is 1-change and able to simulate F . Therefore there is no automaton in $\mathbf{FCA}_{\text{VN}_2}$ which is 1-change and $\mathbf{FCA}_{\text{VN}_2}$ -universal.*

Monotone FCA: Synchronous vs. Asynchronous Information. As for classical real function, we can consider the property of *monotonicity* in CA: given two configurations, one smaller than the other, their images by the CA compare in the same order. We are particularly interested in the case where the order on configurations is given by the order on states of a freezing CA. Several examples of such monotone FCAs were studied in literature. In particular, a simple model called *bootstrap percolation* was proposed by Chalupa in 1979 [3] to understand the properties in some magnetic materials. This model and several variants were studied from the point of view of percolation theory [2, 10], but also from the point of view of complexity of prediction [8].

Definition 4. *A \prec -freezing CA F of dimension d with alphabet Q is monotone if it satisfies: $\forall c, c' \in Q^{\mathbb{Z}^d} : c \prec c' \Rightarrow F(c) \prec F(c')$, where \prec is naturally extended to configurations by $c \prec c' \Leftrightarrow \forall z \in \mathbb{Z}^d : c_z \prec c'_z$.*

The intuitive limitation of monotone freezing CAs is that they must always produce a smaller state when two signals arrive simultaneously at some cell compared to when one of the two signals arrives before the other. We now exhibit a freezing non monotone CA F that does precisely the opposite (non-simultaneous arrival produces a smaller state). Next theorem shows that F cannot be simulated by any freezing monotone CA. F is defined by $(\mathbb{Z}^2, \{0, s_1, s_2, w, \Delta\}, LN, f)$ with f given by:

$$\begin{array}{lll}
 - f \left(\begin{array}{|c|c|} \hline s_i & \\ \hline w & 0 \\ \hline \end{array} \right) = s_i & - f \left(\begin{array}{|c|c|} \hline s_2 & \\ \hline s_1 & 0 \\ \hline \end{array} \right) = s_2 & - f \left(\begin{array}{|c|c|} \hline s_1 & \\ \hline \Delta & s_1 \\ \hline \end{array} \right) = s_1 \\
 - f \left(\begin{array}{|c|c|} \hline 0 & \\ \hline w & s_i \\ \hline \end{array} \right) = s_i & - f \left(\begin{array}{|c|c|} \hline s_2 & \\ \hline s_1 & w \\ \hline \end{array} \right) = s_2 & - f \left(\begin{array}{|c|c|} \hline s_1 & \\ \hline \Delta & w \\ \hline \end{array} \right) = s_2.
 \end{array}$$

and unspecified transitions let the state unchanged. F is \prec -freezing for the following order on states: $s_2 \prec s_1 \prec 0, \Delta, w$. Essentially F is a FCA sending the signals s_1 and s_2 towards south or west along the wires materialized by state w . s_2 can also move on wires made of s_1 . Δ plays the role of a non-monotone local gate: when two signals arrive simultaneously, a s_1 -signal is sent to the south, but when only one signal arrives, a s_2 -signal is sent to the south. F cannot be simulated by any monotone FCA, hence no monotone FCA can be **FCA**-universal.

Theorem 7. *For any $d \geq 1$, there is no freezing monotone CA of dimension d which is $\mathbf{FCA}_{\text{VN}_d}$ -universal.*

5 Perspectives

Here are some questions or research directions that we think are worth being considered:

- do CA with a bounded number of change per cell also exhibit intrinsic universality?
- what about intrinsic universality for non-deterministic or asynchronous freezing CA as a generalization of aTAM models?
- what are the limit sets of **FCA**?
- what can be said about **FCA** from an ergodic theory point of view (this includes questions from percolation theory)?

References

1. Banks, E.R.: Universality in cellular automata. In: Eleventh Annual Symposium on Switching and Automata Theory, Santa Monica, California. IEEE (1970)
2. Bollobás, B., Smith, P., Uzzell, A.: Monotone cellular automata in a random environment. *Comb. Probab. Comput.* **24**(4), 687–722 (2015)

3. Chalupa, J., Leath, P.L., Reich, G.R.: Bootstrap percolation on a Bethe lattice. *J. Phys. C: Solid State Phys.* **12**, L31–L35 (1979)
4. Delorme, M., Mazoyer, J., Ollinger, N., Theyssier, G.: Bulking I: an abstract theory of bulking. *Theor. Comput. Sci.* **412**(30), 3866–3880 (2011)
5. Delorme, M., Mazoyer, J., Ollinger, N., Theyssier, G.: Bulking II: classifications of cellular automata. *Theor. Comput. Sci.* **412**(30), 3881–3905 (2011)
6. Doty, D., Lutz, J.H., Patitz, M.J., Schweller, R.T., Summers, S.M., Woods, D.: The tile assembly model is intrinsically universal. In: *FOCS 2012 Proceedings*, pp. 302–310 (2012)
7. Fuentes, M.A., Kuperman, M.N.: Cellular automata and epidemiological models with spatial dependence. *Physica A: Stat. Mech. Appl.* **267**(3–4), 471–486 (1999)
8. Goles, E., Montealegre-Barba, P., Todinca, I.: The complexity of the bootstrapping percolation and other problems. *Theor. Comput. Sci.* **504**, 73–82 (2013)
9. Goles, E., Ollinger, N., Theyssier, G.: Introducing freezing cellular automata. In: Kari, J., Törmä, I., Szabados, M. (eds.) *Exploratory Papers of Cellular Automata and Discrete Complex Systems (AUTOMATA 2015)*, pp. 65–73 (2015)
10. Gravner, J., Griffeath, D.: Cellular automaton growth on \mathbb{Z}^2 : theorems, examples, and problems. *Adv. Appl. Math.* **21**(2), 241–304 (1998)
11. Griffeath, D., Moore, C.: Life without death is P-complete. *Complex Syst.* **10**, 437–448 (1996)
12. Holland, J.H.: A universal computer capable of executing an arbitrary number of subprograms simultaneously. In: Bukrs, A.W. (ed.) *Essays on Cellular Automata*, pp. 264–276. University of Illinois Press, Champaign (1970)
13. Lind, D.A., Marcus, B.: *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, New York (1995)
14. Meunier, P.-E., Patitz, M.J., Summers, S.M., Theyssier, G., Winslow, A., Woods, D.: Intrinsic universality in tile self-assembly requires cooperation. In: *SODA 2014 Proceedings*, pp. 752–771 (2014)
15. Von Neumann, J.: *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign (1966)
16. Ollinger, N.: Universalities in cellular automata. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) *Handbook of Natural Computing*, pp. 189–229. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9_6
17. Ollinger, N., Richard, G.: Four states are enough!. *Theor. Comput. Sci.* **412**(1–2), 22–32 (2011)
18. Thatcher, J.W.: Universality in the von neumann cellular model. In: Bukrs, A.W. (ed.) *Essays on Cellular Automata*, pp. 132–186. University of Illinois Press, Champaign (1970)
19. Ulam, S.M.: On some mathematical problems connected with patterns of growth of figures. In: Bukrs, A.W. (ed.) *Essays on Cellular Automata*, pp. 219–231. University of Illinois Press, Champaign (1970)
20. Vollmar, R.: On cellular automata with a finite number of state changes. In: Knödel, W., Schneider, H.-J. (eds.) *Parallel Processes and Related Automata. Computing Supplementum*, vol. 3, pp. 181–191. Springer, Vienna (1981). https://doi.org/10.1007/978-3-7091-8596-4_13



A Deontic Logic Reasoning Infrastructure

Christoph Benz Müller^(✉), Xavier Parent, and Leendert van der Torre

Computer Science and Communications, University of Luxembourg,
Esch-sur-Alzette, Luxembourg
c.benzmueller@gmail.com

Abstract. A flexible infrastructure for the automation of deontic and normative reasoning is presented. Our motivation is the development, study and provision of legal and moral reasoning competencies in future intelligent machines. Since there is no consensus on the “best” deontic logic formalisms and since the answer may be application specific, a flexible infrastructure is proposed in which candidate logic formalisms can be varied, assessed and compared in experimental ethics application studies. Our work thus links the historically rich research areas of classical higher-order logic, deontic logics, normative reasoning and formal ethics.

1 Introduction

If humans and intelligent machines are supposed to peacefully coexist, appropriate forms of machine-control and human-machine-interaction are required. This motivates the provision of legal and ethical reasoning competencies in intelligent machines. Bottom-up, learning based approaches (e.g. GenEth [1]) try to acquire ethical behaviour from dialogs with ethicist or from existing data. Top-down approaches (e.g. [21]) try to explicitly model selected ethical theories or policies and to enforce them in intelligent systems; cf. [20, 23] and the references therein. Our research, which is more in line with the top-down approach, assumes that suitable declarative, logical reasoning means and competencies are mandatory in intelligent machines, in particular in the context of legal and moral reasoning. Such competencies seem vital not only for guaranteeing sufficient degrees of reliability and accountability, but also for achieving human intuitive interaction means regarding explainability and transparency of decisions.

This paper therefore addresses the practical development of computational tools for normative reasoning based on deontic logics. Since ethical and legal theories/policies as well as suitable deontic logic formalisms are both still under development [34], we outline a flexible workbench to support empirical studies with such theories/policies in which the preferred logic formalisms themselves can still be varied, complemented, assessed and compared. The infrastructure we propose draws on both recent developments in universal logical reasoning in classical higher-order logic (HOL) [5] and the coalescence and improvements in interactive and automated theorem proving (ATP) in HOL, as witnessed by systems such as Isabelle/HOL [32], LEO-II [11] and Leo-III. We are thus linking

the historically rich research areas of HOL, deontic logics, normative reasoning and formal ethics. Moreover, since modern HOL theorem provers internally collaborate with effective SMT and SAT solvers, a novel bridge is provided from expressive deontic logics to SMT and SAT technology.

The paper is structured as follows. Section 2 surveys relevant deontic logic formalisms and discusses recent extensions to model ethical agency. Section 3 introduces norm-based deontic logic, our preferred framework. Sections 4 and 5, the core contribution of this paper, outline our flexible deontic reasoning infrastructure and sketch a case study in data protection.

2 Traditional Deontic Logic

Deontic logic [25,39] is the field of logic that is concerned with normative concepts such as obligation, permission, and prohibition. Alternatively, a deontic logic is a formal system capturing the essential logical features of these concepts. Typically, a deontic logic uses Op to mean that it is obligatory that p , (or it ought to be the case that p), and Pp to mean that it is permitted, or permissible, that p . The term “deontic” is derived from the ancient Greek *déon*, meaning “that which is binding or proper”. Deontic logic can be used for reasoning about normative multiagent systems, i.e. about multiagent organisations with normative systems in which agents can decide whether to follow the explicitly represented norms, and the normative systems specify how, and to which extent, agents can modify the norms. Normative multiagent systems need to combine normative reasoning with agent interaction, and thus raise the challenge to relate the logic of normative systems to aspects of agency.

Traditional (or “standard”) deontic logic (SDL) is a normal propositional modal logic of type KD, which means that it extends the propositional tautologies with the axioms $K: O(p \rightarrow q) \rightarrow (Op \rightarrow Oq)$ and $D: \neg(Op \wedge O\neg p)$, and it is closed under the inference rules *modus ponens* $p, p \rightarrow q / q$ and *generalization* or *necessitation* p / Op . Prohibition and permission are defined by $Fp = O\neg p$ and $Pp = \neg O\neg p$. SDL is an unusually simple and elegant theory. An advantage of its modal-logical setting is that it can easily be extended with other modalities such as epistemic or temporal operators and modal accounts of action.

Dyadic deontic logic (DDL) introduces a conditional operator $O(p/q)$, to be read as “it ought to be the case that p , given q ”. Many DDLs have been proposed to deal with so-called contrary-to-duty reasoning, cf. [18] for an overview on this area. An example is the DDL proposed by Hansson [28] and Åqvist [3,33], and the one proposed by Carmo and Jones [18,19].

To enable ethical agency a model of decision needs to be integrated in the deontic frames. Horty’s STIT logic [29], which combines deontic logic with a modal logic of action, has been proposed as a starting point. The semantic condition for the STIT-ought is a utilitarian generalisation of the SDL view that “it ought be that A ” means that A holds in all deontically optimal worlds.

3 Norm-Based Deontic Logic

The term “norm-based” deontic logic has been coined by Hansen [27] to refer to a family of frameworks analysing the deontic modalities not with reference to a set of possible worlds (some of them being more ideal than others), but with reference to a set of explicitly given norms. In such a framework, the central question is: given some input (e.g. a fact) and a set of explicitly given conditional norms (a normative system), what norms apply? Thus, the perspective is slightly different from the traditional setting, focusing on inference patterns [35].

We propose to base the AI deontic reasoner on a specific norm-based deontic logic called input/output (I/O) logic. Initially devised by Makinson [31] and further developed over the past years by van der Torre and colleagues, I/O logic has gained increased recognition in the AI community. This is evidenced by the fact that the framework has its own chapter in the aforementioned Handbook of Deontic Logic and Normative Systems [34]. I/O logic can be viewed as a rule-based system. The knowledge base takes the form of a set of rules of the form (a,b) to be read as “if a then b ”. The key feature of I/O logic is that it uses an operational semantics, based on the notion of detachment, rather than a truth-functional one in terms of truth-values and possible worlds. On the semantical side, the meaning of the deontic concepts is given in terms of a set of procedures, called I/O operations, yielding outputs (e.g., obligations) for inputs (facts). On the syntactical side, the proof-theory is formulated as a set of inference rules manipulating pairs of formulas rather than individual formulas. The framework covers functionalities that are unanimously regarded as characteristic of the legal domain, and thus required to enable effective legal reasoning:

1. Support for the modelling of *constitutive rules*, which define concepts or constitute activities that cannot exist without such rules (e.g. legal definitions such as “property”), and *prescriptive rules*, which regulate actions by making them obligatory, permitted, or prohibited.
2. Management of the *reification of rules* that are objects with properties, such as jurisdiction, authority, temporal attributes [26].
3. Implementation of *defeasibility*—see [26,38]; when the antecedent of a rule is satisfied by the facts of a case (or via other rules), the conclusion of the rule presumably holds, but is not necessarily true.

4 Deontic Logic Reasoning Machinery

In a nutshell, a reasoner is a tool that can perform reasoning tasks in a given application domain. Reasoning thereby refers to the process of deriving or concluding information that is not explicitly encoded in the knowledge base. In our context, a number of reasoning tasks are particularly relevant. These include:

- *Compliance checking*: Is the current situation compliant with a given regulation (a set of formally represented norms)?

- *Consistency checking*: Is a given regulation consistent? Is such-and-such norm, as part of a given regulation, consistent with this other set of norms, stemming from another regulation? Is such-and-such legal interpretation consistent with another one?
- *Entailment checking*: Does such-and-such obligation or legal interpretation follow from a given regulation?

Some of these tasks, e.g. consistency checking, are well supported by model finders, while others, such as entailment checking, in general require theorem proving technology. A powerful deontic reasoner should thus ideally provide both model finding and theorem proving. While this is comparably easy to achieve for many decidable propositional fragments of deontic logics, it becomes much less so for their quantified extensions.

The quest for “a single, best deontic logic” is still open, and it eventually will remain so for a long time to come. While I/O logic is the solution favoured in our group (see Sect. 3), we also want to stay open-minded regarding alternative proposals, such as the DDL by Hansson or the one by Carmo and Jones. Moreover, it is unclear yet whether regulatory texts can always be abstracted and simplified to the point that pure propositional logic encodings are feasible and justified, or whether first-order (FO) or even higher-order (HO) encodings are required instead. This poses a multitude of concrete challenges for a flexible deontic reasoning infrastructure.

These raised questions motivate empirical studies in which the different options are systematically compared and assessed within well selected application studies (see Sect. 5). However, for such empirical work to be feasible, implementations of the different deontic candidate logics have to be provided first, both on the propositional level and ideally also on the FO and HO level. Moreover, it is reasonable to ensure that these implementations remain maximally comparable regarding the technological foundations they are based on, since this may improve the fairness and the significance of (conceptual) empirical evaluations.

A Meta-Logical Approach for Flexible, Deontic Logic Reasoning. Our currently preferred solution for the implementation of a most flexible deontic reasoner in the above sense is to work with a meta-logical approach to universal logic reasoning, cf. [5] and the references therein. We subsequently instantiate this approach in our ongoing project for various deontic candidate logics, including I/O logic and the other ones as mentioned. The approach, which is based on shallow semantical embeddings (SSE) [10] of these logics in HOL¹, has a very pragmatic motivation, foremost reuse of tools, simplicity and elegance. It utilises HOL as a unifying meta-logic in which the syntax and semantics of varying other logics can be explicitly modeled and flexibly combined. Off-the-shelf interactive

¹ HOL as addressed here refers to a (simply) typed logic of functions, which has been proposed by Church [2]. It provides lambda-notation, as an elegant and useful means to denote unnamed functions, predicates and sets. Types in HOL eliminate paradoxes and inconsistencies. For more information on HOL see the literature [7].

and automated theorem provers, and model finders can then be employed to reason about and within the shallowly embedded logics.

Evidence from Previous Work. Respective experiments with this approach have e.g. been conducted in metaphysics (cf. [13]). An initial focus thereby has been on computer-supported assessments of rational arguments, in particular, of modern, modal logic variants of the ontological argument for the existence of God. In the course of these experiments, in which the SSE approach was applied for automating different variants of HO quantified modal logics [10], the HO theorem prover LEO-II even detected an previously unnoticed inconsistency in Kurt Gödel’s prominent variant of the ontological argument, while the soundness of the slightly modified variant by Dana Scott was confirmed and all argument steps were verified. Further modern variants of the argument have subsequently been studied with the approach, and theorem provers have even contributed to the clarification of an unsettled philosophical dispute [12].

Deontic Logics Already Covered in the SSE Approach. The following deontic logics have already been “implemented” by utilising the SSE approach:

- SDL: All logics from the modal logic cube, including logic KD, i.e. SDL, have meanwhile been faithfully implemented in the SSE approach [10]. These implementations scale for FO and even HO extensions.
- the DDL by Carmo and Jones [19]: A semantic embedding of the propositional fragment of this logic in Isabelle/HOL is already available [6, 8], and most recently the ATP Leo-III has been adapted to accept DDL as input.
- I/O logic [31]: The main challenge comes from the fact that the framework does not have a truth-functional semantics, but an operational one. First experiments with the semantic embedding of the I/O-operator “*out1*” (called simple-minded) in Isabelle/HOL are promising [9]; see also Sect. 5.

Some relevant I/O logic variants have very recently been studied (see [36]), and we conjecture that some of these variants are related to certain non-normal modal logics, e.g. conditional logics with a selection function semantics or similar logics with a neighbourhood semantics. However, the embedding of such logics has already been studied in the first authors previous work [4]. It should thus be possible to benefit from these existing results in the given context.

A most interesting aspect is, that the SSE approach even supports meta-logical investigations within Isabelle/HOL in which the possible relationships between I/O and conditional logics we hinted at can be formally assessed. Examples of meta-logical studies are e.g. mentioned in [10].

Moreover, the SSE approach enables the reuse of existing model finding and theorem proving technology within the Isabelle/HOL proof assistant [32]. The automated reasoning systems that are integrated with Isabelle/HOL, respectively that are available via remote calls, include state-of-the-art SMT solvers, FO and HO theorem provers, and model finders; cf. [15] and the references therein. This infrastructure, in combination with the SSE approach, meets our demanding requirements regarding flexibility along different axes.

While there is some related work, see e.g. [17,21,24,37], we are not aware of any other existing deontic logic reasoning approach and associated machinery that offers the same amount of flexibility and scalability.

Another advantage of the SSE approach, when implemented within powerful proof assistants such as Isabelle/HOL, is that proof construction (interactive or automated) can be supported at different levels of abstraction. For this note that proof protocols/objects may generally serve two different purposes: (a) they may provide an independently verifiable explanation in a (typically) well-defined logical calculus, or (b) they may provide an intuitive explanation to the user why the problem in question has been answered positively or negatively. Many reasoning tools, if they are offering proof objects at all, do generate only objects of type (a). The SSE approach, however, has already demonstrated its capabilities to provide both types of responses simultaneously in even most challenging logic settings. For example, a quite powerful, abstract level theorem prover for hyper-intensional HO modal logic has been provided by Kirchner [30]. He encoded an abstract level proof calculus for this logic as proof tactics and he demonstrated how these abstract level proof tactics can again be elegantly automated using respective tools in Isabelle/HOL. Kirchner then successfully applied his reasoning infrastructure to reveal, assess and intuitively communicate a non-trivial paradox in Zalta’s “Principia-logico Metaphysica” [40].

Drawing on the results and experiences from previous work, the ambition of our ongoing project is to further extend the already existing implementations of deontic logics in Isabelle/HOL towards a most powerful, flexible and scalable deontic logic reasoning infrastructure. A core motivation thereby is to support empirical studies in various application scenarios, and to assess and compare the suitability, adequacy and performance of individual deontic logic solutions for the engineering of moral agents and explainable intelligent systems.

5 Case Study: Data Protection

The General Data Protection Regulation (GDPR, Regulation EU 2016/679) is a relevant and interesting application scenario for normative reasoning. It is a regulation by which the European Parliament, the Council of the European Union and the European Commission intend to strengthen and unify data protection for all individuals within the European Union. The regulation becomes enforceable from 25 May 2018. We present two sample norms of the GDPR:

1. Personal data shall be processed lawfully (Art. 5). For example, the data subject must have given consent to the processing of his or her personal data for one or more specific purposes (Art. 6/1.a).
2. If the personal data have been processed unlawfully (none of the requirements for a lawful processing applies), the controller has the obligation to erase the personal data in question without delay (Art. 17.d, right to be forgotten).

When combined with the following a typical CTD-structure is exhibited.

3. It is obligatory e.g. as part of a respective agreement between a customer and a company) to keep the personal data (as relevant to the agreement) provided that it is processed lawfully.
4. Some data in the context of such an agreement has been processed unlawfully.

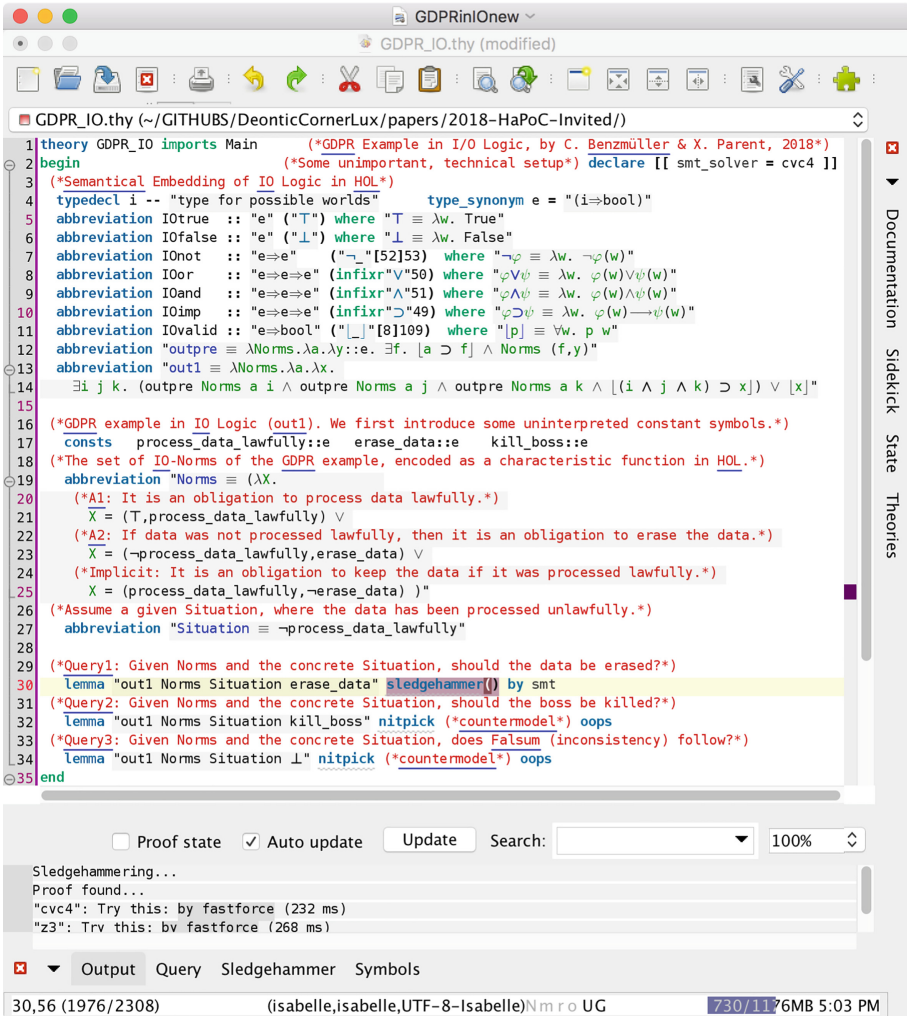


Fig. 1. GDPR example scenario in Isabelle/HOL.

The latter information pieces are not explicit part of the GDPR. Instead they are to be seen as implicit. 3 comes from another regulation, with which the GDPR has to co-exists. 4 is a factual information — it is exactly the kind of world

situations the GDPR wants to regulate. This example is given for illustrative purposes only. It provides a taste of what this knowledge base might look like.

In a recent technical report [8] we illustrate the practical challenge of such a CTD scenario. Namely, when the above norms are encoded in SDL, an inconsistency follows, meaning that everything is implied in the given context, including arbitrarily weird and unethical conclusions such as the obligation to “kill the boss”. In the same report we demonstrate that our SSE based implementation of Carmo and Jones’s DDL is in contrast not suffering from this effect. No inconsistency follows when the above scenario is modelled in DDL. The obligation to erase the data, however, is implied as intended.

We now analyse the above CTD scenario in the context of I/O logic. This is done in Fig. 1, which first presents, in lines 4–13, an SSE based implementation of I/O logic in Isabelle/HOL.² A possible world semantics is employed in this embedding to adequately address an extensionality issue we have revealed in our previous work. This issue, and its solution, is discussed in more detail in a technical report [9]. The prescriptive rules of the GDPR scenario are then modelled in lines 19–25, where the set of given *Norms* is defined as $\{(\top, \text{process_data_lawfully}), (\neg \text{process_data_lawfully}, \text{erase_data}), (\text{process_data_lawfully}, \neg \text{erase_data})\}$. The given *Situation*, in which we have $\neg \text{process_data_lawfully}$, is defined in line 27. Subsequently, three different queries are answered by the reasoning tools integrated with Isabelle/HOL. The first query asks whether the data should be erased in the given context. The ATPs integrated with Isabelle/HOL via the Sledgehammer tool [15] respond quickly: the SMT solver CVC4 [22] and the first-order prover Spass [16] return a proof within a few milliseconds. For queries 2 and 3 the ATPs fail (not shown here), but now the countermodel finder Nitpick [14] responds and presents counterarguments to both queries. That is, we receive the intended negative answers to queries 2 and 3 when the GDPR example is modelled in our preferred I/O logic. It is worth mentioning that I/O logic (and also DDL) have never been automated before.

6 Conclusion

The deontic logic reasoning infrastructure we have presented supports empirical studies on legal and ethical theories/policies in which the particular deontic logic formalisms itself can be varied, assessed and compared in context. We believe that this infrastructure can fruitfully support the development of much needed logic based approaches towards ethical agency. The solution we have presented supports a wide range of specific deontic logic variants, and it also scales for their first-order and higher-order extensions. In fact, our infrastructure already now implements a wider range of deontic and related logics than any other competitor systems we are aware of.

² The semantical embedding of out_1 as presented here is technically still an approximative solution. For a complete embedding, x needs to be defined as a consequence of an arbitrary number of facts (instead of just i , j and k) in lines 13 and 14.

References

1. Anderson, M., Anderson, S.L.: Toward ensuring ethical behavior from autonomous systems: a case-supported principle-based paradigm. *Ind. Robot* **42**(4), 324–331 (2015)
2. Andrews, P.: Church’s type theory. In: Zalta, E. (ed.) *The Stanford Encyclopedia of Philosophy*, Spring 2014 edn (2014)
3. Åqvist, L.: Deontic logic. In: Gabbay, D., Guenther, F. (eds.) *Handbook of Philosophical Logic*, 2nd edn, vol. 8, pp. 147–264. Kluwer Academic Publishers, Dordrecht, Holland (2002)
4. Benzmüller, C.: Cut-elimination for quantified conditional logic. *J. Philos. Logic* **46**(3), 333–353 (2017)
5. Benzmüller, C.: Recent successes with a meta-logical approach to universal logical reasoning (extended abstract). In: da Costa Cavalheiro, S.A., Fiadeiro, J.L. (eds.) *SBMF 2017. LNCS*, vol. 10623, pp. 7–11. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70848-5_2
6. Benzmüller, C., Farjami, A., Parent, X.: Faithful semantical embedding of a dyadic deontic logic in HOL. Technical report, CoRR (2018). <https://arxiv.org/abs/1802.08454>
7. Benzmüller, C., Miller, D.: Automation of higher-order logic. In: Gabbay, D.M., Siekmann, J.H., Woods, J. (eds.) *Handbook of the History of Logic. Computational Logic*, vol. 9, pp. 215–254. North Holland, Elsevier (2014)
8. Benzmüller, C., Parent, X.: First experiments with a flexible infrastructure for normative reasoning. Technical report, CoRR (2018). <http://arxiv.org/abs/1804.02929>
9. Benzmüller, C., Parent, X.: I/O logic in HOL – first steps. Technical report, CoRR (2018). <https://arxiv.org/abs/1803.09681>
10. Benzmüller, C., Paulson, L.: Quantified multimodal logics in simple type theory. *Log. Univers.* **7**(1), 7–20 (2013)
11. Benzmüller, C., Sultana, N., Paulson, L.C., Theiß, F.: The higher-order prover LEO-II. *J. Autom. Reason.* **55**(4), 389–404 (2015)
12. Benzmüller, C., Weber, L., Woltzenlogel Paleo, B.: Computer-assisted analysis of the Anderson-Hájek controversy. *Log. Univers.* **11**(1), 139–151 (2017)
13. Benzmüller, C., Woltzenlogel Paleo, B.: The inconsistency in Gödel’s ontological argument: a success story for AI in metaphysics. In: Kambhampati, S. (ed.) *IJCAI 2016*, vol. 1–3, pp. 936–942. AAAI Press (2016)
14. Blanchette, J.C., Nipkow, T.: Nitpick: a counterexample generator for higher-order logic based on a relational model finder. In: Kaufmann, M., Paulson, L.C. (eds.) *ITP 2010. LNCS*, vol. 6172, pp. 131–146. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14052-5_11
15. Blanchette, J.C., Böhme, S., Paulson, L.C.: Extending Sledgehammer with SMT solvers. *J. Autom. Reason.* **51**(1), 109–128 (2013)
16. Blanchette, J.C., Popescu, A., Wand, D., Weidenbach, C.: More SPASS with Isabelle - superposition with hard sorts and configurable simplification. In: Beringer, L., Felty, A. (eds.) *ITP 2012. LNCS*, vol. 7406, pp. 345–360. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32347-8_24
17. Bringsjord, S., Arkoudas, K., Bello, P.: Toward a general logicist methodology for engineering ethically correct robots. *IEEE Intell. Syst.* **21**, 38–44 (2006)
18. Carmo, J., Jones, A.J.I.: Deontic logic and contrary-to-duties. In: Gabbay, D.M., Guenther, F. (eds.) *Handbook of Philosophical Logic*, vol. 8, pp. 265–343. Springer, Dordrecht (2002). https://doi.org/10.1007/978-94-010-0387-2_4

19. Carmo, J., Jones, A.J.I.: Completeness and decidability results for a logic of contrary-to-duty conditionals. *J. Logic Comput.* **23**(3), 585–626 (2013)
20. Dennis, L.A., Fischer, M.: Practical challenges in explicit ethical machine reasoning. In: ISAIM 2018, Fort Lauderdale, Florida, USA (2018)
21. Dennis, L.A., Fisher, M., Slavkovik, M., Webster, M.: Formal verification of ethical choices in autonomous systems. *Rob. Auton. Syst.* **77**, 1–14 (2016)
22. Deters, M., Reynolds, A., King, T., Barrett, C.W., Tinelli, C.: A tour of CVC4: how it works, and how to use it. In: Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, 21–24 October 2014, p. 7. IEEE (2014)
23. Dignum, V.: Responsible autonomy. In: IJCAI 2017, pp. 4698–4704 (2017)
24. Furbach, U., Schon, C., Stolzenburg, F.: Automated reasoning in deontic logic. In: Murty, M.N., He, X., Chillarige, R.R., Weng, P. (eds.) MIWAI 2014. LNCS (LNAI), vol. 8875, pp. 57–68. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13365-2_6
25. Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.): Handbook of Deontic Logic and Normative Systems. College Publications, London (2013)
26. Gordon, T.: The Pleading Game: An Artificial Intelligence Model of Procedural Approach. Springer, New York (1995)
27. Hansen, J.: Reasoning about permission and obligation. In: Hansson, S.O. (ed.) David Makinson on Classical Methods for Non-Classical Problems. OCL, vol. 3, pp. 287–333. Springer, Dordrecht (2014). https://doi.org/10.1007/978-94-007-7759-0_14
28. Hansson, B.: An analysis of some deontic logics. *Noûs* **3**(4), 373–398 (1969)
29. Horty, J.: Agency and Deontic Logic. OUP, London (2009)
30. Kirchner, D., Benzmüller, C., Zalta, E.N.: Mechanizing principia logico-metaphysica in functional type theory. *CoRR* (2017). <https://arxiv.org/abs/1711.06542>
31. Makinson, D., van der Torre, L.W.N.: Input/output logics. *J. Philos. Logic* **29**(4), 383–408 (2000)
32. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. LNCS, vol. 2283. Springer, Heidelberg (2002). <https://doi.org/10.1007/3-540-45949-9>
33. Parent, X.: Completeness of Åqvist’s systems E and F. *Rev. Symb. Logic* **8**(1), 164–177 (2015)
34. Parent, X., van der Torre, L.: Input/output logic. In: Gabbay et al. [25], pp. 499–544 (2013)
35. Parent, X., van der Torre, L.: Detachment in normative systems: examples, inference patterns, properties. *IfCoLog J. Logics Appl.* **4**(9), 2295–3039 (2017)
36. Parent, X., van der Torre, L.: The pragmatic oddity in a norm-based semantics. In: Governatori, G. (ed.) ICAIL 2017, Proceedings, pp. 169–178. ACM, New York (2017)
37. Pereira, L.M., Saptawijaya, A.: Programming Machine Ethics. Studies in Applied Philosophy, Epistemology and Rational Ethics, vol. 26. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-29354-7>
38. Sartor, G.: Legal Reasoning: A Cognitive Approach to Law. Springer, Dordrecht (2005)
39. von Wright, G.H.: Deontic logic. *Mind* **60**, 1–15 (1951)
40. Zalta, E.N.: Principia logico-metaphysica. Draft version (2016). <https://mally.stanford.edu/principia.pdf>



Optimized Program Extraction for Induction and Coinduction

Ulrich Berger^(✉) and Olga Petrovska

Swansea University, Swansea SA2 8PP, Wales, UK
{u.berger,olga.petrovska}@swansea.ac.uk

Abstract. We prove soundness of an optimized realizability interpretation for a logic supporting strictly positive induction and coinduction. The optimization concerns the special treatment of Harrop formulas which yields simpler extracted programs. We show that wellfounded induction is an instance of strictly positive induction and derive from this a new computationally meaningful formulation of the Archimedean property for real numbers. We give an example of program extraction in computable analysis and show that Archimedean induction can be used to eliminate countable choice.

1 Introduction

This paper studies a constructive logic for strictly positive inductive and coinductive definitions with a realizability interpretation that permits the extraction of programs from proofs in abstract mathematics. Particular attention is paid to a special treatment of Harrop formulas (which have trivial realizers) leading to optimized programs.

Similar work on this topic has been done in [2, 13–15, 18] and to a large extent implemented in the Minlog system [5]. Related methods of optimized program extraction can be found in [16] and in the systems Coq [9] and Nuprl [11].

Our main contribution is the extension of the realizability interpretation to inductive predicates defined by Harrop operators permitting induction over non-Harrop predicates. This enables us to exhibit wellfounded induction as a special case of strictly positive induction.

We show the usefulness of our results by a simple example in computable analysis, where we identify a new formulation of the Archimedean property as an induction principle and use it to obtain a direct and computationally meaningful proof that the inequality of approximable real numbers implies their apartness. The proof using Archimedean induction is technically and conceptually simpler than the usual proof using the Archimedean property, Markov's principle and the axiom of countable choice. The fact that Archimedean induction can be used to eliminate countable choice makes this principle potentially interesting for constructive mathematics, where one tries to avoid choice principles as far as possible [10, 17].

The theoretical results are to a large extent dual for induction and coinduction. However, our focus is on induction. For applications of coinduction see

e.g. [3, 6–8] and for a discussion of coinduction in type-theoretic systems see [1]. The Soundness Theorem (Theorem 1) shows correctness of the extracted program with respect to a domain-theoretic semantics. Correctness with respect to an operational semantics ensuring, for example, termination of programs is obtained via a Computational Adequacy Theorem which is proven in [2]. Although the logical system and realizability interpretation in [2] is slightly different to ours this does not affect adequacy in our context since programs and their semantics are type free and therefore independent of the logical system.

Our system of strictly positive inductive definitions is proof-theoretically rather strong since it permits nested and interleaved inductive definitions. The proof-theoretic strength of a related system has been analysed in [19], a discussion of this topic can be found in [2].

2 Intuitionistic Fixed Point Logic (IFP)

The language of IFP consists of *formulas* A, B defined simultaneously with *predicates* \mathcal{P}, \mathcal{Q} , and *operators* Φ, Ψ . We let X, Y, \dots range over *predicate variables*, and P, Q over *predicate constants*, each with a fixed arity, s, t range over first-order terms. There is a distinguished 0-ary predicate constant \perp for falsity which we identify with the formula $\perp()$.

$$\begin{aligned}
 \text{Formulas } \ni A, B &::= \mathcal{P}(\vec{t}) \quad (\mathcal{P} \text{ not an abstraction, } \vec{t} \text{ arity}(\mathcal{P}) \text{ many terms}) \\
 &\quad | A \wedge B \quad | A \vee B \quad | A \rightarrow B \quad | \forall x A \quad | \exists x A \\
 \text{Predicates } \ni \mathcal{P}, \mathcal{Q} &::= X \quad | P \quad | \lambda \vec{x} A \quad | \mu \Phi \quad | \nu \Phi \\
 &\quad (\text{arity}(\lambda \vec{x} A) = |\vec{x}|, \text{arity}(\mu \Phi) = \text{arity}(\nu \Phi) = \text{arity}(\Phi)) \\
 \text{Operators } \ni \Phi, \Psi &::= \lambda X \mathcal{P} \quad (\text{arity}(\lambda X \mathcal{P}) = \text{arity}(X) = \text{arity}(\mathcal{P}))
 \end{aligned}$$

The application of an operator to a predicate is defined as $(\lambda X \mathcal{P})(\mathcal{Q}) = \mathcal{P}[\mathcal{Q}/X]$. A definition $\mathcal{P} \stackrel{\text{Def}}{=} \mu \Phi$ will also be written $\mathcal{P} \stackrel{\mu}{=} \Phi(\mathcal{P})$ and if $\Phi = \lambda X \lambda \vec{x} A$ it may be written $\mathcal{P}(\vec{x}) \stackrel{\mu}{=} A[\mathcal{P}/X]$ (similarly for $\mathcal{P} \stackrel{\text{Def}}{=} \nu \Phi$). We use \equiv for equivalence, i.e., $A \equiv B \stackrel{\text{Def}}{=} A \leftrightarrow B$ and $\mathcal{P} \equiv \mathcal{Q} \stackrel{\text{Def}}{=} \forall \vec{x} (\mathcal{P}(\vec{x}) \equiv \mathcal{Q}(\vec{x}))$. The bounded quantifiers $\forall x \in \mathcal{P} \dots$ and $\exists x \in \mathcal{P} \dots$ abbreviate $\forall x (\mathcal{P}(x) \rightarrow \dots)$ and $\exists x (\mathcal{P}(x) \wedge \dots)$.

An expression (formula, predicate, operator) is *strictly positive (s.p.)* in a predicate variable X if it does not contain X free in the premise of an implication. $\lambda Y \mathcal{P}$ is *strictly positive* if \mathcal{P} is s.p. in Y . An expression is *regular* if it contains only inductive predicates $\mu \Phi$ and coinductive predicates $\nu \Phi$ where Φ is s.p.. Throughout the paper it is assumed that *all expressions are regular and all operators mentioned are strictly positive*.

The proof rules of IFP are the usual rules of intuitionistic first-order logic extended by the following rules for inductive and coinductive definitions:

$$\begin{array}{cc}
 \frac{}{\Phi(\mu \Phi) \subseteq \mu \Phi} \text{cl} & \frac{\Phi(\mathcal{P}) \subseteq \mathcal{P}}{\mu \Phi \subseteq \mathcal{P}} \text{ind} \\
 \frac{}{\nu \Phi \subseteq \Phi(\nu \Phi)} \text{cocl} & \frac{\mathcal{P} \subseteq \Phi(\mathcal{P})}{\mathcal{P} \subseteq \nu \Phi} \text{coind}
 \end{array}$$

3 Intuitionistic Fixed Point Logic for Realizers (RIFP)

The Scott domain of realizers is defined by the recursive domain equation [12]

$$D = \mathbf{Nil} + \mathbf{Lt}(D) + \mathbf{Rt}(D) + \mathbf{Pair}(D \times D) + \mathbf{F}(D \rightarrow D)$$

where $+$ denotes the separated sum, \times the Cartesian product and $D \rightarrow D$ is the continuous function space. \mathbf{Nil} , \mathbf{Lt} , \mathbf{Rt} , \mathbf{Pair} , \mathbf{F} are mnemonic labels called constructors. The elements of D are of the form \perp (the least element), \mathbf{Nil} , $\mathbf{Lt}(d)$, $\mathbf{Rt}(d)$ where $d \in D$, or $\mathbf{F}(f)$ where f is a continuous function from D to D .

Programs denote elements of D . They are defined simultaneously with *function terms* (*functions* for short), which denote continuous functions on D .

$$\begin{aligned} \text{Programs } \ni p, q &::= a, b \text{ (variables)} \mid \mathbf{Nil} \mid \mathbf{Lt}(p) \mid \mathbf{Rt}(p) \mid \mathbf{Pair}(p, q) \mid \mathbf{F}(\alpha) \\ &\quad \mid \mathbf{case}(p, \alpha, \beta) \mid \mathbf{proj}_i(p) \ (i \in \{0, 1\}) \mid \alpha p \mid \mathbf{rec}(\alpha) \\ \text{Functions } \ni \alpha, \beta &::= f, g \text{ (variables)} \mid \lambda a p \mid \mathbf{app}(p) \end{aligned}$$

Since Scott domains and continuous functions form a Cartesian closed category and the mapping $(D \rightarrow D) \ni f \mapsto \bigsqcup_n f^n(\perp) \in D$ defines a continuous fixed point operator, programs and abstractions have an obvious denotational semantics [12].

To reason formally about realizers and programs we extend IFP by a sort δ for elements of D and a sort $\delta \rightarrow \delta$ for continuous functions from D to D . The terms of sort D are the programs, the terms of sort $\delta \rightarrow \delta$ are the function terms. We further add predicate variables $\tilde{X}, \tilde{Y}, \dots$ which admit an extra argument of sort δ , and extend the notions of formula, predicate and operator as well as the rules and axioms of IFP accordingly. We add the axioms:

$$\begin{aligned} \mathbf{case}(\mathbf{Lt}(a), f, g) &= f a & \mathbf{app}(\mathbf{F}(f)) a &= f a \\ \mathbf{case}(\mathbf{Rt}(a), f, g) &= g a & (\lambda a p) b &= p[b/a] \quad (\text{for every prog. } p) \\ \mathbf{proj}_i(\mathbf{Pair}(a_0, a_1)) &= a_i & \mathbf{rec}(f) &= f(\mathbf{rec}(f)) \end{aligned}$$

The resulting system is called *Intuitionistic fixed point logic for realizers* (RIFP).

4 Realizability and Soundness

An expression is *Harrop* if it contains no disjunction or free predicate variable at a s.p. position, it is *non-computational* (*nc*) if it contains no disjunction or free predicate variable at all. Hence, nc-expressions are Harrop. We assume that to every predicate variable X there is assigned, in a one-to-one fashion, a predicate variable \tilde{X} with one extra argument place for realizers. We define for every expression \square an expression $\mathbf{R}(\square)$ (the realizability interpretation of \square) and, if \square is Harrop, an expression $\mathbf{H}(\square)$ (a simplified realizability interpretation), more precisely, for a

- formula A a predicate $\mathbf{R}(A)$ with one argument for realizers,
- predicate \mathcal{P} a predicate $\mathbf{R}(\mathcal{P})$ with an extra argument for realizers,
- non-Harrop operator Φ an operator $\mathbf{R}(\Phi)$ with an extra argument for realizers,
- Harrop formula A a formula $\mathbf{H}(A)$,
- Harrop predicate \mathcal{P} a predicate $\mathbf{H}(\mathcal{P})$ of the same arity,
- Harrop operator Φ an operator $\mathbf{H}(\Phi)$ of the same arity.

We sometimes write $a \mathbf{r} A$ for $\mathbf{R}(A)(a)$ and $\mathbf{r} A$ for $\exists a a \mathbf{r} A$. Set $\mathbf{H}_X(\mathcal{P}) \stackrel{\text{Def}}{=} (\mathbf{H}(\mathcal{P}[P/X]))[X/P]$ where P is a fresh predicate constant.

$$\begin{aligned}
 a \mathbf{r} A &= \mathbf{H}(A) \wedge a = \mathbf{Nil} \quad (A \text{ Harrop}) \\
 \mathbf{R}(\mathcal{P}) &= \lambda(\vec{x}, a) (\mathbf{H}(\mathcal{P}) \wedge a = \mathbf{Nil}) \quad (\mathcal{P} \text{ Harrop}) \\
 \text{Otherwise} & \\
 a \mathbf{r} \mathcal{P}(\vec{t}) &= \mathbf{R}(\mathcal{P})(\vec{t}, a) & \mathbf{H}(\mathcal{P}(\vec{t})) &= \mathbf{H}(\mathcal{P})(\vec{t}) \\
 c \mathbf{r} (A \wedge B) &= \exists a, b (c = \mathbf{Pair}(a, b) \wedge a \mathbf{r} A \wedge b \mathbf{r} B) & \mathbf{H}(A \wedge B) &= \mathbf{H}(A) \wedge \mathbf{H}(B) \\
 & \quad (\text{neither } A \text{ nor } B \text{ Harrop}) \\
 a \mathbf{r} (A \wedge B) &= a \mathbf{r} A \wedge \mathbf{H}(B) \quad (B \text{ Harrop}) \\
 b \mathbf{r} (A \wedge B) &= \mathbf{H}(A) \wedge b \mathbf{r} B \quad (A \text{ Harrop}) \\
 c \mathbf{r} (A \vee B) &= \exists a (c = \mathbf{Lt}(a) \wedge a \mathbf{r} A \vee c = \mathbf{Rt}(a) \wedge a \mathbf{r} B) \\
 c \mathbf{r} (A \rightarrow B) &= \exists f (c = \mathbf{F}(f) \wedge \forall a (a \mathbf{r} A \rightarrow (f a) \mathbf{r} B)) & \mathbf{H}(A \rightarrow B) &= \mathbf{r} A \rightarrow \mathbf{H}(B) \\
 & \quad (\text{neither } A \text{ nor } B \text{ Harrop}) \\
 b \mathbf{r} (A \rightarrow B) &= \mathbf{H}(A) \rightarrow b \mathbf{r} B \quad (A \text{ Harrop}) \\
 a \mathbf{r} \diamond x A &= \diamond x (a \mathbf{r} A) \quad (\diamond \in \{\forall, \exists\}) & \mathbf{H}(\diamond x A) &= \diamond x \mathbf{H}(A) \\
 \mathbf{R}(X) &= \tilde{X} & \mathbf{H}(P) &= P \\
 \mathbf{R}(\diamond \Phi) &= \diamond \mathbf{R}(\Phi) \quad (\diamond \in \{\nu, \mu\}) & \mathbf{H}(\diamond \Phi) &= \diamond \mathbf{H}(\Phi) \\
 \mathbf{R}(\lambda \vec{x} A) &= \lambda \vec{x} \mathbf{R}(A) \quad (= \lambda(\vec{x}, a) a \mathbf{r} A) & \mathbf{H}(\lambda \vec{x} A) &= \lambda \vec{x} \mathbf{H}(A) \\
 \mathbf{R}(\lambda X \mathcal{P}) &= \lambda \tilde{X} \mathbf{R}(\mathcal{P}) & \mathbf{H}(\lambda X \mathcal{P}) &= \lambda X \mathbf{H}_X(\mathcal{P})
 \end{aligned}$$

It is clear that the operations \mathbf{R} and \mathbf{H} preserve regularity and strict positivity, hence realizability is well-defined. Furthermore, if A is Harrop, then $\mathbf{H}(A) \equiv \mathbf{r} A$, and if A is nc, then $\mathbf{H}(A) = A$.

The Soundness Theorem below is restricted to proofs where every instance of induction or coinduction satisfies the condition that either Φ and \mathcal{P} are both Harrop or both non-Harrop or Φ is Harrop and simple (see below) and \mathcal{P} is non-Harrop. We call such proofs *admissible*. This is not a severe restriction since in all practical applications proofs turn out to be admissible. An expression \square is *X-simple* if no sub-expression of \square of the form $\mu \Phi$ or $\nu \Phi$ contains X free. A (strictly positive) operator $\lambda X \mathcal{P}$ is *simple* if \mathcal{P} is *X-simple*. We conjecture that the Soundness Theorem also holds without the admissibility assumption. From now on we tacitly assume that all proofs are admissible.

We write pq for $\mathbf{app}(p)q$, $\lambda a p$ for $\mathbf{F}(\lambda a p)$, and $p \circ q$ for $\lambda a (p(q a))$.

Theorem 1 (Soundness). *Let Γ be a set of Harrop formulas and Δ a set of formulas that are not Harrop. Then, from an admissible IFP-proof of a formula A from the assumptions Γ, Δ one can extract a program p with $\text{FV}(p) \subseteq \vec{u}$ such that $p \mathbf{r} A$ is RIFP-provable from the assumptions $\mathbf{H}(\Gamma)$ and $\vec{u} \mathbf{r} \Delta$.*

Proof. By induction on derivations one shows simultaneously

- (1) If $\Gamma, \Delta \vdash_{\text{IFP}} A$ where A is not Harrop, then $\mathbf{H}(\Gamma), \vec{u} \mathbf{r} \Delta \vdash_{\text{RIFP}} p \mathbf{r} A$ for some program p with $\text{FV}(p) \subseteq \vec{u}$.
- (2) If $\Gamma, \Delta \vdash_{\text{IFP}} A$ where A is Harrop, then $\mathbf{H}(\Gamma), \vec{u} \mathbf{r} \Delta \vdash_{\text{RIFP}} \mathbf{H}(A)$.

The logical rules are easy. To see that the rules for induction and coinduction are realizable in all admissible cases one needs to do a case distinction on whether or not the operator Φ and the predicate \mathcal{P} are Harrop. Note that the case that Φ is non-Harrop but \mathcal{P} is Harrop is excluded due to the admissibility condition.

W.l.o.g. we assume that Φ is not constant; i.e., if $\Phi = \lambda X \mathcal{Q}$ then X does occur freely (and hence strictly positively) in \mathcal{Q} . We first look at induction

$$\frac{\Phi(\mathcal{P}) \subseteq \mathcal{P}}{\mu \Phi \subseteq \mathcal{P}} \text{ ind}$$

If Φ and \mathcal{P} are both not Harrop, then the premise of induction gives us a realizer s of $\Phi(\mathcal{P}) \subseteq \mathcal{P}$, i.e., $\forall b, \vec{x} (b \mathbf{r} \Phi(\mathcal{P})(\vec{x}) \rightarrow (s b) \mathbf{r} \mathcal{P}(\vec{x}))$. Using the notation $g^{-1} \circ \mathcal{Q} \stackrel{\text{Def}}{=} \lambda(\vec{x}, b) \mathcal{Q}(\vec{x}, g b)$ and the easily provable fact that $\mathbf{R}(\Phi(\mathcal{P})) = \mathbf{R}(\Phi)(\mathbf{R}(\mathcal{P}))$ if Φ and \mathcal{P} are both Harrop, this can be written as

- (1) $\mathbf{R}(\Phi)(\mathbf{R}(\mathcal{P})) \subseteq s^{-1} \circ \mathbf{R}(\mathcal{P})$.

By recursion on the build-up of Φ one can define a closed term \mathbf{map} realizing the formula $X \subseteq Y \rightarrow \Phi(X) \subseteq \Phi(Y)$. Using the notation above, this means that for all g, \tilde{X}, \tilde{Y} , we have $\tilde{X} \subseteq g^{-1} \circ \tilde{Y} \rightarrow \mathbf{R}(\Phi)(\tilde{X}) \subseteq (\mathbf{map} g)^{-1} \circ \mathbf{R}(\Phi)(\tilde{Y})$, in particular for $\tilde{X} \stackrel{\text{Def}}{=} g^{-1} \circ \tilde{Y}$ one has

- (2) $\mathbf{R}(\Phi)(g^{-1} \circ \tilde{Y}) \subseteq (\mathbf{map} g)^{-1} \circ \mathbf{R}(\Phi)(\tilde{Y})$.

We need a realizer of $\mu \Phi \subseteq \mathcal{P}$. Since $\mu \Phi$ is not Harrop, the realizer f must satisfy $\forall \vec{x} \forall b (\mu \mathbf{R}(\Phi))(\vec{x}, b) \rightarrow (f b) \mathbf{r} \mathcal{P}(\vec{x})$, i.e., $\mu \mathbf{R}(\Phi) \subseteq f^{-1} \circ \mathbf{R}(\mathcal{P})$. We attempt to prove this by induction (with a yet unknown f). Therefore, we try to show $\mathbf{R}(\Phi)(f^{-1} \circ \mathbf{R}(\mathcal{P})) \subseteq f^{-1} \circ \mathbf{R}(\mathcal{P})$.

Using (1) and (2) with $g \stackrel{\text{Def}}{=} f$ and $\tilde{Y} \stackrel{\text{Def}}{=} \mathbf{R}(\mathcal{P})$ we obtain

$$\begin{aligned} \mathbf{R}(\Phi)(f^{-1} \circ \mathbf{R}(\mathcal{P})) &\subseteq (\mathbf{map} f)^{-1} \circ \mathbf{R}(\Phi)(\mathbf{R}(\mathcal{P})) \\ &\subseteq (\mathbf{map} f)^{-1} \circ (s^{-1} \circ \mathbf{R}(\mathcal{P})) \\ &\equiv (s \circ \mathbf{map} f)^{-1} \circ \mathbf{R}(\mathcal{P}) \end{aligned}$$

Hence if we define f recursively by $f = s \circ \mathbf{map} f$ we are done.

The case that Φ and \mathcal{P} are both Harrop is easy, since then premise and conclusion of the induction rule are Harrop and therefore the realizability interpretation

of the premise is $\mathbf{H}(\Phi(\mathcal{P})) \subseteq \mathbf{H}(\mathcal{P})$ and that of the conclusion $\mu \mathbf{H}(\Phi) \subseteq \mathbf{H}(\mathcal{P})$. Since one can prove by structural induction that $\mathbf{H}(\Phi(\mathcal{P}))$ is the same as $\mathbf{H}(\Phi)(\mathbf{H}(\mathcal{P}))$ and $\mathbf{H}(\Phi)$ inherits strict positivity from Φ , we obtain an instance of the induction rule for the $\mathbf{H}(\Phi)$ and the $\mathbf{H}(\mathcal{P})$.

The last case to consider is that Φ is Harrop and simple, and \mathcal{P} is not Harrop. The premise of induction gives us a realizer s of $\Phi(\mathcal{P}) \subseteq \mathcal{P}$ i.e., since $\Phi(\mathcal{P})$ is not Harrop, $\forall b, \vec{x} (b \mathbf{r} \Phi(\mathcal{P})(\vec{x}) \rightarrow (sb) \mathbf{r} \mathcal{P}(\vec{x}))$. Using the notation $\mathcal{P}_a \stackrel{\text{Def}}{=} \lambda \vec{x} (a \mathbf{r} \mathcal{P}(\vec{x}))$ this can be written as $\forall b (\Phi(\mathcal{P})_b \subseteq \mathcal{P}_{sb})$. We need a realizer a of $\mu \Phi \subseteq \mathcal{P}$. Since $\mu \Phi$ is Harrop, this means that a must satisfy $\forall \vec{x} (\mu \mathbf{H}(\Phi)(\vec{x}) \rightarrow a \mathbf{r} \mathcal{P}(\vec{x}))$, i.e., $\mu \mathbf{H}(\Phi) \subseteq \mathcal{P}_a$. We attempt to prove this by induction (with a yet unknown a). Therefore, we show $\mathbf{H}(\Phi)(\mathcal{P}_a) \subseteq \mathcal{P}_a$. By recursion on the build-up of Φ one can construct a closed (recursion-free) term ψ such that $\mathbf{H}(\Phi)(\mathcal{P}_b) \subseteq \Phi(\mathcal{P})_{\psi(b)}$ for all b . It follows that $\mathbf{H}(\Phi)(\mathcal{P}_a) \subseteq \Phi(\mathcal{P})_{\psi(a)} \subseteq \mathcal{P}_{s\psi(a)}$. Hence, if a is defined recursively as $a = s\psi(a)$, we are done.

For coinduction the proof is completely dual in the first two cases (Φ, \mathcal{P} both non-Harrop or both Harrop) and similar to induction in the third case (Φ Harrop, \mathcal{P} non-Harrop).

5 Wellfounded Induction

In the following we let upper-case Roman letters range over arbitrary predicates.

The usual formulation of induction over a wellfounded relation $<$ is

$$\frac{\mathbf{Prog}_{<}(P)}{\forall x P(x)} \mathbf{WfI}(<)$$

where $\mathbf{Prog}_{<}(P) \stackrel{\text{Def}}{=} \forall x (\forall y (y < x \rightarrow P(y)) \rightarrow P(x))$. In order to be computationally meaningful we formulate this principle in a relativized form where we require the relation $<$ to be wellfounded only on a given predicate A (which is typically non-Harrop). This is expressed by the condition that A is contained in the wellfounded (or accessible) part of $<$. Hence *Wellfounded induction* is the principle

$$\frac{A \subseteq \mathbf{Acc}_{<} \quad \mathbf{Prog}_{<,A}(P)}{A \subseteq P} \mathbf{WfI}(<, A)$$

where $\mathbf{Acc}_{<}(x) \stackrel{\mu}{=} \forall y < x \mathbf{Acc}_{<}(y)$

$$\mathbf{Prog}_{<,A}(P) \stackrel{\text{Def}}{=} \forall x \in A (\forall y \in A (y < x \rightarrow P(y)) \rightarrow P(x))$$

Proposition 1. *Wellfounded induction follows from admissible s.p. induction. If P is not Harrop, then the extracted program is the least fixed point operator; i.e., if f realizes $\mathbf{Prog}_{<,A}(P)$, then the least fixed point of f realizes $A \subseteq P$.*

Proof. Set $\Phi \stackrel{\text{Def}}{=} \lambda X \lambda x \forall y < x X(y)$ which is a simple Harrop operator. Then $\mathbf{Acc}_< = \mu \Phi$ and the assumed progressivity, $\mathbf{Prog}_{<,A}(P)$, is equivalent to $\Phi(A \Rightarrow P) \subseteq (A \Rightarrow P)$, where $A \Rightarrow P \stackrel{\text{Def}}{=} \lambda x (A(x) \rightarrow P(x))$. Hence $\mu \Phi \subseteq (A \Rightarrow P)$, and therefore, by the assumption $A \subseteq \mathbf{Acc}_<$, $A \subseteq (A \Rightarrow P)$. It follows $A \subseteq P$.

Now let P be non-Harrop and let f realize $\mathbf{Prog}_{<,A}(P)$. By the proof of the Soundness Theorem, a defined recursively as $a = f \psi(a)$ realizes $\mathbf{Acc}_< \subseteq (A \Rightarrow P)$ where ψ satisfies $\mathbf{H}(\Phi)(Q_b) \subseteq \Phi(Q)_{\psi(b)}$. Unfolding this formula one sees that ψ is the identity. Therefore, the least fixed point of f realizes $\mathbf{Acc}_< \subseteq (A \Rightarrow P)$ and therefore, as can be easily seen, also $A \subseteq P$.

6 Archimedean Induction

We give an application of wellfounded induction and hence inductive definitions in computable analysis. We let the variables x, y, \dots range over abstract reals. We assume that the basic arithmetic operations $(0, 1, +, *, | \cdot |, \dots)$ and relations $(=, <, \leq, \dots)$ are given as function and predicate symbols and we will freely use any true arithmetic nc-properties of them. Hence $x = y$, $x < y$, $x \leq y$ are atomic formulas. We write $x \neq y$ as a shorthand for $\neg(x = y)$, i.e. $x = y \rightarrow \perp$. All these formulas are nc.

Natural numbers are inductively defined as a subset of the real numbers by

$$\mathbf{N}(x) \stackrel{\mu}{=} (x = 0 \vee \mathbf{N}(x - 1))$$

(i.e. $\mathbf{N} \stackrel{\text{Def}}{=} \mu(\lambda X \lambda x (x = 0 \vee X(x - 1)))$). The formula $\mathbf{N}(x)$ is not Harrop since it contains a disjunction at a strictly positive position. Integers (\mathbf{Z}) and rational numbers (\mathbf{Q}) are defined from the natural numbers in the usual way.

Cauchy reals are represented as real numbers satisfying the predicate

$$\mathbf{A}(x) \stackrel{\text{Def}}{=} \forall n \in \mathbf{N} \exists q \in \mathbf{Q} |x - q| \leq 2^{-n}$$

The realizability interpretation of the predicate \mathbf{N} is

$$a \mathbf{r} \mathbf{N}(x) \stackrel{\mu}{=} a = \mathbf{Lt}(\mathbf{Nil}) \wedge x = 0 \vee \exists b (a = \mathbf{Rt}(b) \wedge b \mathbf{r} \mathbf{N}(x - 1))$$

Hence realizers of the elements of \mathbf{N} are numerals $\mathbf{Rt}^n(\mathbf{Lt}(\mathbf{Nil}))$. We write $\mathbf{0}$ for $\mathbf{Lt}(\mathbf{Nil})$ and $\mathbf{S}(a)$ for $\mathbf{Rt}(a)$.¹ A realizer of $\mathbf{A}(x)$ is a function f such that

$$\forall n, a (a \mathbf{r} \mathbf{N}(n) \rightarrow \exists q \in \mathbf{Q} (f(a) \mathbf{r} \mathbf{Q}(q) \wedge |x - q| \leq 2^{-n}))$$

Hence, essentially, f is a sequence of (representations of) rational numbers converging quickly to x . To work in the model of Cauchy reals one simply

¹ The *binary* representation of natural numbers is obtained by defining the (same) set of natural numbers as $\mathbf{N}(x) \stackrel{\mu}{=} \exists y ((y = 0 \vee y > 0 \wedge \mathbf{N}(y)) \wedge \exists i \in \{0, 1\} (x = 2y + i))$.

relativizes all quantifiers to \mathbf{A} . However, we refrain from doing so since there are principles (such as Archimedean induction below) which are valid without such relativization.

The usual apartness relation between real numbers is defined by

$$x \# y \stackrel{\text{Def}}{=} \exists k \in \mathbf{N} |x - y| \geq 2^{-k}$$

Clearly, $x \# 0$ implies $x \neq 0$ but the converse implication only holds with extra assumption on x , for example $x \in \mathbf{A}$. We are interested in a proof of the converse implication that permits the extraction of a program, possibly admitting classically true assumptions as long as they are Harrop or realizable and therefore do not spoil program extraction.

We first prove the implication $x \neq y \rightarrow x \# y$ relativized to $x, y \in \mathbf{A}$ with the help of a Harrop formulation of the Archimedean property, Markov's principle and the countable axiom of choice:

Archimedean property (AP): $(\forall n \in \mathbf{N} |x| < 2^{-n}) \rightarrow x = 0$.

Markov's principle (MP):

$$\forall n \in \mathbf{N} (A(n) \vee \neg A(n)) \rightarrow \neg \neg \exists n \in \mathbf{N} A(n) \rightarrow \exists n \in \mathbf{N} A(n)$$

Axiom of countable choice (ACC):

$$\forall n \in \mathbf{N} \exists x A(n, x) \rightarrow \exists f \forall n \in \mathbf{N} A(n, f(n)).$$

Note that AP is a Harrop formula which is equivalent to $\mathbf{H}(\text{AP})$. MP is realized by an unbounded search operator which can be easily defined by recursion. ACC quantifies over functions, hence requires an extension of IFP, and is realized by the identity.

Proposition 2 (AP, MP, ACC). $\forall x \in \mathbf{A} (x \neq 0 \rightarrow x \# 0)$.

Proof. Assume $\mathbf{A}(x)$ and $x \neq 0$. By ACC there exists an infinite sequence of rational numbers q_k ($k \in \mathbf{N}$) such that $|x - q_k| \leq 2^{-k}$ for all $k \in \mathbf{N}$. It is impossible that $|q_{k+1}| \leq 2^{-k}$ for all $k \in \mathbf{N}$ since this would clearly imply that $|x| \leq 2^{-k}$ for all $k \in \mathbf{N}$ and therefore $x = 0$, by AP. Since $|q_{k+1}| \leq 2^{-k}$ is a decidable property of k , by MP we can find some $k \in \mathbf{N}$ with $|q_{k+1}| > 2^{-k}$. It follows that $|x| \geq 2^{-(k+1)}$.

We now introduce an alternative formulation of the Archimedean property in the form of an induction principle. This will allow us to prove the implication $x \neq 0 \rightarrow x \# 0$ for $x \in \mathbf{A}$ without using Markov's principle or countable choice and will directly yield a simple extracted program.

Archimedean induction is the rule

$$\frac{\forall x \neq 0 ((|x| \leq 3 \rightarrow P(2x)) \rightarrow P(x))}{\forall x \neq 0 P(x)} \text{ AI}$$

Of course, the number 3 can be replaced by any positive rational number and the number 2 by any rational number > 1 .

Proposition 3. *AI follows classically from AP and wellfounded induction. If P is not Harrop, then AI is realized by \mathbf{rec} .*

Proof. By the Archimedean property, for each $x \neq 0$, the sequence $|x|, |2x|, |4x|, \dots$ is unbounded, hence will eventually exceed 3. Therefore, $A \subseteq \mathbf{Acc}_{\prec}$ holds, where $A(x) \stackrel{\text{Def}}{=} x \neq 0$ and $y \prec x \stackrel{\text{Def}}{=} |x| \leq 3 \wedge y = 2x$. The premise of AI is $\mathbf{Prog}_{\prec, A}(P)$, hence $A \subseteq P$, by $\mathbf{Wfl}(\prec, A)$. By Proposition 1 the extracted realizer is **rec**.

A useful variant of Archimedean induction is its relativization to \mathbf{A} :

$$\frac{\forall x \in \mathbf{A} \setminus \{0\} ((|x| \leq 3 \rightarrow P(2x)) \rightarrow P(x))}{\forall x \in \mathbf{A} \setminus \{0\} P(x)} \text{ AIC}$$

Proposition 4. AIC follows from AI and hence is realizable.

Proof. Apply AI to the predicate $\mathbf{A} \Rightarrow P \stackrel{\text{Def}}{=} \lambda x (\mathbf{A}(x) \rightarrow P(x))$ and use the fact that \mathbf{A} is closed under doubling.

If s realizes the premise of AIC, then a realizer of the conclusion of AIC is extracted as the recursively defined function $\chi g = s g (\chi (d g))$ where $d = \lambda g \lambda n \ 2*(g(\mathbf{S}(n)))$ is the realizer extracted from the easy proof of $\mathbf{A}(x) \rightarrow \mathbf{A}(2x)$ and $2*$ implements doubling of (unary representations of) natural numbers.

Proposition 5 (AIC). $\forall x \in \mathbf{A} (x \neq 0 \rightarrow x \neq 0)$.

Proof. We show $\forall x \in \mathbf{A} \setminus \{0\} x \neq 0$ using AIC. Let $x \in \mathbf{A} \setminus \{0\}$ and assume, as induction hypothesis, $|x| \leq 3 \rightarrow 2x \neq 0$. Since $x \in \mathbf{A}$ there is $q \in \mathbf{Q}$ such that $|x - q| \leq 1$. If $|q| > 2$, then $|x| \geq 1$ and we are done. If $|q| \leq 2$, then $|x| \leq 3$ so we can apply the induction hypothesis to obtain $2x \neq 0$, which implies $x \neq 0$.

Program extraction for Proposition 5: The program extracted from above proof is

$$\varphi f = \mathbf{if} \ |f \mathbf{0}| > \mathbf{2} \ \mathbf{then} \ \mathbf{1} \ \mathbf{else} \ \mathbf{S}(\varphi (\lambda n \ \mathbf{2} * f(\mathbf{S}(n))))$$

where $|\cdot|$ and $>$ implement the absolute value function and $>$ relation on (representations of) rational numbers and $\mathbf{if} \ t \ \mathbf{then} \ p \ \mathbf{else} \ q$ stands for $\mathbf{case}(t, \lambda a \ p, \lambda a \ q)$ assuming that the Booleans are encoded as $\mathbf{Lt}(\mathbf{Nil})$ and $\mathbf{Rt}(\mathbf{Nil})$.

7 Conclusion

We presented a constructive theory of strictly positive inductive and coinductive definitions that permits (co)induction over a Harrop operator to be applied to non-Harrop predicates. This allowed us to exhibit wellfounded induction as a special case of strictly positive induction and, in turn, to give a new presentation of the Archimedean property for real numbers as a computationally meaningful induction principle. A simple example in computable analysis reveals that Archimedean induction is able to provide a new computationally meaningful

proof (Proposition 5) of a result that would normally be proven using countable choice plus Markov's principle (Proposition 2). Hence Archimedean induction allowed us to eliminate these constructively questionable principles. We leave it as an open question whether for this particular example (approximable non-zero reals are apart from 0) a computationally meaningful proof using the Archimedean property and Markov's principle alone could be given, but we conjecture that this is not the case. Archimedean induction (even with Markov's principle) is weaker than countable choice since the computable reals validate the former while, classically, they do not validate the latter. Regarding the constructive status of Archimedean induction it must be noted that its reduction to wellfounded induction (Proposition 3) uses classical logic but no choice. Hence this achieves only classical elimination of choice. However, there might be an independent constructive justification of Archimedean induction. At least computationally this principle *is* justified through its realizability interpretation.

It is straightforward to extend our results to generally positive inductive and coinductive definitions. There is even a possibility of extending this to a system of higher-order logic and monotone inductive and coinductive definition as presented in [4].

Acknowledgments. This work was supported by the Marie Curie International Research Staff Exchange Schemes *Computable Analysis* (PIRSES-GA-2011-294962) and *Correctness by Construction* (FP7-PEOPLE-2013-IRSES-612638) as well as the Marie Curie RISE project *Computing with Infinite Data* (H2020-MSCA-RISE-2016-731143) and the EPSRC Doctoral Training Grant No. 1818640.

References

1. Abel, A., Pientka, B., Thibodeau, D., Setzer, A.: Copatterns: programming infinite structures by observations. In: 40th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2013), pp. 27–38 (2013)
2. Berger, U.: Realisability for induction and coinduction with applications to constructive analysis. *J. Univ. Comput. Sci.* **16**(18), 2535–2555 (2010)
3. Berger, U.: From coinductive proofs to exact real arithmetic: theory and applications. *Logical Methods Comput. Sci.* **7**(1), 1–24 (2011)
4. Berger, U., Hou, T.: A realizability interpretation of Church's simple theory of types. *Math. Struct. Comput. Sci.* **27**, 1–22 (2016)
5. Berger, U., Miyamoto, K., Schwichtenberg, H., Seisenberger, M.: Minlog - a tool for program extraction supporting algebras and coalgebras. In: Corradini, A., Klin, B., Cirstea, C. (eds.) CALCO 2011. LNCS, vol. 6859, pp. 393–399. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22944-2_29
6. Berger, U., Miyamoto, K., Schwichtenberg, H., Tsuiki, H.: Logic for Gray-code computation. In: *Concepts of Proof in Mathematics, Philosophy, and Computer Science*. De Gruyter (2016)
7. Berger, U., Seisenberger, M.: Proofs, programs, processes. *Theory Comput. Syst.* **51**(3), 213–329 (2012)
8. Berger, U., Spreen, D.: A coinductive approach to computing with compact sets. *J. Logic Anal.* **8**, 1–35 (2016)

9. Bertot, Y., Castéran, P.: *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer, Heidelberg (2004). <https://doi.org/10.1007/978-3-662-07964-5>
10. Bridges, D., Richman, F., Schuster, P.: Linear independence without choice. *Ann. Pure Appl. Logic* **101**(1), 95–102 (1999)
11. Constable, R.: *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, New Jersey (1986)
12. Gierz, G., Hofmann, K., Keimel, K., Lawson, J., Mislove, M., Scott, D.: *Continuous Lattices and Domains*. *Encyclopedia of Mathematics and its Applications*, vol. 93. CUP (2003)
13. Miranda-Perea, F.: Realizability for monotone clausal (co)inductive definitions. *Electr. Notes Theoret. Comput. Sci.* **123**, 179–193 (2005)
14. Miyamoto, K.: *Program extraction from coinductive proofs and its application to exact real arithmetic*. Ph.D. thesis, Mathematisches Institut LMU, Munich (1993)
15. Miyamoto, K., Nordvall Forsberg, F., Schwichtenberg, H.: Program extraction from nested definitions. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) *ITP 2013*. LNCS, vol. 7998, pp. 370–385. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39634-2_27
16. Parigot, M.: Recursive programming with proofs. *Theor. Comput. Sci.* **94**(2), 335–356 (1992)
17. Richman, F.: The fundamental theorem of algebra: a constructive development without choice. *Pacific J. Math.* **196**(1), 213–230 (2000)
18. Tatsuta, M.: Realizability of monotone coinductive definitions and its application to program synthesis. In: Jeuring, J. (ed.) *MPC 1998*. LNCS, vol. 1422, pp. 338–364. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054298>
19. Tupailo, S.: On the intuitionistic strength of monotone inductive definitions. *J. Symb. Logic* **69**(3), 790–798 (2004)



Computing Tree Width: From Theory to Practice and Back

Sebastian Berndt(✉)

Department of Computer Science, Kiel University, Kiel, Germany
seb@informatik.uni-kiel.de

Abstract. While the theoretical aspects concerning the computation of tree width – one of the most important graph parameters – are well understood, it is not clear how it can be computed practically. As tree width has a wide range of applications, e.g. in bioinformatics or artificial intelligence, this lack of understanding hinders the applicability of many important algorithms in the real world. The Parameterized Algorithms and Computational Experiments (PACE) challenge therefore chose the computation of tree width as one of its challenge problems in 2016 and again in 2017. In 2016, Hisao Tamaki (Meiji University) presented a new algorithm that outperformed the other approaches (including SAT solvers and branch-and-bound) by far. An implementation of Tamaki’s algorithm allowed Larisch (King-Abdullah University of Science and Engineering) and Salfelder (University of Leeds) to solve over 50% of the test suite of PACE 2017 (containing graphs with over 3500 nodes) in under six seconds (and the remaining 50% in under 30 min). Before PACE 2016, no algorithm was known to compute tree width on graphs with about 100 nodes. As a wide range of parameterized algorithms require the computation of a tree decomposition as a first step, this breakthrough result allows practical implementations of these algorithms for the first time.

This work starts with a gentle introduction to tree width and its use in parameterized complexity, followed by an algorithmic approach for the exact computation of the tree width of a graph, based on a variant of the well-studied cops-and-robber game. Finally, we present a streamlined version of Tamaki’s algorithms due to Bannach and Berndt based on this game.

Keywords: Tree width · Algorithms · Experimental evaluation
Graph searching · Parameterized complexity

1 Talk Summary

1.1 Introducing Tree Width

Consider your favorite optimization problem on graphs. With high probability, it is easy, i. e. solvable in polynomial time, if restricted to trees. Let us consider

the MAXIMUM INDEPENDENT SET problem, where we want to find a set of vertices $V' \subseteq V(G)$ of maximum cardinality such that $\{u, v\} \notin E(G)$ for all $u, v \in V'$. This is one of the classical NP-hard problems, but solving it on trees is fairly simple. Root the tree T at some arbitrary vertex $r \in V(T)$ to obtain the rooted tree T^r . For a vertex $v \in V(T)$, let $\text{ch}(v)$ be the children of v in T^r and let $T^r[v]$ be the set of descendants of v in T^r . Furthermore, let $I^+[v]$ be the size of a maximal independent set of $T^r[v]$ that contains v and $I^-[v]$ be the size of a maximal independent set of $T^r[v]$ that does not contain v . For all leafs v of T^r , we have $I^+[v] = 1$ and $I^-[v] = 0$. If v is an inner node, we clearly have

$$I^+[v] = 1 + \sum_{w \in \text{ch}(v)} I^-[w],$$

$$I^-[v] = \sum_{w \in \text{ch}(v)} \max\{I^+[w], I^-[w]\}.$$

As $\max\{I^+[r], I^-[r]\}$ is the size of a maximal independent set in $T^r[r] = V(T)$, we can conclude that this problem is indeed solvable in linear time.

Now suppose that we add an edge $\{u, w\}$ to the tree T and obtain a graph G . Intuitively, the problem should not become much harder on G : After all, the graph G is just a tree and a single edge. The simple observation that a maximal independent set can contain u , or w , or none of them allows us to reuse our previous dynamic program with simple adaptations. Generalizing this principle, if G is a graph consisting of a tree and an additional set of edges E' , we can compute the size of a maximal independent set in time $\mathcal{O}(2^{2|E'|} \cdot |V(G)|)$ by trying out all $2^{2|E'|}$ independent sets in E' . Note that some of the edges $\{u, v\}, \{u', v'\}$ in E' may be independent of each other in the sense that the inclusion of u into the maximal independent set does not tell us whether u', v' or none of them need to be part of the maximal independent set. See e. g. the edges $\{d, e\}$ and $\{f, h\}$ in Fig. 1(a).

The concept of the *tree width* of a graph G captures the idea that a graph is tree-like and also integrates our previous discussion on independent edges. A *tree decomposition* of a graph¹ $G = (V, E)$ is a pair (\mathcal{T}, ι) such that \mathcal{T} is a tree and ι maps nodes of \mathcal{T} to subsets of V . These subsets are called *bags*. Furthermore, such a tree decomposition must have the following properties: (i) for every vertex $v \in V$, there is a node X in \mathcal{T} such that $v \in \iota(X)$, (ii) for every edge $u, v \in E$, there is a node X in \mathcal{T} such that $\{u, v\} \subseteq \iota(X)$, and (iii) for every vertex $v \in V$, the set $\{X \mid v \in \iota(X)\}$ is connected in \mathcal{T} . The *width* of a tree decomposition (\mathcal{T}, ι) is the maximum size of one of its bags minus one. As the width corresponds to the complexity of such a decomposition, we want to minimize this quantity. Hence, the *tree width* $\text{tw}(G)$ of a graph G is the minimum width over all tree decompositions of G . See Fig. 1 for an example.

The concept of tree decompositions has been used for a wide number of different applications. It is one of the fundamental tools in parameterized complexity

¹ All graphs in this work are undirected, unless stated otherwise.

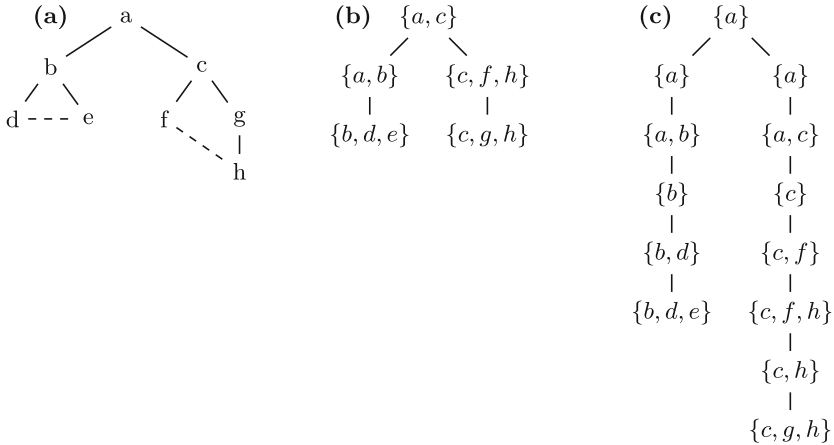


Fig. 1. Two tree decompositions of an undirected graph $G = (V, E)$ shown at (a). The decomposition in (b) shows $\text{tw}(G) \leq 2$ and the expanded one in (c) gives more intuition on the construction.

(see e. g. [6,9–11]), but also found applications in bioinformatics (see e. g. [15]) or artificial intelligence (see e. g. [12]).

Given a graph G on n nodes, one can test in time $f(k) \cdot n$ whether $\text{tw}(G) \leq k$ by the celebrated algorithm of Bodlaender [3]. Hence, this problem is fixed-parameter-tractable (and belongs to the class FPT), but the algorithm is not useful in practice, as the function $f(k)$ is quite large [4]. A wide number of FPT-algorithms compute a tree decomposition as a first step. Hence, any practical implementation of these algorithms relies on a practical algorithm for tree width. See also [2] for a more detailed discussion on this. Finding a practically fast algorithm for tree width – like modern algorithms for the SAT problem – is thus still an open and intriguing problem. Fortunately, due to the Parameterized Algorithms and Computational Experiments (PACE) challenge, huge steps toward such an algorithm were made. The aim of the PACE challenge, initiated in 2016, is the development of practically FPT algorithms [7,8]. The winning algorithm of PACE 2016, due to Hisao Tamaki, was in fact so successful, that *all* participants in PACE 2017 used a variant of this algorithm. The winning implementation of 2017 by Larisch and Salfelder “is basically Hisao Tamakis implementation” [13]. A variant of this algorithm is described in [17]. In the remainder of this paper, we will give an simplified presentation of the core of Tamaki’s algorithm. This presentation is due to Bannach and Berndt [1].

1.2 An Algorithmic Approach: Cops-and-Robber

In contrast to other graph parameters, the definition of tree width does not give immediate rise to an exponential-time algorithm: If one tries to find a tree decomposition with width k , the number of possible bags is n^{k+1} , and the number

of possible trees on these bags is thus $(n^{k+1})^{n^{k+1}/2}$, giving a super-exponential running time. Nevertheless, there are alternative characterizations of tree width that allow the design of practical exponential-time algorithms. We will take a closer look at one of them, namely the *cops-and-robber game*.

In the cops-and-robber game (sometimes also called searchers-and-fugitive game or graph-searching game), two players – the *cops* and the *robber* – play a game on the vertices of a graph $G = (V, E)$. The goal of the cops is to catch the robber, while the robber tries to avoid this. In the first turn, the cops player choose k vertices to put the cops on. Then the robber player chooses some vertex to position the robber. In each subsequent turn, the cops player announces that he moves $\ell \leq k$ cops from their current position v_1, \dots, v_ℓ to new vertices v'_1, \dots, v'_ℓ . The cops are then removed from v_1, \dots, v_ℓ . Before the cops are put on v'_1, \dots, v'_ℓ , the robber player may move the robber along the edges of the graph, but is not allowed to visit vertices currently occupied by a cop (hence, he may now visit v_1, \dots, v_ℓ , as those are currently not occupied). After the movement of the robber, the cops are put on v'_1, \dots, v'_ℓ . The cops player wins, if he places a cop on the vertex occupied by the robber player. The minimum number of cops to win this game on the graph G is denoted as $vs(G)$. Figure 2 depicts the first turns of such a game.

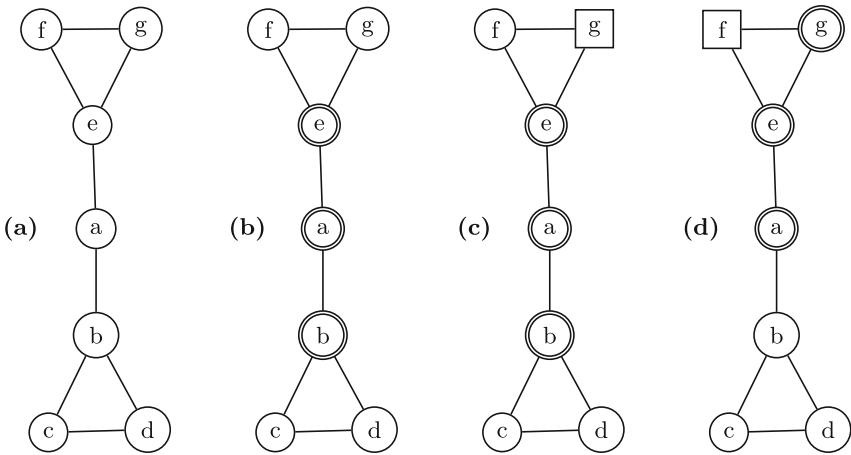


Fig. 2. A possible game on the graph depicted in (a). First, the cops player (depicted by double circles) choose vertices a, b and e in (b). Then, the robber player (depicted as rectangle) chooses vertex g in (c). The cop player announces that he moves a cop from b to g and the robber player moves the robber to f in (d). In order to win, the cops player will now move the cop from a to f .

The close connection between this game and the tree width of G was already observed by Seymour and Thomas who proved that $vs(G) = tw(G) + 1$ [14]. We will give some intuition behind this relation. An important aspect of this game

is that if $vs(G) = k$, there is a *monotone* winning strategy for the k cops, i. e. the area where the robber may move is monotonically decreasing. This non-trivial fact was also shown in [14].

- If we know that $vs(G) = k + 1$, the cops player can catch the robber with $k + 1$ cops in a monotone way. Intuitively, the positions occupied by the cops correspond to the bags of the tree decomposition. This is a valid tree decomposition of width k , as
 - (i) each vertex must at some point be occupied by some cop (otherwise, the robber would simply stay on this vertex);
 - (ii) for each edge, there must be a situation, where both endpoints of the edge are occupied by cops (otherwise, the robber could alternate between these endpoints);
 - (iii) the connectedness property of the tree decomposition is guaranteed due to the monotonicity of the strategy (otherwise, if the bags containing some vertex v were not connected, the robber may “escape” through v , which would violate the monotonicity).
- On the other hand, if we are given a tree decomposition (\mathcal{T}, ι) of width k , this corresponds to a valid monotone winning strategy for $k + 1$ cops. Consider some bag $X \in \mathcal{T}$ in the tree decomposition, which we will treat as root. The cops player puts its k cops on $\iota(X)$. The robber player now decides the position v of the robber. The cops player will now determine a child Y of X , such that v occurs in some bag in the tree rooted at Y . Such a child must always exist, as each vertex must occur in some bag. As $v \notin \iota(X)$ (otherwise, the game would be over now), the connectedness property of the tree decomposition guarantees that such a child Y is unique. The cops player then announces that the cops in $\iota(X) \cap \iota(Y)$ stay on their positions and that the cops in $\iota(X) \setminus \iota(Y)$ move to $\iota(Y) \setminus \iota(X)$. The game then goes on inductively until a leaf is reached, where the robber is captured.

For a formal proof of this relation, see e. g. [14].

This connection gives us a simple $\mathcal{O}(n^{k+2})$ algorithm to test whether $tw(G) \leq k$. Construct the so called *arena graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is directed. Each vertex (C, H, r) in \mathcal{V} has three components: (i) a set C of current positions of the cops, (ii) a set H of future positions of the cops (i. e. these positions are not currently occupied by cops, but will be at the end of the turn), and (iii) a vertex $r \in V(G)$ determining the position of the robber or the value \perp , if the robber is not yet positioned.

Consider the game in Fig. 2. The situation in Fig. 2(c) would be described as $(C = \{a, b, e\}, H = \emptyset, r = g)$. As the cops player announces that he moves a cop from b to g , this situation is described as $(C = \{a, e\}, H = \{g\}, r = g)$, as the cop is removed from b . The robber player now moves the robber to f and the resulting situation is $(C = \{a, e\}, H = \{g\}, r = f)$. Finally, the cop is put on g and the resulting situation of Fig. 2(d) would be $(C = \{a, e, g\}, H = \emptyset, r = f)$.

As $|C \cup H| \leq k + 1$, there are at most n^{k+2} such nodes in \mathcal{V} . The *start vertices* are the vertices having the form (C, \emptyset, \perp) for all $C \subseteq V(G)$ with $|C| \leq k + 1$.

These vertices correspond to situations, where the cops are positioned in the first turn, but the robber is not yet placed. The *final vertices* are of the form (C, \emptyset, r) for all $C \subseteq V(G)$ with $|C| \leq k+1$ and $r \in C$. These vertices correspond to situations, where the robber was captured.

For each vertex (C, \emptyset, r) , we add a *cops edge* to each vertex (C', H, r) , where $H \subseteq V(G)$ with $|H| \leq |C|$ and $C' \subseteq C$ with $|C'| \leq |C| - |H|$. Such an edge corresponds to the announced move of the cops player. It also allows us to forget some cops that will never appear again. This will be useful later on to reduce the final vertices. For each vertex (C, H, r) with $H \neq \emptyset$, we add a *robber edge* to each vertex (C, H, r') , where r' is reachable from r in the graph $G \setminus C$. We also add a robber edge from each start vertex (C, \emptyset, \perp) to each vertex (C, \emptyset, r) . Finally, if there is a robber edge between (C, H, r) and (C, H, r') , we place a *placement edge* between (C, H, r') and $(C \cup H, \emptyset, r')$.

We can now determine whether there is a winning strategy by using the following *backward labeling*: First, mark all final vertices. Every vertex (C, H, r) with an outgoing placement edge to a marked vertex $(C \cup H, \emptyset, r)$ is also marked. If a vertex (C, \emptyset, r) has an outgoing cop edge to some marked vertex (C', H, r) , it is also marked. Finally, we mark a vertex (C, H, r) if *all* its children reachable by robber edges are marked. One can easily see that a start vertex (C, \emptyset, \perp) is marked by this process iff there is a winning strategy for $k+1$ cops that start at C . One can thus simply construct the arena graph \mathcal{G} in time $\mathcal{O}(n^{k+2})$ and perform this backward labeling.

1.3 Tamaki's Algorithm

The following description of Tamaki's algorithm is due to Bannach and Berndt [1], who generalized Tamaki's algorithm to a wide range of different graph parameters including *path width*, *tree depth*, and *D-width*. A closer look at the algorithm on the arena graph \mathcal{G} in the previous section reveals some places that might be optimized. First of all, there are n^{k+1} start vertices, but most of these start positions will be clearly unsuited for a winning strategy. But listing all of these start vertices takes nearly as much time as the complete algorithm. Furthermore, there might be a lot of *dead ends* in the graph, i. e. non-final vertices that can not lead to final vertices. Finally, we actually do not need every final vertex (C, \emptyset, r) , but only those of the form $(N[r], \emptyset, r)$, where $N[r]$ denotes the neighbourhood of r in G (including r itself). Note that we modeled our arena graph in such a way that we might forget some cops along the way in order to make this work. The naive approach described in the previous section lists all of these superfluous vertices.

We still want to perform the backward labeling, but we do not want to build the complete arena graph in order to do so. We will rather build the *active* vertices of the arena graph iteratively by using a queue-like structure. First, we add all n final vertices $(N[r], \emptyset, r)$ to the queue. While this queue is non-empty, we dequeue one vertex (C, H, r) and add all vertices with outgoing edges to (C, H, r) to the queue. While doing this, we also maintain the same backward labeling as before. If we ever mark a start vertex, we know that a winning strategy with

$k+1$ cops exists and have thus proved that $\text{tw}(G) \leq k$. Intuitively, this approach allows us to only look at situations that really might occur within in the game and get rid of the superfluous ones. Furthermore, by ordering the vertices within the queue in such a way that vertices with short distance to the start vertices are preferred, we might also be able to leave non-necessary parts of the arena graph unexplored.

This approach of getting rid of superfluous situations was coined *positive instance driven* (PID) dynamic programming [17]. The algorithm described above was used to win the PACE challenge in 2016 by Tamaki [16] on the normal arena graph. An alternative implementation of Tamaki’s algorithm due to Larisch and Salfelder [13] was used to win the PACE challenge in 2017. Tamaki’s submission to the PACE challenge in 2017 [18] (where he took the second place) made use of the fact that one can reduce the vertices in the arena graph to those corresponding to *maximal potential cliques*, as described by Bouchitté and Todinca in [5]. This improved algorithm is described in [17], but is not described in terms of the arena graph. To the best knowledge of the author, there is no formal description of Tamaki’s original algorithm used in the PACE challenge 2016 before the work of Bannach and Berndt [1]. Bannach and Berndt [1] used their description in terms of the arena graph to show that the running time of both variants of Tamaki’s algorithm [16, 18] is of the form $|\text{Active}(\mathcal{G})|^2 \cdot \text{poly}(n)$, where $\text{Active}(\mathcal{G})$ corresponds to the active vertices of \mathcal{G} described above.

Acknowledgments. The author likes to thank Max Bannach for many fruitful discussions around theoretical and practical approaches to tree width.

References

1. Bannach, M., Berndt, S.: Positive-Instance Driven Dynamic Programming for Graph Searching, unpublished
2. Bannach, M., Berndt, S., Ehlers, T.: Jdrasil: a modular library for computing tree decompositions. In: Proceedings of SEA. LIPIcs, vol. 75, pp. 28:1–28:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
3. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. In: Proceedings of STOC, pp. 226–234. ACM (1993)
4. Bodlaender, H.L., Koster, A.M.C.A.: Treewidth computations I. Upper bounds. *Inf. Comput.* **208**(3), 259–275 (2010)
5. Bouchitté, V., Todinca, I.: Listing all potential maximal cliques of a graph. *Theor. Comput. Sci.* **276**(1–2), 17–32 (2002)
6. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: *Parameterized Algorithms*. Springer, Heidelberg (2015). <https://doi.org/10.1007/978-3-319-21275-3>
7. Dell, H., Husfeldt, T., Jansen, B.M.P., Kaski, P., Komusiewicz, C., Rosamond, F.A.: The first parameterized algorithms and computational experiments challenge. In: Proceedings of IPEC. LIPIcs, vol. 63, pp. 30:1–30:9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)

8. Dell, H., Komusiewicz, C., Talmon, N., Weller, M.: The PACE 2017 parameterized algorithms and computational experiments challenge: the second iteration. In: Proceedings of IPEC. LIPIcs, pp. 30:1–30:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)
9. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer, London (2016). <https://doi.org/10.1007/978-1-4471-5559-1>
10. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, New York (2012). <https://doi.org/10.1007/978-1-4612-0515-9>
11. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006). <https://doi.org/10.1007/3-540-29953-X>
12. Karger, D.R., Srebro, N.: Learning markov networks: maximum bounded tree-width graphs. In: Proceedings of SODA, pp. 392–401. ACM/SIAM (2001)
13. Larisch, L., Salfelder, F.: p17 (2017). <https://github.com/freetdi/p17>. Accessed 1 Mar 2018
14. Seymour, P.D., Thomas, R.: Graph searching and a min-max theorem for tree-width. J. Comb. Theory Ser. B **58**, 22–33 (1993)
15. Song, Y., Liu, C., Malmberg, R.L., Pan, F., Cai, L.: Tree decomposition based fast search of RNA structures including pseudoknots in genomes. In: Proceedings of CSB, pp. 223–234. IEEE Computer Society (2005)
16. Tamaki, H.: Treewidth-exact (2016). <https://github.com/TCS-Meiji/treewidth-exact>. Accessed 1 Mar 2018
17. Tamaki, H.: Positive-instance driven dynamic programming for treewidth. In: Proceedings of ESA. LIPIcs, vol. 87, pp. 68:1–68:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
18. Tamaki, H., Ohtsuka, H.: tw-exact (2017). <https://github.com/TCS-Meiji/PACE2017-TrackA>. Accessed 1 Mar 2018



Using Structural Properties for Integer Programs

Sebastian Berndt¹(✉) and Kim-Manuel Klein²

¹ Department of Computer Science, Kiel University, Kiel, Germany
seb@informatik.uni-kiel.de

² École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
kim-manuel.klein@epfl.ch

Abstract. Integer programs (IPs) are one of the fundamental tools used to solve combinatorial problems in theory and practice. Understanding the structure of solutions of IPs is thus helpful to argue about the existence of solutions with a certain simple structure, leading to significant algorithmic improvements. Typical examples for such structural properties are solutions that use a specific type of variable very often or solutions that only contain few non-zero variables. The last decade has shown the usefulness of this method. In this paper we summarize recent progress for structural properties and their algorithmic implications in the area of approximation algorithms and fixed parameter tractability. Concretely, we show how these structural properties lead to optimal approximation algorithms for the classical MAKESPAN SCHEDULING scheduling problem and to exact polynomial-time algorithm for the BIN PACKING problem with a constant number of different item sizes.

1 Introduction

Integer programming is one of the fundamental tools used to solve scheduling problems in practice. Understanding the structure of optimal solutions of integer programs is helpful in many ways: First, we might want to find optimal solutions that have certain characteristics like having a small number of non-zero variables. Using structural theorems, we can show the (non)-existence of solutions with such characteristics. Second, understanding the structure of the integer program might also lead to algorithmic improvements: If one knows that some optimal solutions belong to a certain set S , we only need to search through S instead of the whole solution space.

This work was partially supported by the Swiss National Science Foundation (SNSF) within the project *Convexity, geometry of numbers, and the complexity of integer programming* (Nr. 163071) and DFG Project “Entwicklung und Analyse von effizienten polynomiellen Approximationsschemata für Scheduling- und verwandte Optimierungsprobleme”, Ja 612/14-2.

Our main focus is on IPs of the following form. Given a polytope $\mathcal{P} = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ for some matrix $A \in \mathbb{Z}^{d \times m}$ and vector $b \in \mathbb{Z}^d$. Then the IP is defined by

$$\begin{aligned} \min \quad & \sum x_p \\ \sum_{p \in \mathcal{P} \cap \mathbb{Z}^d} \quad & x_p p = a \\ & x \in \mathbb{Z}_{\geq 0}^d, \end{aligned} \tag{1}$$

with variables $x = (x_p)_{p \in \mathcal{P} \cap \mathbb{Z}^d}$. Throughout this work, let α be the largest entry in $\mathcal{P} \cap \mathbb{Z}^d$, i. e. $\alpha = \max_{p \in \mathcal{P} \cap \mathbb{Z}^d} \{\|p\|_\infty\}$. IPs of this specific shape are especially interesting when it comes to packing and scheduling problems, as the well-known *configuration IPs* are of this form. Intuitively, a configuration corresponds to a certain schedule of a single machine or a valid packing of a single object such as a bin. The configuration IP then chooses suitable configurations depending on the concrete programs. Here, we mainly consider the classical BIN PACKING problem and the classical MAKESPAN SCHEDULING problem on identical machines. In this paper, we summarize and explain the structural properties needed to obtain two important algorithmic advances in combinatorial optimization of the recent years.

BIN PACKING: In the BIN PACKING problem, item sizes $s_1, \dots, s_d \in (0, 1]$ are given. Every item size s_i occurs with a certain multiplicity a_i and the objective is to pack the given item into as few unit sized bins as possible. This problem can be written in the form of IP (1) by setting $\mathcal{P} = \{x \in \mathbb{R}_{\geq 0}^d \mid s_1 x_1 + \dots + s_d x_d \leq 1\}$. This polytope \mathcal{P} is called the *knapsack polytope* and an element $p \in \mathcal{P}$ (called a *pattern*) corresponds to a valid packing of a single bin. Hence, the optimal solution of IP (1) corresponds to a packing with a minimal number of bins.

For BIN PACKING we give an overview of the structural properties needed to obtain the following theorem, which solved a long standing open problem (see e. g. [3, 4, 10]).

Theorem 1 (Goemans and Rothvoß [5]). *Assuming the number of different item sizes d is constant, then there exists a polynomial time algorithm for BIN PACKING.*

MAKESPAN SCHEDULING: In the MAKESPAN SCHEDULING problem on identical machines, a set of n jobs with processing times p_1, \dots, p_d are given. The goal is to assign all jobs to m machines with identical speeds such that the maximum load over all machines (the so called *makespan*) is minimized. If T is a guess for the optimal makespan, we can determine whether a schedule with makespan at most T exists by using IP (1): We set $\mathcal{P}_T = \{x \in \mathbb{R}_{\geq 0}^d \mid p_1 x_1 + \dots + p_d x_d \leq T\}$ as the corresponding knapsack polytope. An element $p \in \mathcal{P}_T$ (called a *configuration*) corresponds to the packing of a single machine. Any solution of IP (1) with objective value at most m thus corresponds to a schedule with makespan at most T .

For MAKESPAN SCHEDULING, we summarize an algorithmic result in the area of approximation algorithms that settles the complexity of this problem (under a widely believed complexity assumption):

Theorem 2 (Jansen et al. [7]). *There is an approximation algorithm for classical MAKESPAN SCHEDULING on identical machines with a running time of*

$$2^{O(1/\epsilon \cdot \log^4(1/\epsilon))} + O(n \log n),$$

that produces an $(1 + \epsilon)$ -approximation.

The optimality of this algorithm is based on a theorem due to Chen et al. [2], which states that for all $\delta > 0$, the existence of an $(1 + \epsilon)$ -approximation algorithm for MAKESPAN SCHEDULING on identical machines running in time $2^{O((1/\epsilon)^{1-\delta})} \cdot \text{poly}(n)$ implies that the *Exponential Time Hypothesis (ETH)* is false. The ETH states that the satisfiability problem SAT can not be solved in truly subexponential time.

As we will see, both results are based on elementary structural results of the IP (1). However, the structural properties themselves differ in a very important point. In case of the scheduling problem, we do not require an optimal solution. This allows us to round processing times and ignore small jobs, as those small jobs can be placed greedily on an existing schedule without losing too much. Hence the entries of the MAKESPAN SCHEDULING IP turn out to be relatively small (smaller than $1/\epsilon$). Therefore, bounds in the structural properties may depend on the largest entry in the IP. In BIN PACKING however, where the goal is to get an optimal solution, small items can not be added greedily and therefore have to be placed optimally by the IP. This implies that the sizes of entries (which correspond to the inverse of the sizes of the items) of the IP can be very large. Therefore, structural properties that yield bounds independent of the size of the entries in IP (1) are needed.

2 Solutions of Bounded Support

One of the first structural results for IP (1) was to show that there always exists a solution (assuming the IP is feasible) with a bounded support. The support $\text{supp}(\lambda)$ of a solution $\lambda \in \mathbb{Z}^d$ is defined to be the set of non-zero components, i. e. $\text{supp}(\lambda) = \{i \mid \lambda_i > 0\}$ for $\lambda = (\lambda_1, \dots, \lambda_d)$.

Theorem 3 (Eisenbrand and Shmonin [3]). *Assuming that IP (1) is feasible, then there exists a feasible solution $\lambda \in \mathbb{Z}^d$ of (1) with*

$$|\text{supp}(\lambda)| \leq 2d \cdot \log(4d\alpha).$$

Actually this result does not only hold for IP (1) but for arbitrary integer programs with entries bounded by α . Very recently, this bound has been improved by Aliev et al. [1] to $|\text{supp}(\lambda)| \leq 2d \cdot \log(2\sqrt{d}\alpha)$. Furthermore, they

showed that this bound also holds for *optimal solutions* and arbitrary objective functions $\min c^T x$.

Similar to the previous theorem, Eisenbrand and Shmonin [3] proved a bound for the support of an integral solution of IP (1) that does not depend on the largest entry α in $\mathcal{P} \cap \mathbb{Z}^d$. We also give a sketch of the proof due to its very elementary nature.

Theorem 4 (Eisenbrand and Shmonin [3]). *Assuming IP (1) is feasible, then there exists a feasible solution λ of (1) such that $|\text{supp}(\lambda)| \leq 2^d$.*

Proof. We assign a *potential* $\varphi(\lambda)$ to each feasible solution λ of IP (1), with

$$\varphi(\lambda) = \sum_{p \in \mathcal{P} \cap \mathbb{Z}^d} \lambda_p \| (1, p)^\top \|^2,$$

where $\| \cdot \|$ is the Euclidean norm in \mathbb{R}^{d+1} . Informally, this potential is larger for solutions that are not in the center of the polytope. Let λ^* be the solution of IP (1) with the smallest potential. If $|\text{supp}(\lambda^*)| > 2^d$, the pigeonhole principle implies that there are $p_1, p_2 \in \mathcal{P} \cap \mathbb{Z}^d$ with the same parity, i. e. $p_1 \equiv p_2 \pmod 2$, as there are at most 2^d different parities. Hence, $p' = (p_1 + p_2)/2$ is also an element of $\mathcal{P} \cap \mathbb{Z}^d$. If $\lambda_{p_1}^* \geq \lambda_{p_2}^*$, we construct a new solution λ' that reassigns the weight from p_1 and p_2 to p' . Informally, we will shift weight towards the center of the polytope. Formally, we have $\lambda'_p = \lambda_p^*$ for $p \notin \{p_1, p_2, p'\}$. Furthermore, we set $\lambda'_{p_1} = \lambda_{p_1}^* - \lambda_{p_2}^*$, $\lambda'_{p_2} = 0$, and $\lambda'_{p'} = \lambda_{p'}^* + 2\lambda_{p_2}^*$. Clearly, λ' is a feasible solution of IP (1). As $(1, p_1)^\top$ and $(1, p_2)^\top$ are not co-linear, we have $\varphi(\lambda') < \varphi(\lambda^*)$. This is a contradiction to the minimality of $\varphi(\lambda^*)$. Hence, the solution with minimal potential has support of size at most 2^d . \square

3 Structural Results for IPs with Large Entries

By the previous section, we got an understanding of the support of solutions of IP (1). However, we would actually like to know more about the shape of these solutions. For example, we would like to understand what kind of configurations are actually used and what their respective multiplicities are. In this spirit, Goemans and Rothvoß showed the following theorem in order to prove Theorem 1.

Theorem 5 (Goemans and Rothvoß [5]). *Let \mathcal{P} be a polytope as in IP (1). There exists a distinguished set of configurations $X \subseteq \mathcal{P} \cap \mathbb{Z}^d$ with $|X| \leq m^d d^{O(d)} (\log \Delta)^d$ such that for every right hand side a of IP (1), there exists a vector $\lambda \in \mathbb{Z}_{\geq 0}^{\mathcal{P} \cap \mathbb{Z}^d}$ with $\sum_{p \in \mathcal{P} \cap \mathbb{Z}^d} \lambda_p p = a$ and*

1. $\lambda_p \leq 1 \quad \forall p \in (\mathcal{P} \cap \mathbb{Z}^d) \setminus X$
2. $|\text{supp}(\lambda) \cap X| \leq 2^{2d}$
3. $|\text{supp}(\lambda) \setminus X| \leq 2^{2d}$

Furthermore, X can be computed in time $\text{poly}(|X|)$.

The set $X \subseteq \mathcal{P} \cap \mathbb{Z}^d$ of distinguished configurations actually comes from a rather technical covering of the polytope \mathcal{P} into parallelepipeds. The set X is then the set of vertices of the parallelepipeds. So one might wonder if there exists a structure theorem with a rather natural set of distinguished point. Therefore, we consider the integer hull \mathcal{P}_I of the polytope \mathcal{P} which is defined by the convex hull of all integral points inside \mathcal{P} , i. e. $\mathcal{P}_I = \text{Conv}(\mathcal{P} \cap \mathbb{Z}^d)$. The following structure theorem uses the set of vertices V_I of \mathcal{P}_I , which gives a more natural alternative to the set X used in Theorem 5.

Theorem 6 (Jansen and Klein[6]). *Let \mathcal{P} be a polytope as in IP (1). For every right hand side a of IP (1), there exists a vector $\lambda \in \mathbb{Z}_{\geq 0}^{\mathcal{P} \cap \mathbb{Z}^d}$ with $\sum_{p \in \mathcal{P} \cap \mathbb{Z}^d} \lambda_p p = a$ and*

1. $\lambda_p \leq 2^{2^{O(d)}} \quad \forall p \in (\mathcal{P} \cap \mathbb{Z}^d) \setminus V_I$
2. $|\text{supp}(\lambda) \cap V_I| \leq d \cdot 2^d$
3. $|\text{supp}(\lambda) \setminus V_I| \leq 2^{2^d}$

Theorem 6 can now be used to prove Theorem 1. Note that the cardinality of the set of vertices V_I of \mathcal{P}_I is bounded by $d^d \cdot O(\log \alpha)^d$ and can be computed in time $|V_I| \cdot d^{O(d)}$. The right hand side a is given by the BIN PACKING instance. In order to determine the solution λ implied by Theorem 6, we first find the set $V_\lambda = \text{supp}(\lambda) \cap V_I$. Due to the second property of λ , there are at most $\binom{|V_I|}{d \cdot 2^d} \leq |V_I|^{O(2^d)}$ such sets. We also guess the cardinality of the set $\bar{V}_\lambda = \text{supp}(\lambda) \setminus V_I$. Due to the third property of λ , we have $|\bar{V}_\lambda| \leq 2^{2^d}$ and there are thus at most 2^{2^d} choices for k .

For guess V_λ and guess $|\bar{V}_\lambda|$, we need to (i) find the multiplicities λ_p for the pattern $p \in V_\lambda$ and we need to (ii) find the elements of \bar{V}_λ and their multiplicities. For (i), let $\hat{\lambda}_p$ be a variable describing the multiplicity of $p \in V_\lambda$. In order to solve (ii), the i -th element of \bar{V}_λ will be described by d variables $x_{i,1}, x_{i,2}, \dots, x_{i,d}$. As we also need to determine the multiplicity of the i -th element of \bar{V}_λ , we actually use variables $x_{i,j}^{(\ell)}$ indicating the j -th coordinate of the i -th element with multiplicity 2^ℓ . Note that there are at most 2^d variables $\hat{\lambda}_p$ and at most $d \cdot 2^{O(d)}$ variables $x_{i,j}^{(\ell)}$. One can thus formulate this task as an integer program with at most $2^{O(d)}$ variables which can be solved in time $(2^{O(d)})^{O(d)} \cdot \text{poly}(n)$ by the algorithms of Lenstra and Kannan [8,9].

By using a binary search on the number of bins, we can thus solve BIN PACKING in time $\text{poly}(n)^{f(d)}$ for some function f , which proves Theorem 1. A closer inspection shows that the running time is actually of the form $f(|V_I|) \cdot \text{poly}(n)$. The BIN PACKING problem is thus *fixed parameter tractable* for parameter $|V_I|$. See [6] for a more detailed discussion on this.

4 Structural Results for IPs with Small Entries

In contrast to the previous section, we are now interested in IPs that only contain small numbers. In order to determine the optimal makespan, we perform a binary

search. Let T be the current guess and $\mathcal{P}_T = \{x \in \mathbb{R}_{\geq 0}^d \mid p_1x_1 + \dots + p_dx_d \leq T\}$ be the polytope of valid configurations. To simplify notation, let π be the vector of the processing times. A configuration $p \in \mathcal{P}_T$ is called *simple* if $|\text{supp}(p)| \leq \log(T + 1)$. Otherwise, it is called *complex*. Let P_C be the set of complex configurations and P_S be the set of simple configurations. In the BIN PACKING setting, we saw that there is always an optimal solution that is mostly built on elements of V_I . Similarly, we will show that the MAKESPAN SCHEDULING problem always has an optimal solution that is mostly built on the simple configurations P_S .

Theorem 7 (Jansen et al. [7]). *Let α be the largest entry of any vector in $\mathcal{P}_T \cap \mathbb{Z}^d$. Assume that IP (1) is feasible. Then there exists a feasible solution λ to (1) such that:*

1. if $\lambda_p > 1$ then the configuration p is simple,
2. the support of λ satisfies $|\text{supp}(\lambda)| \leq 4d \log(4d\alpha)$, and
3. $\sum_{p \in P_C} \lambda_p \leq 2d \log(4d\lambda)$, where P_C denotes the set of complex configurations.

Proof. The main idea in the proof is that for any complex configuration p , we can write $2p$ as $2p = p_1 + p_2$ for two configurations p_1, p_2 with $\pi \cdot p_1 = \pi \cdot p_2$ and $\text{supp}(p_i) \subsetneq \text{supp}(p)$. For $S \subseteq \text{supp}(p)$, let $p[S]$ be the configuration with

$$p[S]_i = \begin{cases} 1 & i \in S \\ 0 & \text{else.} \end{cases}$$

Note that $|\{p[S] \mid S \subseteq \text{supp}(p)\}| = 2^{|\text{supp}(p)|} > 2^{\log(T+1)} = T + 1$. As $\pi \cdot p[S] \leq \pi \cdot p \leq T$, the pidgeonhole principle implies that there are different sets $S_1, S_2 \subseteq \text{supp}(p)$ with $\pi \cdot p[S_1] = \pi \cdot p[S_2]$. Clearly, $\pi \cdot p[S'_1] = \pi \cdot p[S'_2]$ also holds for $S'_1 = S_1 \setminus S_2$ and $S'_2 = S_2 \setminus S_1$. Then $p_1 = p + p[S'_1] - p[S'_2]$ and $p_2 = p + p[S'_2] - p[S'_1]$ are the desired configurations.

If $\lambda^{(0)}$ is a feasible solution, we can now iteratively reduce complex configurations $p \in P_C$ with $\lambda_p^{(0)} \geq 2$ into smaller configurations p_1 and p_2 and achieve a solution $\lambda^{(1)}$. We repeat this, until our final solution $\lambda^{(k)}$ has the property that $\lambda_p^{(k)} \leq 1$ for all $p \in P_C$. Finally, we can use Theorem 3 to bound the cardinality of the support. One needs to be careful here, as the solution implied by Theorem 3 might contain a complex configuration with higher multiplicity. This problem can be avoided by making use of a potential function that guarantees that complex configurations do not reappear in this step. \square

If one wants to compute a $(1 + \epsilon)$ -approximation, we can discard all processing times smaller than $\epsilon \cdot \text{LB} = \epsilon \cdot \max\{p_{\max}, \sum_j p_j/m\}$, where p_{\max} is the maximal processing time of a single job. The discarded jobs can be added greedily later on. This process is the reason why the corresponding IP has only small entries. The remaining n' large jobs with processing time p such that $\epsilon \text{LB} \leq p \leq \text{LB}$ can be rounded geometrically to the form $\epsilon \cdot \text{LB} \cdot (1 + \epsilon)^i$ without losing too much precision. After this preprocessing, the number of different processing times is thus $d' = O(1/\epsilon \log(1/\epsilon))$. By normalizing our instance, we can assume that

$T \leq 1/\epsilon^2$ and that all processing times are integers between $1/\epsilon$ and $1/\epsilon^2$. Let $\pi' = (\pi'_1, \dots, \pi'_{d'})$ be the vector corresponding to these processing times and n'_j be the number of jobs with processing times π'_j .

By formulating IP (1) for this reduced problem, we can use the following approach:

1. For each $j = 1, \dots, d'$, guess the number $n_j^C \leq n'_j$ of jobs covered by complex configurations.
2. Find the minimum number of machines m^C to schedule the $n^C = \sum_j n_j^C$ jobs with makespan T .
3. To schedule the remaining jobs with simple configurations, guess the simple configurations $S \subseteq P_S$.
4. Solve the IP (1) restricted to the $n^S = n' - n^C$ jobs that will be scheduled by simple configurations in S .

For step 1, note that Theorem 7 guarantees that m^C – the number of machines scheduled according to complex configurations – is at most $2d' \log(4d'\alpha) \in O(1/\epsilon \log(1/\epsilon))$. As only $1/\epsilon$ jobs belong to a configuration (they are of size at least $1/\epsilon$), there are at most $O(1/\epsilon^2 \log(1/\epsilon))$ many jobs covered by complex configurations. There are thus at most $\sum_{k=0}^{O(1/\epsilon^2 \log(1/\epsilon))} \binom{k+d'-1}{d'-1} \leq 2^{O(1/\epsilon \log^2(1/\epsilon))}$ many such choices. Step 2 can thus be solved by a dynamic program in the same running time of $2^{O(1/\epsilon \log^2(1/\epsilon))}$. For step 3, note that $|P_S| \leq 2^{O(\log^2(1/\epsilon))}$, as $T \leq 1/\epsilon^2$ and each job has size at least $1/\epsilon$. As our desired solution uses at most $4d' \log(4d'\alpha) \in O(1/\epsilon \log^2(1/\epsilon))$ different configurations in total, we can try out all $\sum_{k=0}^{O(1/\epsilon \log^2(1/\epsilon))} \binom{|P_S|}{k} \leq 2^{O(1/\epsilon \log^4(1/\epsilon))}$ choices for the support S of the simple configurations. Finally, the IP in step 4 has at most $O(1/\epsilon \log^2(1/\epsilon))$ many variables and can thus be solved by the algorithms of Lenstra and Kannan [8, 9] in time $2^{O(1/\epsilon \log^3(1/\epsilon))} \cdot \log(n)$. Due to the preprocessing and the binary search, the complete algorithm thus runs in time $2^{O(1/\epsilon \log^4(1/\epsilon))} + O(n \log n)$. This proves Theorem 2.

5 Conclusions

Understanding the structure of solutions of combinatorial problems has been one of the most successful strategies to design efficient algorithms for a long time. Recent advances in the theory of integer programming allow to take a closer look at the structure of solution space of IP formulations of these combinatorial problems. This led to a series of exciting results for longstanding open problems. We have seen that BIN PACKING can be solved in time $f(|V_I|) \cdot \text{poly}(n)$, which can be bounded by $\text{poly}(n)^{g(d)}$ for some function g . Hence, for constant d , this running time is polynomial. A major open question is whether one can obtain a running time of $f(d) \cdot \text{poly}(n)$, i. e. whether BIN PACKING is fixed parameter tractable with parameter d .

References

1. Aliev, I., De Loera, J., Eisenbrand, F., Oertel, T., Weismantel, R.: The support of integer optimal solutions. [arxiv:1712.08923](https://arxiv.org/abs/1712.08923) (2017)
2. Chen, L., Jansen, K., Zhang, G.: On the optimality of approximation schemes for the classical scheduling problem. In: Proceedings of SODA. SIAM, pp. 657–668 (2014)
3. Eisenbrand, F., Shmonin, G.: Carathodory bounds for integer cones. *Oper. Res. Lett.* **34**(5), 564–568 (2006)
4. Filippi, C.: On the bin packing problem with a fixed number of object weights. *Eur. J. Oper. Res. (EJOR)* **181**(1), 117–126 (2007)
5. Goemans, M.X., Rothvoß, T.: Polynomiality for bin packing with a constant number of item types. In: Proceedings of SODA, pp. 830–839 (2014)
6. Jansen, K., Klein, K.: About the structure of the integer cone and its application to bin packing. In: Proceedings of SODA. SIAM, pp. 1571–1581 (2017)
7. Jansen, K., Klein, K., Verschae, J.: Closing the gap for makespan scheduling via sparsification techniques. In: Proceedings of ICALP, LIPIcs, vol. 55, pp. 72:1–72:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
8. Kannan, R.: Minkowski's convex body theorem and integer programming. *Math. Oper. Res.* **12**(3), 415–440 (1987)
9. Lenstra, H.W.: Integer programming with a fixed number of variables. *Math. Oper. Res.* **8**, 538–548 (1983)
10. McCormick, S.T., Smallwood, S.R., Spieksma, F.C.R.: Polynomial algorithms for multiprocessor scheduling with a small number of job lengths. In: Proceedings of SODA. SIAM, pp. 509–517 (1997)



From Eventually Different Functions to Pandemic Numberings

Achilles A. Beros¹, Mushfeq Khan¹, Bjørn Kjos-Hanssen¹ ^(✉),
and André Nies²

¹ University of Hawai'i at Mānoa, Honolulu, HI 96822, USA
{beros,mushfeq,bjoernkh}@hawaii.edu

² University of Auckland, Auckland, New Zealand
nies@cs.auckland.ac.nz

Abstract. A function is strongly non-recursive (SNR) if it is eventually different from each recursive function. We obtain hierarchy results for the mass problems associated with computing such functions with varying growth bounds. In particular, there is no least and no greatest Muchnik degree among those of the form SNR_f consisting of SNR functions bounded by varying recursive bounds f .

We show that the connection between SNR functions and canonically immune sets is, in a sense, as strong as that between DNR (diagonally non-recursive) functions and effectively immune sets. Finally, we introduce pandemic numberings, a set-theoretic dual to immunity.

1 Introduction

It has been known for over a decade that bounding diagonally non-recursive functions by various computable functions leads to a hierarchy of computational strength [1, 16] and this hierarchy interacts with Martin-Löf random reals and completions of Peano Arithmetic [10, 11]. The strongly non-recursive functions form an arguably at least as natural class, and here we start developing analogous hierarchy results for it.

Definition 1. A function $f : \omega \rightarrow \omega$ is strongly nonrecursive (or SNR) if for every recursive function g , for all but finitely many $n \in \omega$, $f(n) \neq g(n)$. It is strongly non-partial-recursive (or SNPR) if for every partial recursive function g , for all but finitely many n , if $g(n)$ is defined, $f(n) \neq g(n)$.

Note that every SNPR function f is SNR, as well as *almost* DNR: for all but finitely many n , if $\varphi_n(n)$ is defined, then $f(n) \neq \varphi_n(n)$. Also, a function is strongly nonrecursive iff it is *eventually different* from each recursive function. Thus it is eventually different in the sense of set theory with the recursive sets as ground model [2].

This work was partially supported by a grant from the Simons Foundation (#315188 to Bjørn Kjos-Hanssen).

Definition 2. An order function is a recursive, nondecreasing, and unbounded function $h : \omega \rightarrow \omega$ such that $h(0) \geq 2$. For a class C of functions from ω to ω , let C_h denote the subclass consisting of those members of C that are bounded by h .

Theorem 1. For each order function h , there exists an order function g such that every DNR_g function computes an SNPR_h function.

In order to prove this theorem, we will need a result due to Cenzer and Hinman [8], in a form presented in Greenberg and Miller [10].

Definition 3 (Greenberg and Miller[10]). Let $a \geq 2$ and let $c > 0$. Let \mathcal{P}_a^c denote the class of functions f bounded by a such that for all e and for all $x < c$, if $\varphi_e(x) \downarrow$, then $f(e) \neq \varphi_e(x)$.

Theorem 2 (Cenzer and Hinman [8]). Let $a \geq 2$ and $c > 0$. Then any DNR_a function computes a function in \mathcal{P}_{ca}^c . Moreover, the reduction is uniform in a and c .

Proof (Proof of Theorem 1). Let r be the recursive function such that $\varphi_{r(x)}(e) = \varphi_e(x)$. For each $n \geq 2$, let $x_n \in \omega$ be the least such that $h(x_n) \geq n^2$.

We construct g to ensure that any DNR_g function computes a function f that is bounded by h and such that for all $x > x_n$ and for all $e < n$, if $\varphi_e(x) \downarrow$, then $f(x) \neq \varphi_e(x)$.

In order to compute f on the interval $[x_n, x_{n+1})$, a function in $\mathcal{P}_{n^2}^n$ suffices and such a function can be uniformly obtained from a DNR_n function, by Theorem 2. However, we only need a finite part of this function, and we can recursively determine how much. Note that the reductions in Theorem 2 can be assumed to be total. For $n \geq 2$, let γ_n denote the use of reduction that, given a DNR_n function, computes a $\mathcal{P}_{n^2}^n$ function.

Then, letting

$$m_n = \max(\{\gamma_n(r(x)) : x \in [x_n, x_{n+1})\}),$$

it suffices for g to be any recursive function such that for all $x \leq m_n$, $g(x) \leq n$.

We also have the following counterpart to Theorem 1:

Theorem 3. For each order function g , there exists an order function h such that every DNR_g function computes an SNPR_h function.

Proof. For $n \geq 1$, let $\tau_n : \omega \rightarrow \omega^n$ be a uniformly recursive sequence of bijections, and for $i < n$, $\pi_i^n : \omega^n \rightarrow \omega$ denote the projection function onto the i -th coordinate.

Let r be the recursive function such that for $n \geq 1$ and $e < n$, if $\varphi_e(n)$ converges, then $\varphi_{r(e,n)}$, on any input, outputs $\pi_e^n(\tau_n(\varphi_e(n)))$.

Now, given a DNR_g function f , let $j(0) = 0$ and for $n \geq 1$, let

$$j(n) = \tau_n^{-1}(\langle f(r(0, n)), \dots, f(r(n-1, n)) \rangle).$$

Then $j(n) \neq \varphi_e(n)$ for any $e < n$: If it were, then we would have

$$f(r(e, n)) = \pi_e^n(\tau_n(j(n))) = \pi_e^n(\tau_n(\varphi_e(n))) = \varphi_{r(e, n)}(r(e, n)),$$

which contradicts the fact that f is DNR. Thus, j is SNPR_h where for $n \geq 1$,

$$h(n) = \max(\{\tau_n^{-1}(\langle i_1, \dots, i_{n-1} \rangle) : i_k < g(r(k, n)) \text{ for all } k < n\}).$$

Theorem 4 (Kjos-Hansen et al. [13]). *Every non-high SNR is SNPR .*

Proof. Suppose that $f : \omega \rightarrow \omega$ is not high, and that ψ is a partial recursive function that is infinitely often equal to it. For each $n \in \omega$, let $g(n)$ be the least stage such that $|\{x \in \omega : \psi(x)[g(n)] \downarrow = f(x)\}| \geq n + 1$. Then g is recursive in f .

Since f is not high, there is a recursive function h that escapes g infinitely often. We define a recursive function j that is infinitely often equal to f . Let $j_0 = \emptyset$. Given j_n , let

$$A = \{\langle x, \psi(x) \rangle : x \notin \text{dom}(j_n), \psi(x)[h(n)] \downarrow\}.$$

Let y be the least such that it is not in the domain of $j_n \cup A$. Finally, let $j_{n+1} = j_n \cup A \cup \langle y, 0 \rangle$.

It is easily seen that $j = \bigcup_n j_n$ is recursive and infinitely often equal to f .

Corollary 1. *Given any order function h , every non-high SNR_h function computes a DNR_h function.*

2 The SNR Hierarchy

2.1 Definitions and Combinatorial Lemmas

The following definitions can also be found in [10] and [12].

Definition 4. *Given $\sigma \in \omega^{<\omega}$, we say that a tree $T \subseteq \omega^{<\omega}$ is n -bushy above σ if every element of T is comparable with σ , and for every $\tau \in T$ that extends σ and is not a leaf of T , τ has at least n immediate extensions in T . We refer to σ as the stem of T .*

Definition 5. *Given $\sigma \in \omega^{<\omega}$, we say that a set $B \subseteq \omega^{<\omega}$ is n -big above σ if there is a finite n -bushy tree T above σ such that all its leaves are in B . If B is not n -big above σ then we say that B is n -small above σ .*

Proofs of the following lemmas can be found in [10] and [12].

Lemma 1 (Smallestness preservation property). *Suppose that B and C are subsets of $\omega^{<\omega}$ and that $\sigma \in \omega^{<\omega}$. If B and C are respectively n - and m -small above σ , then $B \cup C$ is $(n + m - 1)$ -small above σ .*

Lemma 2 (Small set closure property). *Suppose that $B \subseteq \omega^{<\omega}$ is n -small above σ . Let $C = \{\tau \in \omega^{<\omega} : B \text{ is } n\text{-big above } \tau\}$. Then C is n -small above σ . Moreover C is n -closed, meaning that if C is n -big above a string ρ , then $\rho \in C$.*

Definition 6. Given an order function h , let $h^{<\omega}$ denote the set of finite strings in $\omega^{<\omega}$ whose entries are bounded by h , and let h^n denote the set of such strings of length n .

Theorem 5. Let h be any order function. Then, uniformly in h , we can find a recursive function π such that if g is any order function such that $h(n)/g(\pi(n))$ is unbounded, then there is a low $f \in \text{DNR}_h$ that computes no DNR_g function.

Proof. Given $\sigma \in h^{<\omega}$, let $q(\sigma, e, k)$ be an index for the partial recursive function that searches for a k -big set $A \subset h^{<\omega}$ above σ such that $\Phi_e^\tau(q(\sigma, e, k))$ converges and is constant as τ ranges over A , and which then outputs this constant value. Let

$$\pi(n) = \max\{q(\sigma, e, k) : \sigma \in h^n, e, k \leq n\}.$$

Next, we describe a $0'$ -recursive construction of f . We define a sequence

$$f_0 \preceq f_1 \preceq f_2 \preceq \dots$$

of finite strings in $h^{<\omega}$, and

$$B_0 \subseteq B_1 \subseteq B_2 \subseteq \dots$$

of r.e. subsets of $h^{<\omega}$ such that for each $s \in \omega$, B_s is $h(|f_s|)$ -small and $h(|f_s|)$ -closed above f_s .

Let $f_0 = \langle \rangle$, and let B_0 be the set of non-DNR strings. Next, we describe how to construct f_{s+1} and B_{s+1} given f_s and B_s .

If $s = 2e$ is even: We ensure that Φ_e^f is not DNR_g . Let $k = h(|f_s|)$ and let $n \geq k, e$ be the least such that $h(n) \geq k(g(\pi(n)) + 1)$. We begin by extending f_s to a string $\sigma \notin B_s$ of length n . Note that B_s is k -small and k -closed above σ . Let $x = q(\sigma, e, k)$, and note that $x \leq \pi(n)$.

Now, if $\varphi_x(x) \downarrow$ to some value i less than $g(x)$, then the set

$$A_i = \{\tau \succeq \sigma : \Phi_e^\tau(x) \downarrow = i\}$$

is k -big above σ , so there is an extension τ of σ such that $\tau \in A_i \setminus B_s$. Let $f_{s+1} = \tau$ and $B_{s+1} = B_s$. This forces Φ_e^f to not be DNR.

Otherwise, for each $i < g(x)$, A_i is k -small above σ , and so

$$C = \bigcup_{i < g(x)} A_i$$

is $kg(x)$ -small above σ , and $C \cup B_s$ is $k(g(x) + 1)$ -small above σ . Since $h(n) \geq k(g(\pi(n)) + 1) \geq k(g(x) + 1)$, we can let $B_{s+1} = B_s \cup C$ and $f_{s+1} = \sigma$. This forces $\Phi_e^f(x)$ to either diverge, or to converge to value greater than or equal to $g(x)$.

If $s = 2e + 1$ is odd: We ensure that f is low. We begin by extending f_s to a string $\sigma \notin B_s$ such that $h(|\sigma|) \geq 2h(|f_s|)$. If the set

$$F_e = \{\tau \succeq \sigma : \varphi_e^\tau(e) \downarrow\}$$

is $h(|f_s|)$ -big above σ , then there is a $\tau \in F_e \setminus B_s$. Let $f_{s+1} = \tau$ and $B_{s+1} = B_s$. This forces e into the jump of f . Otherwise, let $f_{s+1} = \sigma$ and let $B_{s+1} = B_s \cup F_e$, which is $h(|\sigma|)$ -small above σ . This forces e out of the jump of f .

By making g grow slowly enough, we get:

Corollary 2. *For every order function h there is an order function g such that there is a low DNR_h that computes no DNR_g .*

Additionally, we have:

Corollary 3. *For every order function g there is an order function h such that there is a low DNR_h that computes no DNR_g .*

Proof. Using the uniformity in Theorem 5 along with the recursion theorem, we construct h knowing its index in advance, thereby obtaining π , and ensuring that $h(n)/g(\pi(n))$ is unbounded.

By combining the strategies for the two corollaries above, we get:

Corollary 4. *For every order function h there is an order function g such that there is a low $f_1 \in \text{DNR}_g$ that computes no DNR_h as well as a low $f_2 \in \text{DNR}_h$ that computes no DNR_g .*

Corollary 5. *Given any order function h there is an order function g such that there is an SNR_h that computes no SNR_g .*

Proof. By Theorem 1, there is an order function h' such that any $\text{DNR}_{h'}$ computes an SNR_h . By Corollary 2, there is an order function g such that there is a low $\text{DNR}_{h'}$ function f' that computes no DNR_g function. Then f' computes an SNR_h function f that computes no SNR_g function: if j is recursive in f and is an SNR_g function and since it is low, it is itself DNR_g by Corollary 1, a contradiction.

Corollary 6. *Given any order function g there is an order function h such that there is an SNR_h that computes no SNR_g .*

Proof. By Corollary 3, there is an h' and a low $\text{DNR}_{h'}$ function f' that computes no DNR_g function. By Theorem 3 there is an h such that f' computes an SNR_h function f . Then f cannot compute an SNR_g function since the latter would be DNR_g by Corollary 1.

Let \mathfrak{D} denote the set of all order functions. Recall the Muchnik and Medvedev reducibilities of mass problems:

Definition 7. *A mass problem \mathcal{A} is Muchnik reducible to a mass problem \mathcal{B} , written $\mathcal{A} \leq_w \mathcal{B}$ and sometimes read “weakly reducible”, if for each $B \in \mathcal{B}$, there is an $A \in \mathcal{A}$ such that $A \leq_T B$, where \leq_T is Turing reducibility. If there is a single Turing reduction Φ such that for all $B \in \mathcal{B}$, $\Phi^A \in \mathcal{A}$ then \mathcal{A} is Medvedev reducible to \mathcal{B} , written $\mathcal{A} \leq_S \mathcal{B}$ and sometimes read “strongly reducible”.*

We can phrase Corollaries 6 and 5 as follows:

$$\forall h \in \mathfrak{D} \quad \exists g \in \mathfrak{D} \quad \text{SNR}_g \not\leq_w \text{SNR}_h;$$

$$\forall g \in \mathfrak{D} \quad \exists h \in \mathfrak{D} \quad \text{SNR}_g \not\leq_w \text{SNR}_h.$$

Thus, the Muchnik degrees of various mass problems SNR_f have no least or greatest element.

3 Canonical Immunity

Canonical immunity was introduced by three of the present authors in [6] and shown there to be equivalent, as a mass problem, to SNR, and studied further in [5]. Here we give a new Theorem 10 below, analogous to the case of DNR, that was not obtained in [6].

Considering lowness notions associated with Schnorr randomness was what lead those authors to this new notion of immunity.

Definition 8. A canonical numbering of the finite sets is a surjective function $D : \omega \rightarrow \{A : A \subseteq \omega \text{ and } A \text{ is finite}\}$ such that $\{(e, x) : x \in D(e)\}$ is recursive and the cardinality function $e \mapsto |D(e)|$, or equivalently, $e \mapsto \max D(e)$, is also recursive.

We write $D_e = D(e)$.

Definition 9. R is canonically immune (CI) if R is infinite and there is a recursive function h such that for each canonical numbering of the finite sets D_e , $e \in \omega$, we have that for all but finitely many e , if $D_e \subseteq R$ then $|D_e| \leq h(e)$.

We include proofs of some of the results from [6].

Theorem 6 (Beros et al. [6]). Schnorr randoms are canonically immune.

Proof. Fix a canonical numbering of the finite sets, $\{D_e\}_{e \in \omega}$. Define $U_c = \{X : (\exists e > c)(|D_e| \geq 2e \wedge D_e \subset X)\}$. Since $e \mapsto |D_e|$ is recursive, $\mu(U_c)$ is recursive and bounded by 2^{-c} . Thus, the sequence $\{U_c\}_{c \in \omega}$ is a Schnorr test. If A is a Schnorr random, then $A \in U_c$ for only finitely many $c \in \omega$. We conclude that A is canonically immune.

Theorem 7 (Beros et al. [6]). Each canonically immune set is immune.

Proof. Suppose A has an infinite recursive subset R . Let h be any recursive function. Let R_n denote the set of the first n elements of R , and let $\{D_e : e \in \omega\}$ be a canonical numbering of the finite sets such that $D_{2n} = R_{h(2n)+1}$ for all $n \in \omega$. For all n , $D_{2n} \subseteq R \subseteq A$ and $|D_{2n}| = h(2n) + 1 > h(2n)$, and so h does not witness the canonical immunity of A .

We now show that canonically immune is the “correct” analogue of effectively immune. Let W_0, W_1, W_2, \dots be an effective enumeration of the recursively enumerable (or r.e.) sets of natural numbers. An infinite set A of natural numbers is said to be *immune* if it contains no infinite r.e. subset. It is said to be *effectively immune* when there is a recursive function f such that for all e , if W_e is a subset of A , then $|W_e| \leq f(e)$. The interest in sets whose immunity is effectively witnessed in this manner originally arose in the search for a solution to Post’s problem; for more on this the reader may see [6].

Theorem 8 (Beros et al. [6]). *Each canonically immune (CI) set computes a strongly nonrecursive function, i.e.,*

$$\text{SNR} \leq_w \text{CI}.$$

Incidentally, Beros and Beros [4] showed that the index set of Medvedev reductions from CI to SNR is Π_1^1 -complete.

Theorem 9 (Kjos-Hanssen [13]). *Each SNR function is either of high or DNR Turing degree.*

Corollary 7 (Beros et al. [6]). *The following are equivalent for an oracle A :*

1. *A computes a canonically immune set,*
2. *A computes an SNR function,*
3. *A computes an infinite subset of a Schnorr random.*

If a canonically immune set A is of non-high Turing degree then by Theorem 8, A computes an SNR function, which by Theorem 9 means that A computes a DNR function, hence A computes an effectively immune set. Our new result is to make this more direct: A is itself that effectively immune set.

Theorem 10. *If A is non-high and canonically immune, then A is effectively immune.*

Proof. Let us introduce the notation

$$W_{e,s}\#u$$

to mean the set of $k \in W_{e,s}$ such that when k enters $W_{e,s}$, at most u other numbers have already entered $W_{e,s}$. Roughly speaking, $D\#u$ consists of the first $u + 1$ elements of D .

Let A be non-high and not effectively immune. We need to show that A is not canonically immune.

Since A is not effectively immune, there are infinitely many e for which there is an s with $\langle e, s \rangle \in \mathcal{F}$ where

$$\mathcal{F} = \{ \langle e, s \rangle : |W_{e,s}| > h(e), \text{ and } W_{e,s}\#h(e) \subseteq A \}.$$

Here we assume h is nondecreasing. So

$$\forall d \ \exists s \ \exists e_d > d \ \langle e_d, s \rangle \in \mathcal{F}$$

Let $f(d) = s$. Then there is a recursive function $g(d)$ which is not dominated by f .

We may assume $s > e_d$ since increasing s will keep $\langle e, s \rangle \in \mathcal{F}$. Let

$$D_{2\langle e,d \rangle} = W_{e,g(d)} \# h(d)$$

Let $\tilde{h}(\langle e, d \rangle) = h(d)$ for $d \leq e \leq g(d)$. Using a recursive bijection between the domain of \tilde{h} ,

$$\{\langle e, d \rangle : d \leq e \leq g(d)\}$$

and ω , we may assume the domain of \tilde{h} is ω .

Then for infinitely many e (namely, there are infinitely many d with $f(d) \leq g(d)$, and for each such d there is an e with $d \leq e \leq g(d)$ that works) we have

1. $D_{2\langle e,d \rangle} = W_{e,g(d)} \# h(d) \subseteq W_{e,g(d)} \# h(e) \subseteq A$, and
2. $|D_{2\langle e,d \rangle}| > \tilde{h}(\langle e, d \rangle) = h(d)$.

We have $\lim_{n \rightarrow \infty} \tilde{h}(n) = \infty$ since for each d we only include e up to the e_d above.

Let the sets D_{2k+1} be a canonical list of all the finite sets, just in case we missed some of them using the sets D_{2k} .

4 Pandemic Numberings

Brendle et al. [7] explored an analogy between the theory of cardinal characteristics in set theory and highness properties in computability theory, following up on work of Rupprecht [15].

Their main results concerned a version of Cichon’s diagram [3] in computability theory. They expressed the cardinal characteristics in Cichon’s diagram as either

$$\mathfrak{d}(R) = \min\{|F| : F \subseteq Y \text{ and } \forall x \in X \exists y \in F (x R y)\},$$

$$\mathfrak{d}(R) = \min\{|F| : F \subseteq Y \text{ and } \forall x \in X \exists y \in F (x R y)\}$$

or

$$\mathfrak{b}(R) = \min\{|G| : G \subseteq X \text{ and } \forall y \in Y \exists x \in G \neg(x R y)\},$$

$$\mathfrak{b}(R) = \min\{|G| : G \subseteq X \text{ and } \forall y \in Y \exists x \in G \neg(x R y)\},$$

where X and Y are two spaces and R is a relation on $X \times Y$. As the spaces considered admit a notion of relative computability, it is natural to say that an element $x \in X$ is computable (in $A \subset \omega$).

They defined two computability-theoretic notions corresponding to $\mathfrak{d}(R)$ and $\mathfrak{b}(R)$ as follows:

$$\mathcal{B}(R) = \{A : \exists y \leq_T A \forall x (x \text{ is computable} \rightarrow x R y)\},$$

$$\mathcal{B}(R) = \{A : \exists y \leq_T A \forall x (x \text{ is computable} \rightarrow x R y)\},$$

and

$$\mathcal{D}(R) = \{A : \exists x \leq_T A \forall y (y \text{ is computable} \rightarrow \neg(x R y))\},$$

$$\mathfrak{D}(R) = \{A : \exists x \leq_T A \forall y (y \text{ is computable} \rightarrow \neg(x R y))\}.$$

They found that $\mathcal{B}(R)$ and $\mathcal{D}(R)$ tend to be highness properties in computability theory, often equivalent to well-known notions.

They finally mapped a cardinal characteristic $\mathfrak{d}(R)$ or $\mathfrak{b}(R)$ to $\mathcal{D}(R)$ or $\mathcal{B}(R)$ respectively, and showed that if $\mathfrak{b}(R) \leq \mathfrak{d}(S)$, then $\mathcal{B}(R) \subseteq \mathcal{D}(S)$ and so on.

The resulting analog of Cichon’s diagram is not isomorphic to the original diagram, in that some strict inequalities of cardinal characteristics are consistent with ZFC but the corresponding computability-theoretic notions coincide.

Using their point of view we find a new dual notion to canonical immunity: that of a numbering such that no recursive set is large with respect to it.

Definition 10. Let $D = (e \mapsto D_e)$ be a numbering of the finite subsets of ω and let $R \subseteq \omega$. We say that D is h -endemic to R if there are infinitely many e with $|D_e| \geq h(e)$ and $D_e \subseteq R$. D is a pandemic numbering if there is an order function h such that for all infinite recursive sets R , D is h -endemic to R .

Recall also that an escaping function is a function $f : \omega \rightarrow \omega$ such that f is not dominated by any recursive function.

Theorem 11. The Muchnik degrees of pandemic numberings and of hyperimmune sets coincide.

Proof. In one direction, if $e \mapsto \max D_e$ is recursively bounded then we show that D is not a pandemic numbering. Namely, we construct a recursive set R which waits for h to get large ($h(e) > d$ say) and only then lets its d th element r_d enter R , and lets r_d be large enough (larger than $\max D_k, k \leq e$) to prevent $|D_e| \geq h(e), D_e \subseteq R$.

In the other direction, given a hyperimmune set, we (as is well known) also have an escaping function f . At those inputs e where f is greater than a function associated with a potential infinite recursive set R_k (given as the graph of a partial recursive $\{0, 1\}$ -valued function), we define the next D_e so as to ensure $D_e \subseteq R$ and $|D_e| \geq h(e)$. Namely, the function to escape is the time $e \mapsto t_k(e) = t(\langle e, k \rangle)$ it takes for R_k to get $h(e)$ many elements. If R_k is really infinite then t_k is total and so $f(e) \geq t_k(e)$ for some (infinitely many) e . So we define $D_{\langle e, k \rangle}$ to consist of the first $h(\langle e, k \rangle)$ elements of R_k , if any, as found during a search time of $f(\langle e, k \rangle)$ or even just $f(e)$.

By Corollary 7 and Theorem 11, the dualism between immunity and pandemics is the same, Muchnik-degree-wise, as that between, in the notation of Brendle et al. [7],

- eventually different functions, $\mathfrak{b}(\neq^*)$, and
- functions that are infinitely often equal to each recursive function, $\mathfrak{d}(\neq^*)$.

Remark 1. As the recursive sets are closed under complement, a notion of bi-pandemic would be the same as pandemic. Thus, while we do not know whether canonical bi-immunity is Muchnik equivalent to canonical immunity, there is no corresponding open problem on the dual side.

References

1. Ambos-Spies, K., Kjos-Hanssen, B., Lempp, S., Slaman, T.A.: Comparing DNR and WWKL. *J. Symb. Log.* **69**(4), 1089–1104 (2004)
2. Bartoszyński, T., Judah, H.: *Set Theory: On the Structure of the Real Line*. A K Peters Ltd., Wellesley (1995)
3. Bartoszyński, T., Judah, H., Shelah, S.: The Cichoń diagram. *J. Symb. Log.* **58**(2), 401–423 (1993)
4. Beros, A.A., Beros, K.A.: Index sets of universal codes. arXiv e-prints, October 2016
5. Beros, A.A., Beros, K.A.: Canonical immunity and genericity. arXiv e-prints, July 2017
6. Beros, A.A., Khan, M., Kjos-Hanssen, B.: Effective Bi-immunity and randomness. In: Day, A., Fellows, M., Greenberg, N., Khossainov, B., Melnikov, A., Rosamond, F. (eds.) *Computability and Complexity*. LNCS, vol. 10010, pp. 633–643. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-50062-1_38
7. Brendle, J., Brooke-Taylor, A., Ng, K.M., Nies, A.: An analogy between cardinal characteristics and highness properties of oracles. In: *Proceedings of the 13th Asian Logic Conference*, pp. 1–28. World Scientific Publishing, Hackensack (2015)
8. Cenzer, D., Hinman, P.G.: Degrees of difficulty of generalized r.e. separating classes. *Arch. Math. Log.* **46**(7–8), 629–647 (2008)
9. Greenberg, N., Miller, J.S.: Lowness for Kurtz randomness. *J. Symb. Log.* **74**(2), 665–678 (2009)
10. Greenberg, N., Miller, J.S.: Diagonally non-recursive functions and effective Hausdorff dimension. *Bull. Lond. Math. Soc.* **43**(4), 636–654 (2011)
11. Khan, M.: Some results on algorithmic randomness and computability-theoretic strength. ProQuest LLC, Ann Arbor, MI, Ph.D. thesis, The University of Wisconsin, Madison (2014)
12. Khan, M., Miller, J.S.: Forcing with bushy trees. *Bull. Symb. Log.* **23**(2), 160–180 (2017)
13. Kjos-Hanssen, B., Merkle, W., Stephan, F.: Kolmogorov complexity and the recursion theorem. *Trans. Am. Math. Soc.* **363**(10), 5465–5480 (2011)
14. Nies, A., Stephan, F., Terwijn, S.A.: Randomness, relativization and Turing degrees. *J. Symb. Log.* **70**(2), 515–535 (2005)
15. Rupperecht, N.: Relativized Schnorr tests with universal behavior. *Arch. Math. Log.* **49**(5), 555–570 (2010)
16. Simpson, S.G.: Turing degrees and muchnik degrees of recursively bounded DNR functions. In: Day, A., Fellows, M., Greenberg, N., Khossainov, B., Melnikov, A., Rosamond, F. (eds.) *Computability and Complexity*. LNCS, vol. 10010, pp. 660–668. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-50062-1_40



Divide and Conquer Computation of the Multi-string BWT and LCP Array

Paola Bonizzoni, Gianluca Della Vedova, Serena Nicosia,
Yuri Pirola, Marco Previtoli^(✉), and Raffaella Rizzi

DISCo, University of Milano-Bicocca, Milan, Italy
`marco.previtoli@disco.unimib.it`

Abstract. Indexing huge collections of strings, such as those produced by the widespread sequencing technologies, heavily relies on multi-string generalizations of the Burrows-Wheeler Transform (BWT) and the Longest Common Prefix (LCP) array, since solving efficiently both problems are essential ingredients of several algorithms on a collection of strings.

In this paper we explore lightweight and parallel computational strategies for building the BWT and LCP array. We design a novel algorithm based on a divide and conquer approach that leads to a simultaneous and parallel computation of multi-string BWT and LCP array.

1 Introduction

In this paper we address the problem of building in external memory the Burrows-Wheeler Transform (BWT) and the Longest Common Prefix (LCP) array for a large collection of strings. Efficient indexing of very large collections of strings is strongly motivated by the widespread use of Next-Generation Sequencing (NGS) technologies that cheaply produce data that fill several terabytes of secondary storage, that has to be analyzed. The Burrows-Wheeler Transform (BWT) [8] is a reversible transformation of a text that was originally designed for text compression (and it's still at the core of the widely used `bzip2` tool). The BWT of a text T is a permutation of T and is strictly related to the Suffix Array of T . In fact, the i^{th} symbol of the BWT is the symbol preceding the i^{th} smallest suffix of T according to the lexicographical order. The Burrows-Wheeler Transform has gained importance beyond its initial purpose, becoming the basis for self-indexing structures such as the FM-index [12], which allows to efficiently search a pattern in a text [12, 15, 18] (see [13] for a recent survey) or in a graph [4] and other crucial tasks on sequences in bioinformatics. Multiple generalization of the BWT to set of sequences [16], trees [11], and graphs [3] have been proposed in the last years. The generalization of the BWT (and the FM-index) to a collection of strings [16] is an ideal tool in genome assembly under the Overlap-Layout-Consensus (OLC) approach [17]. This approach requires the efficient construction of a string graph [17] which is the result of finding all prefix-suffix matches (or overlaps) between reads. For this purpose,

the BWT of a collection of strings allows to achieve time and space efficiency for building a string graph when dealing with a huge amount of available biological data [19]. Indeed, Light String Graph [5] (LSG) is an external memory algorithm for computing the string graph, while Fast String Graph [7] (FSG) is a faster in-memory alternative: both algorithms require the construction of the BWT and LCP array of a collection of strings. We note that most of the approaches for building the BWT and the LCP array of a set of strings usually build them independently or in two successive steps. Since the two data structures are closely related, it makes sense from a theoretical point of view to design methods that build them together to show even more their interconnection.

The construction of the BWT and LCP array of a huge collection of strings is a challenging task and the investigation of possible algorithmic approaches is of theoretical interest, besides its practical application. For this purpose we explore new strategies for computing the BWT and the LCP array of a collection of strings that can be adapted to both external memory and parallel approaches. Current external memory algorithms for the construction of the BWT and the LCP array of a set of strings [1, 2, 9] exploit the disk by storing *partial BWTs* and *partial LCP* arrays that will eventually converge to the BWT and LCP array of the input set in k iterations (where k is the length of the input sequences). Such procedures are hard to parallelize since each element of each partial BWT must be computed after its preceding element in the same data structure.

The algorithm we describe in this paper consists of two phases; the first one is common with our previous algorithm [6], whereas the second phase is different. More precisely, while the strategy in [6] follows the approach of [14] for merging a set of BWTs based on the well-known backward extension operation on a BWT, the strategy of the second phase is based on the opposite forward extension operation on a BWT. This forward extension operation is the novel ingredient of our approach and it allows for a simple parallel implementation of our algorithm. Moreover, our approach is entirely based on linear scans of the data which makes it more amenable to actual disk-based implementations.

2 Preliminaries

Let $\Sigma = \{c_0, \dots, c_\sigma\}$ be a finite alphabet where $c_0 = \$$ (called *sentinel*), and $c_0 < \dots < c_\sigma$ where $<$ specifies the lexicographic ordering over alphabet Σ . We consider a collection $S = \{s_1, \dots, s_m\}$ of m strings, where each string s_j consists of k symbols over the alphabet $\Sigma \setminus \{\$\}$ and is terminated by the symbol $\$$. For ease of presentation and to simplify the presentation of the following methods, we will assume that all the strings in S have the same length. The i^{th} symbol of string s_j is denoted by $s_j[i]$ and the substring $s_j[i]s_j[i+1] \dots s_j[t]$ of s_j is denoted by $s_j[i : t]$. The *suffix* and *prefix* of s_j of length l are the substrings $s_j[k-l+1 : k]$ (denoted by $s_j[k-l+1 :]$) and $s_j[1 : l]$ (denoted by $s_j[: l]$), respectively. Then the *l-suffix* and *l-prefix* of a string s_j are the suffix and prefix with length l , respectively. The lexicographic ordering among the strings in S is defined in the usual way. Although we use the same sentinel to terminate strings, we sort equal

suffixes of different strings by assuming an implicit ordering of the sentinels that is induced by the ordering of the input strings. More precisely, we assume that given $s_i, s_j \in S$, with $i < j$, then the sentinel of s_i is lexicographically smaller than the sentinel of s_j .

Given the lexicographic ordering X of the suffixes of S , the *Suffix Array* is the $(m(k+1))$ -long array SA s.t. $SA[i]$ is equal to (p, j) if and only if the i^{th} element of X is the p -suffix of string s_j . The *multi-string Burrows-Wheeler Transform (BWT)* of S is the $(m(k+1))$ -long array B s.t. if $SA[i] = (p, j)$, then $B[i]$ is the first symbol of the $(p+1)$ -suffix of s_j if $p < k$, or $\$$ otherwise. In other words B consists of the symbols preceding the ordered suffixes of X . The *Longest Common Prefix (LCP) array* of S is the $(m(k+1))$ -long array LCP s.t. $LCP[i]$ is the length of the longest prefix between suffixes $X[i-1]$ and $X[i]$. Conventionally, $LCP[1] = -1$.

Given $n+1$ arrays V_0, V_1, \dots, V_n , an array W is an *interleave* of V_0, V_1, \dots, V_n if W is the result of merging the arrays s.t.: (i) there is a 1-to-1 function ψ_W from the set $\cup_{i=0}^n \{(i, j) : 1 \leq j \leq |V_i|\}$ to the set $\{q : 1 \leq q \leq |W|\}$, (ii) $V_i[j] = W[\psi_W(i, j)]$ for each i, j , and (iii) $\psi_W(i, j_1) < \psi_W(i, j_2)$ for each $j_1 < j_2$.

By denoting with $\mathcal{L} = \sum_{i=0}^n |V_i|$ the total length of the arrays, the interleave W is a \mathcal{L} -long array representing the fusion of the arrays V_0, V_1, \dots, V_n that preserves the relative order of the elements of each array. Hence, for each i with $0 \leq i \leq n$, the j^{th} element of V_i corresponds to the j^{th} occurrence in W of an element of V_i . This fact allows to encode the function ψ_W as an \mathcal{L} -long array I_W s.t. $I_W[q] = i$ if and only if $W[q]$ is an element of V_i . Given I_W , we reconstruct W by noticing that $W[q]$ is equal to $V_{I_W[q]}[j]$, where j is the number of values equal to $I_W[q]$ in the interval $I_W[1, q]$; we refer to this value as the *rank* of the element $I_W[q]$ at position q . In the following, we will refer to vector I_W as *encoding*. Algorithm 1 shows how to reconstruct an interleave from its encoding.

Let l be an integer between 0 and k and let B_l and X_l be m -long arrays s.t. $B_l[i]$ is the symbol preceding the i^{th} smallest l -suffix of S and $X_l[i]$ is the i^{th} smallest l -suffix of S . Note that the BWT B is an interleave of the $k+1$ arrays B_0, B_1, \dots, B_k , since the ordering of symbols in B_l is preserved in B , i.e. B is *stable* w.r.t. each array B_0, B_1, \dots, B_k . This fact is a direct consequence of the definition of B and B_l . For the same reason, the lexicographic ordering X of all suffixes of S is an interleave of the arrays X_0, X_1, \dots, X_k . Let I_B be the encoding of the interleave of arrays B_0, B_1, \dots, B_k and let I_X be the encoding of the interleave of arrays X_0, X_1, \dots, X_k . Then $I_B = I_X$. Therefore, computing either I_B or I_X is equivalent to computing the BWT of the input collection S .

3 The Algorithm

Our algorithm for building the BWT and the LCP array of a set $S = \{s_1, \dots, s_m\}$ of strings with length k (provided as a file on disk) consists of two distinct steps: in the first step the arrays B_0, \dots, B_k are computed, while the second step determines $I_X = I_B$ by implicitly reordering the whole set of suffixes in vectors X_l , thus reconstructing the BWT B as an interleave of B_0, \dots, B_k whose encoding is I_B . At the same time the algorithm computes the LCP array.

Algorithm 1. Reconstruct the interleave W from the encoding I_W

```

1 for  $i \leftarrow 0$  to  $n$  do
2   |  $rank[i] \leftarrow 0$ ;
3 for  $q \leftarrow 1$  to  $|I_W|$  do
4   |  $i \leftarrow I_W[q]$ ;
5   |  $rank[i] \leftarrow rank[i] + 1$ ;
6   |  $W[q] \leftarrow V_i[rank[i]]$ ;

```

The first step requires $\mathcal{O}(mk)$ time. A first preprocessing step splits the sequences in input column-wise in $k + 1$ arrays S_0, \dots, S_k s.t. $S_i[j]$ is equal to the symbol $s_j[i]$. These arrays are used to compute the arrays B_0, \dots, B_k as follows. Array B_0 is trivially the vector of the last characters $s_1[k], \dots, s_m[k]$ of the reads, *i.e.*, it is equal to the array S_k . For $l > 0$, array B_l is a permutation of S_{k-l} and it is computed from B_{l-1} by a bucket sort strategy. The procedure for computing vectors B_1, \dots, B_k for a set of string of equal length is postponed to the full version of the paper due to lack of space. Extending the procedure to handle a set of strings with different lengths is rather simple (see [6]) and causes variable length vectors B_i . A variable length input set is justified, for example, when the strings undergo a trimming step to remove low-quality regions.

The second step of the method computes the interleave $I_X = I_B$ in $\mathcal{O}(kmL)$ time, where L is the length of the longest common substring in the input set, by implicitly sorting the whole set of suffixes of S . The idea is to achieve I_B through L iterations, where each iteration p , from 1 to L , computes the encoding of the interleave of vectors X_l giving the sorting of the suffixes by their first p symbols from the encoding giving the sorting by the first $p-1$ symbols. The first iteration starts from the encoding of the interleave given by the concatenation X_0, \dots, X_k (that is, suffixes in X_0 are followed by suffixes in X_1, \dots , are followed by suffixes in X_k). We can maintain a partial LCP array Lcp_p together with the encoding I_{X^p} , where Lcp_p is the LCP array of the p -prefixes of the suffixes sorted by I_{X^p} . Since L is the length of the longest common substring in the input set of reads, the encoding I_{X^p} computed by the last iteration $p = L$ gives the lexicographic ordering X of the suffixes in S and Lcp_p is the LCP array of S . We point out that the task of implicitly sorting the suffixes of S can be accomplished by following two different strategies, both exploiting a bucket sort approach.

The first strategy [6] is to adopt the approach proposed in [14] for merging a set of BWTs, revisited in [10], to compute also the LCP array together with the BWT. This approach is based on the *backward extension* of the suffixes, where at each iteration p the order of the suffixes sorted by their p -prefix is computed by the order of the suffixes sorted by their $(p-1)$ -suffix by considering the symbol that must be prepended to the latter suffixes to produce the former.

The second strategy is based on the *forward extension* of the $(p-1)$ -prefixes of the suffixes in the ordering given by the encoding I_{X^p} , and is the strategy detailed in this paper. It requires $\mathcal{O}(mkL)$ time where m is the number of input sequences, k is their length, and L is the length of the longest common

string between two sequences in the input set. The main idea is to iteratively group together the suffixes by prefixes of increasing length, similarly to a bucket sort procedure. Nevertheless, storing all the suffixes would require too much space and we provide efficient algorithms to induce the order of the suffixes from the arrays B_l computed in the previous step.

The rest of this section is laid out as follows. In Sect. 3.1 we describe how to compute the interleave of arrays B_l , *i.e.*, the core of the second step of the method proposed in this article. In Sect. 3.2 we extend the previous procedure to compute the LCP array of the input sequence and provide an efficient algorithm to compute arrays Q_l^p that are fundamental for this step. For space constraint, the first step of the method, *i.e.*, the computation of arrays B_l , is deferred to the full version of the paper since it is similar to the one in [6].

3.1 Computing the Interleave

The second step of our algorithm computes the encoding I_X of the interleave X of the arrays X_0, X_1, \dots, X_k , giving the lexicographic ordering of all suffixes of S and at the same time computes the LCP array. Recall that I_X is equal to the encoding I_B of the interleave of the arrays B_0, B_1, \dots, B_k giving the BWT B . This section is devoted to describe how to compute I_X from which it is easy to obtain the BWT B as explained in Algorithm 1, while the description of the approach to obtain the LCP array is postponed until Sect. 3.2. Before entering into the details, we provide some fundamental definitions.

Definition 1. Let $\alpha = s_{i_\alpha}[k - l_\alpha + 1 :]$ and $\beta = s_{i_\beta}[k - l_\beta + 1 :]$ be two generic suffixes of S , with length respectively l_α and l_β . Then, given an integer p , $\alpha \prec_p \beta$ (and we say that α p -precedes β) *iff* one of the following conditions hold: (1) $\alpha[: p]$ is lexicographically strictly smaller than $\beta[: p]$, (2) $\alpha[: p] = \beta[: p]$ and $l_\alpha < l_\beta$, (3) $\alpha[: p] = \beta[: p]$, $l_\alpha = l_\beta$ and $i_\alpha < i_\beta$.

Definition 2. Given the arrays X_0, X_1, \dots, X_k , the p -interleave X^p ($0 \leq p \leq k$) is the interleave *s.t.* $X^p[i]$ is the i^{th} smallest suffix in the \prec_p -ordering of all the suffixes of S .

It is immediate to verify that X^k (that is, the suffixes sorted according to the \prec_k relation) is equal to X , hence $I_X = I_{X^k}$. Our approach determines I_{X^k} by iteratively computing I_{X^p} by increasing values of p , starting from I_{X^0} . Observe that X^0 lists the suffixes in the same order given by the concatenation of arrays X_0, X_1, \dots, X_k and the encoding I_{X^0} is trivially given by $|X_0|$ 0s, followed by $|X_1|$ 1s, \dots , followed by $|X_k|$ values equal to k .

Definition 3. Let X^p be the p -interleave of X_0, X_1, \dots, X_k , and let i be a position. Then, the p -segment of i in X^p is the maximal interval $[b, e]$ *s.t.* $b \leq i \leq e$ and all suffixes in $X^p[b, e]$ have the same p -prefix. Positions b and e are called respectively begin and end position of the segment, and the common p -prefix is denoted by $w_p(b, e)$.

It is immediate to observe that the set of all the p -segments of a p -interleave form a partition of its positions $(1, \dots, (k + 1)m)$. Observe that, by definition, a suffix smaller than p belongs to a p -segment $[b, e]$ having $b = e$. In other words, such suffix is the unique element of the p -segment.

Before describing the approach, the computation of X^p from X^{p-1} is explained. Let Q_l^p be the m -long array s.t. $Q_l^p[i]$ is the p^{th} symbol of the suffix $X_l[i]$ if $p \leq l$, or $\$$ otherwise. Moreover, let Q^p be the interleave of the arrays $Q_0^p, Q_1^p, \dots, Q_k^p$ s.t. $I_{Q^p} = I_{X^{p-1}}$. In other words, $Q^p[i]$ is the p^{th} symbol of the suffix $X^{p-1}[i]$. The following lemma shows that each p -segment $[b_p, e_p]$ can be computed from the unique $(p - 1)$ -segment $[b_{p-1}, e_{p-1}]$ s.t. $b_{p-1} \leq b_p$ and $e_p \leq e_{p-1}$.

Lemma 4. *Let $[b, e]$ be a $(p - 1)$ -segment of X^{p-1} . Then, $X^p[b, e]$ is a permutation of $X^{p-1}[b, e]$ defined by the permutation $\Pi_{b,e}^{p-1}$ of the indexes $(b, b + 1, \dots, e)$ producing the stable ordering of the symbols in $Q^p[b, e]$, s.t. the r^{th} suffix of $X^p[b, e]$ is the suffix of X^{p-1} in position $\Pi_{b,e}^{p-1}[r]$.*

Given the suffix in position i of X^{p-1} , s.t. i is in the $(p - 1)$ -segment $[b, e]$, Lemma 4 allows to compute its position $i' \in [b, e]$ on X^p . Let $\#^<$ be the number of symbols of $Q^p[b, e]$ that are strictly smaller than $Q^p[i]$ and let $\#_q^=$ be the number of symbols of $Q^p[b, q]$ which are equal to $Q^p[i]$. Then, the rank of suffix $X^{p-1}[i]$ in $X^p[b, e]$ is $r = \#^< + \#_q^=$, thus deriving that its position in X^p is $i' = b + r - 1$. Note that the positions $(b, b + 1, \dots, e)$ on X^p are partitioned into n p -segments $[b_1, e_1], \dots, [b_n, e_n]$ (referred as *induced* by the $(p - 1)$ -segment $[b, e]$ of X^{p-1}), where n is the number of distinct non- $\$$ symbols in $Q^p[b, e]$ plus the number $\#_\$$ of symbols $\$$ in $Q^p[b, e]$. Observe that the first $\#_\$$ p -segments $[b_1, e_1], \dots, [b_{\#_\$}, e_{\#_\$}]$ have width 1, while the width of the last $n - \#_\$$ p -segments $[b_{\#_\$+1}, e_{\#_\$+1}], \dots, [b_n, e_n]$ can be computed as follows. Let $\{c_1, \dots, c_{n-\#_\$}\}$ be the ordered set of the distinct non- $\$$ symbols in $Q^p[b, e]$. Then, the width of $[b_{\#_\$+i}, e_{\#_\$+i}]$ ($1 \leq i \leq n - \#_\$$) is equal to the number of occurrences of the symbol c_i in $Q^p[b, e]$. From what described above, it derives that the p -segments on X^p form a partition of its positions $(1, \dots, (k + 1)m)$ that is a refinement of the partition formed by the $(p - 1)$ -segments on X^{p-1} .

Algorithm 2. Compute $X^p[b, e]$ from the $(p - 1)$ -segment $[b, e]$ on X^{p-1}

```

1  $L_{c_0}, L_{c_1}, \dots, L_{c_\sigma} \leftarrow$  empty lists;
2 for  $i \leftarrow b$  to  $e$  do
3    $c \leftarrow Q^p[i]$ ;
4   Append  $X^{p-1}[i]$  to  $L_c$ ;
5  $r \leftarrow 1$ ;
6 for  $h \leftarrow 0$  to  $\sigma$  do
7   for  $r_h \leftarrow 1$  to  $|L_{c_h}|$  do
8      $X^p[b + r - 1] \leftarrow L_{c_h}[r_h]$ ;
9      $r \leftarrow r + 1$ ;
```

Algorithm 3. Compute I_{X^p} from $I_{X^{p-1}}$

```

1  $L_{c_0}, L_{c_1}, \dots, L_{c_\sigma} \leftarrow$  empty lists;
2  $rank \leftarrow$  a vector of  $k$  zeroes;
3  $pick\_up\_start \leftarrow true$ ;
4 for  $i \leftarrow 1$  to  $(k+1)m$  do
5     if  $pick\_up\_start = true$  then
6          $b \leftarrow i$ ;
7          $pick\_up\_start = false$ ;
8      $j \leftarrow I_{X^{p-1}}[i]$ ;
9      $rank[j] \leftarrow rank[j] + 1$ ;
10     $c \leftarrow Q_j^p[rank[j]]$ ;
11    Append  $j$  to  $L_c$ ;
12    if  $i$  is the end position of a  $(p-1)$ -segment then
13         $pick\_up\_start = true$ ;
14         $r \leftarrow 1$ ;
15        for  $h \leftarrow 0$  to  $\sigma$  do
16            for  $r_h \leftarrow 1$  to  $|L_{c_h}|$  do
17                 $j \leftarrow L_{c_h}[r_h]$ ;
18                 $I_{X^p}[b+r-1] \leftarrow j$ ;
19                if  $r_h > 1$  and  $h > 0$  then
20                     $Lcp_p[b+r-1] = p$ ;
21                else
22                     $Lcp_p[b+r-1] = Lcp_{p-1}[b+r-1]$ ;
23                 $r \leftarrow r + 1$ ;
24     $L_{c_0}, L_{c_1}, \dots, L_{c_\sigma} \leftarrow$  empty lists;

```

Algorithm 2 describes a simple procedure to compute $X^p[b, e]$ from the $(p-1)$ -segment $[b, e]$ of X^{p-1} . The procedure uses $\sigma+1$ lists $L_{c_0}, \dots, L_{c_\sigma}$ initially empty. Each position $i \in [b, e]$ is considered and each suffix $X^{p-1}[i]$ is appended to the list L_c s.t. c is the p^{th} symbol of $X^{p-1}[i]$. Afterwards, the new order of the elements in $X^{p-1}[b, e]$ provided by the concatenation of the lists $L_{c_0}, \dots, L_{c_\sigma}$ is copied in $X^p[b, e]$

Algorithm 2 can be easily modified in order to produce also the p -segments $[b, e_1], [b_2, e_2], \dots, [b_n, e]$ induced by the $(p-1)$ -segment $[b, e]$ on I_{X^p} without directly storing X . The entire interleave I_{X^p} is obtained by computing $I_{X^p}[b, e]$ for each distinct $(p-1)$ -segment $[b, e]$ of $I_{X^{p-1}}$. Algorithm 3 is an iterative procedure that computes the encoding I_{X^p} from $I_{X^{p-1}}$. Even though the description is sequential, it is possible to give a parallel version, since all $(p-1)$ -segments on $I_{X^{p-1}}$ can be managed independently to obtain the induced p -segments.

All the k -segments of the encoding I_{X^k} have width equal to 1. Moreover, if L is the length of the longest common substring of two strings in S , after L iterations the two following properties hold: (1) the encoding I_{X^L} is equal to I_{X^k} , and (2) each I_{X^j} with $j > L$ is identical to I_{X^L} . These two properties are a consequence of the following two observations: (i) the length p of the longest common prefix between two strings is equal to the length of the longest

Algorithm 4. Compute all lists Q_l^p for any given $p \geq 2$.

Input : The lists B_0, \dots, B_k on alphabet c_0, \dots, c_σ , an integer p with $2 \leq p \leq k$, and all Q_l^{p-1} .

Output: The lists Q_l^p for each $k \geq l \geq p$

```

1 for  $l \leftarrow p$  to  $k$  do
2    $Q_l^p \leftarrow$  empty list;
3   for  $h \leftarrow 0$  to  $\sigma$  do
4      $Q_l^p(c_h) \leftarrow$  empty list;
5   for  $j \leftarrow 1$  to  $m$  do
6     Append  $Q_{l-1}^{p-1}[j]$  to  $Q_l^p(B_{l-1}[j])$ ;
7   for  $h \leftarrow 0$  to  $\sigma$  do
8     Append  $Q_l^p(c_h)$  to  $Q_l^p$ ;

```

common substring in S , and (ii) if all the $(p+1)$ -prefixes of the suffixes are distinct, then the \prec_{p+1} relation does not affect the ordering given by I_{X^p} , that is $I_{X^{p+1}} = I_{X^p}$. Finally, notice that all p -segments are independent, therefore they can be computed in parallel from the $(p-1)$ -segments.

3.2 Computing the LCP Array and the Q_l^p Arrays

The LCP array is obtained by exploiting Proposition 5 which follows from the definition of p -segment.

Proposition 5. *Let i be a position on the Longest Common Prefix array LCP. Then $LCP[i]$ is the largest p s.t. i is the start of a $(p+1)$ -segment (of $I_{X^{p+1}}$) and is not the start of a p -segment (of I_{X^p}).*

Let Lcp_p be the $((k+1)m)$ -long array s.t. $Lcp_p[i]$ is the length of the longest common prefix between the p -prefix of suffix $X^p[i]$ and the p -prefix of suffix $X^p[i-1]$. The array Lcp_k is equal to the LCP array of the input set S .

Algorithm 3 exploits Proposition 5 to compute the LCP array iteratively along with the interleave I_X . More precisely, at each iteration p , Algorithm 3 computes Lcp_p from Lcp_{p-1} by increasing each entry of Lcp_{p-1} that is not the starting position of a p -segment. Since the LCP between two empty string is 0, the array Lcp_0 is set to all 0s (apart from the first position that is set to -1) before the first iteration. The following invariant, which directly implies the correctness of the procedure, is maintained at each iteration.

Lemma 6. *At the end of iteration p , $Lcp_p[i] = p$ iff i is not the start position of any p -segment.*

We now show how to compute the arrays Q_0^p, \dots, Q_k^p used by iteration p of Algorithm 2. Recall that Q_l^p is the m -long array s.t. $Q_l^p[i]$ is the p^{th} symbol of the i^{th} smallest l -suffix of X_l if $p \leq l$, or $\$$ otherwise. The following proposition establishes a recursive definition of Q_l^p .

Algorithm 5. Computation of the interleave

Input : The arrays B_0, B_1, \dots, B_k
Output: The encoding I_{X^k} .

- 1 **for** $l \leftarrow 0$ **to** k **do**
- 2 **for** $i \leftarrow 1$ **to** m **do**
- 3 $I_{X^0}[lm + i] \leftarrow l$; $Lcp[lm + i] \leftarrow 0$;
- 4 Compute lists Q_l^1 for $0 \leq l \leq k$;
- 5 $p \leftarrow 1$;
- 6 **while** *there exists some $(p - 1)$ -segment on $I_{X^{p-1}}$ which is wider than 1* **do**
- 7 Compute I_{X^p} from $I_{X^{p-1}}$;
- 8 Compute lists Q_l^{p+1} for $0 \leq l \leq k$;
- 9 **Output** I_{X^p} ;

Proposition 7. *Let X_l and X_{l-1} be respectively the sorted l -suffixes and $(l-1)$ -suffixes of the set S . Let α_l and α_{l-1} be respectively the l -suffix and the $(l-1)$ -suffix of a generic input string s_i . Then the p^{th} symbol of α_l is the $(p-1)^{\text{th}}$ symbol of α_{l-1} .*

Algorithm 4 shows how to compute all the lists Q_l^p iteratively from Q_{l-1}^{p-1} exploiting Proposition 7. We note that, for $l \geq 1$, Q_l^1 is the result of sorting B_{l-1} whereas Q_0^1 is a sequence of sentinels. Therefore the arrays Q_0^1, \dots, Q_k^1 can be trivially computed and form the initial step of the recursion.

In order to prove the correctness of Algorithm 4 we need to show that the permutation St^{l-1} over indexes $1, \dots, m$ of B_{l-1} induced by the lexicographic ordering of B_{l-1} , is the correct permutation of Q_{l-1}^{p-1} to obtain Q_l^p . Indeed, observe that St^{l-1} is the permutation that relates positions of indexes of strings in X_{l-1} to their positions in X_l . More precisely, given a string s_q of S , s.t. its $(l-1)$ -suffix is in position j of list X_{l-1} , then if $St^{l-1}[j] = t$, it means that the l -suffix is of the string s_q is in position t of list X_l . The above observation is a consequence of the fact that in order to get the lexicographic ordering of X_l from the list X_{l-1} we simply sort the $(l-1)$ -suffixes by the first symbol that precedes them, i.e., they are sorted by the list B_{l-1} .

Finally, Algorithm 5 shows how to combine Algorithms 3 and 4 in order to compute I_{X^k} from the input arrays B_0, \dots, B_k .

4 Time Complexity and Conclusions

The overall time complexity of the method is $\mathcal{O}(kmL)$. Trivially, computing arrays S_l and B_l requires $\mathcal{O}(mk)$ (see [6] for a more thorough analysis). The second step (Algorithm 5) requires $\mathcal{O}(kmL)$ time since initializing the support data structures (lines 1–5) and each call in lines 7 and 8 require $\mathcal{O}(km)$. Moreover, as said before, the **while** loop at lines 6–8 is executed L times, where L is the length of the longest common substring in the input set.

In this paper, we presented a new algorithm to compute the BWT and the LCP array of a set of strings, that is suitable for either parallel or external memory approaches. Notice that all the arrays are accessed sequentially, therefore they can be stored in external files and it is immediate to view our procedure as an external memory approach, where only the arrays S_l ($0 \leq l \leq k - 1$) of the symbols of the input strings are kept in main memory (together with some additional data structures). The overall I/O complexity of the algorithm is $O(mkL(\log k + \log \sigma))$ (detailed in the full version of the paper). Moreover, we note that Algorithms 3 and 4 are suitable for parallel approaches and that these two algorithms are the fundamental steps of Algorithm 5, *i.e.*, the main approach proposed in this article. Finally, we notice that our time complexity ($\mathcal{O}(mkL)$) is no worse than the one of the best known lightweight approaches.

References

1. Bauer, M.J., Cox, A.J., Rosone, G.: Lightweight algorithms for constructing and inverting the BWT of string collections. *Theor. Comp. Sci.* **483**, 134–148 (2013)
2. Bauer, M.J., Cox, A.J., Rosone, G., Sciortino, M.: Lightweight LCP construction for next-generation sequencing datasets. In: Raphael, B., Tang, J. (eds.) WABI 2012. LNCS, vol. 7534, pp. 326–337. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33122-0_26
3. Belazzougui, D., Gagie, T., Mäkinen, V., Previtalli, M., Puglisi, S.J.: Bidirectional variable-order de Bruijn graphs. In: Kranakis, E., Navarro, G., Chávez, E. (eds.) LATIN 2016. LNCS, vol. 9644, pp. 164–178. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49529-2_13
4. Beretta, S., Bonizzoni, P., Denti, L., Previtalli, M., Rizzi, R.: Mapping RNA-seq data to a transcript graph via approximate pattern matching to a hypertext. In: Figueiredo, D., Martín-Vide, C., Pratas, D., Vega-Rodríguez, M.A. (eds.) AlCoB 2017. LNCS, vol. 10252, pp. 49–61. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58163-7_3
5. Bonizzoni, P., Della Vedova, G., Pirola, Y., Previtalli, M., Rizzi, R.: LSG: an external-memory tool to compute string graphs for next-generation sequencing data assembly. *J. Comput. Biol.* **23**(3), 137–149 (2016)
6. Bonizzoni, P., Della Vedova, G., Pirola, Y., Previtalli, M., Rizzi, R.: Computing the BWT and LCP array of a set of strings in external memory. *CoRR* abs/1705.07756 (2017). <http://arxiv.org/abs/1705.07756>
7. Bonizzoni, P., Della Vedova, G., Pirola, Y., Previtalli, M., Rizzi, R.: FSG: fast string graph construction for de novo assembly. *J. Comput. Biol.* **24**(10), 953–968 (2017)
8. Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm. Technical report, Digital Systems Research Center (1994)
9. Cox, A.J., Garofalo, F., Rosone, G., Sciortino, M.: Lightweight LCP construction for very large collections of strings. *J. Discrete Algorithms* **37**(C), 17–33 (2016)
10. Egidi, L., Manzini, G.: Lightweight BWT and LCP merging via the gap algorithm. In: Fici, G., Sciortino, M., Venturini, R. (eds.) SPIRE 2017. LNCS, vol. 10508, pp. 176–190. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67428-5_15
11. Ferragina, P., Luccio, F., Manzini, G., Muthukrishnan, S.: Compressing and indexing labeled trees, with applications. *J. ACM* **57**(1), 4:1–4:33 (2009)
12. Ferragina, P., Manzini, G.: Indexing compressed text. *J. ACM* **52**(4), 552–581 (2005)

13. Gagie, T., Manzini, G., Sirén, J.: Wheeler graphs: a framework for BWT-based data structures. *Theor. Comput. Sci.* **698**, 67–78 (2017)
14. Holt, J., McMillan, L.: Merging of multi-string BWTs with applications. *Bioinformatics* **30**(24), 3524–3531 (2014)
15. Li, H.: Fast construction of FM-index for long sequence reads. *Bioinformatics* **30**(22), 3274–3275 (2014)
16. Mantaci, S., Restivo, A., Rosone, G., Sciortino, M.: An extension of the Burrows-Wheeler transform. *Theor. Comput. Sci.* **387**(3), 298–312 (2007)
17. Myers, E.: The fragment assembly string graph. *Bioinformatics* **21**(suppl. 2), ii79–ii85 (2005)
18. Rosone, G., Sciortino, M.: The Burrows-Wheeler transform between data compression and combinatorics on words. In: Bonizzoni, P., Brattka, V., Löwe, B. (eds.) *CiE 2013. LNCS*, vol. 7921, pp. 353–364. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39053-1_42
19. Simpson, J., Durbin, R.: Efficient construction of an assembly string graph using the FM-index. *Bioinformatics* **26**(12), i367–i373 (2010)



Some Observations on Infinitary Complexity

Merlin Carl^(✉)

Fachbereich Mathematik und Statistik, Universität Konstanz, Konstanz, Germany
merlin.carl@uni-konstanz.de

Abstract. Continuing the study of complexity theory of Koepke’s Ordinal Turing Machines (OTMs) that was done in [CLR], we prove the following results:

1. An analogue of Ladner’s theorem for OTMs holds: That is, there are languages \mathcal{L} which are NP^∞ , but neither P^∞ nor NP^∞ -complete. This answers an open question of [CLR].
2. The speedup theorem for Turing machines, which allows us to bring down the computation time and space usage of a Turing machine program down by an arbitrary positive factor under relatively mild side conditions by expanding the working alphabet does not hold for OTMs.
3. We show that, for $\alpha < \beta$ such that α is the halting time of some OTM-program, there are decision problems that are OTM-decidable in time bounded by $|w|^\beta \cdot \gamma$ for some $\gamma \in \text{On}$, but not in time bounded by $|w|^\alpha \cdot \gamma$ for any $\gamma \in \text{On}$.

1 Introduction

After the introduction of Infinite Time Turing Machines (ITTMs) in [HL] and the subsequent development of various other infinitary machine models of computation e.g. in [wITRM, KS, ITRM, OTM, ORM], analogues of several central topics in classical computability theory were developed for these machine types, among them degree theory [W1], computable model theory [HMS], randomness [CS, C14, CS2] and complexity theory. Complexity theory was first studied by Schindler in the case of ITTMs, who proved that $\text{P} \neq \text{NP}$ for ITTMs [Schindler], which was later refined in various ways [DHS, HW]. Results on the space complexity for infinitary computations were given by Winter in [Wi1, Wi2]. It was occasionally remarked that complexity theory for ITTMs is somewhat unsatisfying due to the fact that all inputs for ITTMs have the same length, namely ω .

This motivated the consideration of complexity theory for ‘symmetrical’ models that have the same amount of time and space available, the most prominent of which are Koepke’s ‘Ordinal Turing Machines’ (OTMs), which can be thought of as Turing machines with a tape of proper class length On and unbounded ordinal working time. For an introduction to OTMs, we refer to [OTM]. In

agreement with the theory of classical Turing machines, we explicitly allow multitape OTMs, i.e. OTMs with any finite number of scratch tapes.¹ The study of complexity theory for these machines was started by Löwe in [L]. After this, the subject lay dormant for a while, until it was revived by Löwe and Rin at the CiE 2016, which led to [CLR]. The central contributions of that paper were the introduction of natural infinitary analogues of the classes P and NP, called P^∞ and NP^∞ and of the satisfaction problem SAT for OTMs, called SAT^∞ , the proof of a corresponding Cook-Levin theorem showing that SAT^∞ is NP^∞ -complete, and the proof that SAT^∞ (and hence any other NP^∞ -complete problem) is in fact not OTM-computable (while its restriction to constructibly countable formulas is).

Among the questions left open in [CLR] was whether there is an analogue of Ladner's theorem for OTMs, i.e. whether there are problems in $NP^\infty \setminus P^\infty$ that are not NP^∞ -complete.

In this paper, we respond to this question by showing that the OTM-analogue of Ladner's theorem holds. We then use the same proof idea to show that the hierarchy of OTM-decision problems decidable with time bound $|w|^\alpha \cdot \gamma$, where $|w|$ is the input length, γ is fixed ordinal, and α is the halting time of an OTM-program, is strictly increasing in α . We also show that there is no analogue of the speedup-theorem for Ordinal Turing Machines.

2 Preliminaries

We start by explaining the notations and giving the results that will be used in the course of this paper. Those which are not folklore can be found in [CLR]. For the definition of the complexity classes P^∞ , NP^∞ as well as the problem SAT^∞ , we also refer to [CLR].

We say that a structure (S, E) with $E \subseteq S \times S$, is coded by $c \subseteq \text{On}$ if and only if there is some bijection $f : \gamma \rightarrow S$, $\gamma \in \text{On}$ and $c := \{p(\iota_1, \iota_2) : f(\iota_1)E f(\iota_2)\}$, where p is Cantor's pairing function.

Let $\{0, 1\}^{**}$ denote the class of all ordinal-length binary strings. That is, $\{0, 1\}^{**}$ is the class of functions with domain an ordinal and codomain $\{0, 1\}$. For $x \in \{0, 1\}^{**}$, let $|x|$ denote the length of x , i.e. its domain. A 'language' or 'decision problem' is now simply a subclass X of $\{0, 1\}^{**}$.

For an ordinal β , denote by β_0 and β_1 the summand $< \omega$ and the rest, respectively, when β is written in Cantor normal form, i.e. $\beta = \omega\beta_1 + \beta_0$, $\beta_0 < \omega$. In this way, every ordinal naturally corresponds to a pair consisting of a multiple of ω and a natural number.

Let $(P_i : i \in \omega)$ enumerate the OTM-programs in some natural way.

The following is the main result of [CLR]:

¹ Multitape OTMs can be simulated by single-tape OTMs quickly enough so that this does not make a difference for Theorem 2, our main result. However, the proofs of Theorem 3 and Corollary 2 both make crucial use of the availability of several tapes.

Theorem 1. *The satisfaction problem for infinitary propositional formulas (conjunctions and disjunctions of any ordinal length are allowed) is NP^∞ -complete. At the same time, it is OTM-undecidable.*

Proof. See [CLR], Theorems 4, 5 and 10.

3 An Analogue of Ladner's Theorem

Ladner's theorem answers the question whether any NP-problem that is not in P is already NP-complete. Of course, this is trivially true if it should happen that $P = NP$. Thus, Ladner's theorem is stated in a conditional form [FG]: If $P \neq NP$, there is $A \in NP \setminus P$ such that A is not NP-complete. We will now show that an analogous result holds for OTMs. Since we know that $P^\infty \neq NP^\infty$ from Theorem 1, we can state it unconditionally. Also by Theorem 1, it will suffice for proving an infinitary analogue of Ladner's theorem to find a decidable problem in $NP^\infty \setminus P^\infty$. Such a problem will now be constructed by a diagonalization.

Such a problem will now be constructed by a diagonalization. We start with some preliminary results that will be helpful in the construction. In the following, KP denotes Kripke-Platek set theory, see e.g. [B].

Lemma 1. *If $M \models KP$, then the well-founded part of M is admissible and thus closed under ordinal exponentiation.*

Proof. See [B], Corollary 8.5.

Proposition 1. *For each infinite ordinal α , there is a subset of α^2 that codes a model of KP that has α in its well-founded part and moreover codes $\iota \in \alpha$ by ι .*

Proof. This can be achieved by first observing that there must be such a model of the right cardinality (form the elementary hull of $\alpha + 1$ in some $L_\beta \models KP$ with $\beta > \alpha$, then use condensation) and then re-organizing the code, if necessary.

Lemma 2. *Checking whether some $x \subseteq \alpha$ codes a model of a certain first-order sentence ϕ is possible in polynomial time in α , in fact in time α^ω (in fact in time α^n when ϕ contains n quantifiers and occurrences of \in). Hence, checking whether a subset of an ordinal α codes a model of KP is possible in time α^ω , which is still polynomial in α .*

Proof. This is done by exhaustively searching through the code for every quantifier and evaluating the logical connectives in the obvious way (technically, evaluating the relation needs another searching through the code, which is the reason for the exponent mentioned above).

We refer to [OTM], Lemma 6.1 for a more detailed description of the algorithm.

Theorem 2. *There is a decision problem $X \subseteq \{0, 1\}^{**}$ which is NP^∞ , but neither P^∞ nor NP^∞ -complete. In fact, $X \in NP^\infty \setminus P^\infty$ can be chosen to be OTM-decidable.*

Proof. We construct such a problem $X \subseteq \{0, 1\}^{**}$ by diagonalization. Let $X := \{x \in \{0, 1\}^{**} : P_{|x|_0}(x) \text{ does not halt in } \leq |x|_1^{|x|_1} |x|_1 \text{ many steps or does halt in that many steps but rejects (i.e. outputs 0)}\}$.

It is easy to see that X is OTM-decidable: Given $x \in \{0, 1\}^{**}$, simply simulate $P_{|x|_0}(x)$ for $|x|_1^{|x|_1} |x|_1$ many steps and then flip the output (i.e. accept if the simulated computation rejects or does not halt, otherwise reject).

It is also clear that X is not in P^∞ : If P_k was an OTM-program that decides X in time $\leq |x|^\alpha \beta$, let $x \in \{0, 1\}^{**}$ be such that $|x| > \max\{\alpha, \beta\}$ and $|x|_0 = k$; then $P_k(x)$ will give the wrong result by definition of X .

It remains to see that X is in NP^∞ . Consider the class

$$X' := \{(x, y) : x, y \in \{0, 1\}^{**} \wedge 'y \subseteq |x|^2 \text{ codes a KP-model } M \text{ with well-founded part of height } > |x|' \wedge 'M \text{ believes that } P_{|x|_0}(x) \text{ does not halt in } \leq |x|_1^{|x|_1} |x|_1 \text{ many steps or does halt in } \leq |x|_1^{|x|_1} |x|_1 \text{ many steps but rejects (i.e. outputs 0)}'\}.$$

Such a code exists by Proposition 1 above.

The statement just given is a first-order statement in the parameter x and can be evaluated in time polynomial in $|x| + |y|$, which is polynomial in $|x|$ as $|y| \leq |x|^2$ by assumption. By Lemma 1, the computation within M will belong to the well-founded part of M , and thus be an actual computation in V , so that M will be correct about the result.

Hence X' belongs to P^∞ , and X , as the projection of X' to the first component, belongs to NP^∞ .

Thus X is indeed an OTM-decidable (and thus NP^∞ -incomplete) problem in $NP^\infty \setminus P^\infty$, so X is as desired.

It was shown in [CLR] that $P^\infty \neq NP^\infty$. The above proof, only depending on the fact that KP-models are closed under ordinal polynomials, actually shows much more. For example, let us say that a class $X \subseteq \{0, 1\}^{**}$ is $EXPTIME^\infty$ if and only if there are an OTM-program P , an ordinal α and an ordinal polynomial p such that P decides X and works for $\leq \alpha^{p(\beta)}$ many steps on an input of length β . Similarly, let X be $EXPEXPTIME^\infty$ if and only if this works with time bound $\alpha^{\alpha^{p(\beta)}}$. Then, by the same argument, one also obtains:

Corollary 1. *$EXPTIME^\infty$ and $EXPEXPTIME^\infty$ are properly contained in NP^∞ .*

4 Speedup and Strictness of the ∞ -exponent Time Hierarchy

A well-known theorem from classical complexity theory is the speedup-theorem, see e.g. [Hro]. This theorem says that, under certain mild conditions about the function f , if $0 < c < 1$ and there is a Turing program for deciding a certain language within time or space bounded by f in the length of the input, then there is another Turing program deciding this language in time or space bounded by

cf in the length of the input. This observation is crucial for classical complexity theory, as it justifies the introduction of $O(f)$ -classes for measuring complexities. The proof idea is to let the new machine work on a considerably enriched alphabet, in which long strings of symbols of the original alphabet are condensed into one symbol and thus processed in much fewer steps.

It is rather obvious that this approach will not work for infinitary machines: First, the alphabet is restricted to $\{0, 1\}$; however, this is a formal limitation that could be overcome by slight changes in the definition. More importantly, such a compression of the alphabet will not have much of an effect, since a finite time compression will not reduce the working time when it is a limit ordinal.

In fact, there are speedup theorems also for infinitary machines, such as the speedup-theorem for Infinite Time Turing Machines by Hamkins and Lewis (see [HL]). These appear in the context of clockable ordinals, but they give in a sense only a much weaker speedup: Namely, if there is a program P that halts in $\alpha + n$ many steps and $1 < n \in \omega$, then there is a program P' that halts in $\alpha + 1$ many steps. Note that this statement makes no reference to decision problems.

We will now show that there is in fact no analogue of the classical speedup-theorem for OTMs by showing that there is a decision problem that is solvable in running time $\alpha \cdot 4$, but not in running time $\alpha \cdot 2$.

Theorem 3. *There is a decision problem $\mathcal{L} \subseteq \Sigma^{**}$ such that \mathcal{L} is decidable in running time $\alpha \cdot 4$, but not in running time $\alpha \cdot 2$ (where α denotes the length of the input).*

Proof. We prove this by diagonalization. To this end, we consider the language $\mathcal{L} \subseteq \{0, 1\}^{**}$, where $w \in \{0, 1\}^{**}$ belongs to \mathcal{L} if and only if the following holds: Let w' be the initial segment of w of length ω . Let $i = 0$ if w' consists entirely of 1s, and let i be the length of the longest initial segment of w' consisting entirely of 1s otherwise. Now run P_i on input w for $|w| \cdot 2$ many steps. If the output is 1, output 0, otherwise (i.e. if the output is different from 1 or there is no output as the program didn't halt in that time) output 1.

Claim 1: \mathcal{L} is not OTM-decidable in time complexity $\alpha \cdot 2$.

For suppose that P_j was a program that decides \mathcal{L} in running time bounded by $|w| \cdot 2$. Consider a word w of the form $w = \underbrace{11\dots 1}_{j \times} 0v$ with $|v| > \omega$. If P_j with input w does not halt in $< |w| \cdot 2$ many steps, it does not decide \mathcal{L} in the desired running time. Otherwise, its output will be wrong by definition of \mathcal{L} .

Claim 2: \mathcal{L} is OTM-decidable in time complexity $\alpha \cdot 4$.

To see this, we use a multitape OTM, i.e. an OTM with multiple (but finitely many) tapes. Given w , we run through the first ω many symbols to determine i . Then, we write the i th OTM-program to an extra tape, which will later on direct the simulation of P_i on input w . Further, we write $|w|$ many 1s to the initial

segments of each of two extra tapes T_0, T_1 , filling the rest with 0s; these will serve as a ‘stopwatch’ for our simulation. Now simulate P_i on w ; each simulation step will only take a bounded finite number c of computation steps, which depends only on i . For each simulation step, move the head to the right first on T_0 and, when the head arrives at the first 0 of T_0 , continue on T_1 ; when the right border of T_1 has been reached, stop the simulation.

The first phase needs $\omega + \alpha$ many steps, which is α for α sufficiently large (i.e. $\alpha \geq \omega^2$). The simulation then takes $c\alpha \cdot 2$ many steps. Writing $\alpha = \omega\alpha' + k$, $k \in \omega$, we have $c(\omega\alpha' + k) \cdot 2 = (c\omega\alpha' + ck) \cdot 2 = (\omega\alpha' + ck) \cdot 2 = \omega\alpha' \cdot 2 + ck \cdot 2 < \alpha \cdot 4$, so that we get $< \alpha \cdot 4$ many steps in total.

Remark: As one can easily see from inspecting the proof, neither the choice of the constants nor of the function $\alpha \mapsto \alpha$ instead of e.g. $\alpha \mapsto \alpha^2$ makes a difference.

We now define a rather natural hierarchy on the ∞ -polynomially decidable decision problems.

For $\alpha \in \text{On}$, let us say that a class $X \subseteq \{0, 1\}^{**}$ is \mathcal{P}_α if and only if there is an OTM-program Q and $\beta \in \text{On}$ such that Q decides X and takes fewer than $\gamma^\alpha \beta$ many steps on an input of length γ .

Clearly, every \mathcal{P}_α -class is also \mathcal{P}_δ for $\alpha \leq \delta$. The \mathcal{P}_α -classification is thus a stratification of the class P^∞ , which we name the ∞ -exponent time hierarchy.

An easy adaptation of the argument used for Theorem 3 yields:

Corollary 2. *The ∞ -exponent time hierarchy is strict: If $\alpha < \beta$ and α is the halting time of some OTM-program, there is a decision problem $X \subseteq \{0, 1\}^{**}$ which is \mathcal{P}_β , but not \mathcal{P}_α .*

Proof. Given w , define $i = i(w)$ as in the proof of Theorem 3. Now run P_i on input w for $|w|^\alpha |w|$ many steps. If P_i halts in that many steps with output 1, then let $w \notin \mathcal{L}$, otherwise $w \in \mathcal{L}$.

By the usual argument, \mathcal{L} is not decidable in time bounded by the function $x^\alpha \cdot \gamma$ for any $\gamma \in \text{On}$: To see this, just assume that P_j is an OTM-program that decides \mathcal{L} within that time bound and pick $w \in \{0, 1\}^{**}$ such that $i(w) = j$ and $|w| > \gamma$.

To see that \mathcal{L} belongs to \mathcal{P}_β , notice that, for $|w| > \alpha$, we have $|w|^\beta \cdot 2 > |w|^\alpha |w|$. Thus, $|w|^\beta \cdot \omega$ steps suffice to simulate $P_{i(w)}$ for $|w|^\alpha |w|$ many steps on input w and flip the output, which decides \mathcal{L} . As α is by assumption the halting time of some OTM-program, the time bound $|w|^\alpha |w|$ is OTM-computable in the input w .

5 Conclusion and Further Work

In many respects, the complexity theory of OTMs resembles classical complexity theory; typical results from classical complexity theory also hold in the OTM-concept and can often be proved by adaptations of the classical arguments to

the infinitary framework. This suggests studying infinitary analogues of decision problems considered in classical complexity theory and to see what their OTM-complexity is. In [CLR], this was done for SAT, and it turned out that the usual proof of the Cook-Levin theorem could be adapted to yield an analogue for OTMs. However, things do not always go that smoothly: For example, while the independent set problem (i.e. determining whether a given graph G has a subset of n vertices, no two of which are connected) has a straightforward infinitary analogue (which is obtained by replacing n with an arbitrary ordinal), the classical reduction of SAT to the independent set problem no longer works in the infinitary context since infinitary sets can have infinitary subsets of the same size. More precisely, the classical reduction works by forming a graph based on a propositional formula ϕ given in disjunctive normal form by introducing a vertex for each occurrence of a literal and linking two vertices with an edge if and only if the corresponding literal occurrences belong to the same clause or contradict each other. Now an independent set the cardinality of which is the number of clauses induces a satisfying assignment for ϕ and vice versa. However, in the infinitary case, e.g. given a formula ϕ with ω many clauses, one can pick one literal from countable many clauses and make it true without thereby defining a satisfying assignment for ϕ , since we might have skipped some clauses.

This motivates the general, if somewhat vague question: What is it about decision problems in the classical sense that allows an infinitary generalization? Is there a general transfer principle?

On the other hand, we plan to explore the complexity classes of problems from infinitary combinatorics, such as the existence of infinite paths in a given tree.

Acknowledgements. We thank Philipp Schlicht for checking our proof of Theorem 2. We also thank our three anonymous referees for a considerable amount of constructive comments on the exposition.

References

- [B] Barwise, J.: Admissible sets and structures. *Stud. Log.* **37**, 297–299 (1975)
- [C14] Carl, M.: Randomness and degree theory for infinite time register machines 1. *Computability* **5**, 181–196 (2016)
- [CS] Carl, M., Schlicht, P.: Infinite computations with random oracles. *Notre Dame J. Form. Log.* **58**(2), 249–270 (2017)
- [CS2] Carl, M., Schlicht, P.: Randomness via infinite computation and effective descriptive set theory. *J. Symbol. Log.* (to appear)
- [DHS] Deolalikar, V., Hamkins, J., Schindler, R.: $P \neq NP \cap co - NP$ for infinite time turing machines. *J. Log. Comput.* **15**, 577–592 (2005)
- [CLR] Carl, M., Löwe, B., Rin, B.G.: Koepke machines and satisfiability for infinitary propositional languages. In: Kari, J., Manea, F., Petre, I. (eds.) *CiE 2017*. LNCS, vol. 10307, pp. 187–197. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58741-7_19
- [FG] Fortnow, L., Gasarch, B.: Two proofs of Ladner’s theorem. Online lecture notes. <http://oldblog.computationalcomplexity.org/media/ladner.pdf>

- [HL] Hamkins, J., Lewis, A.: Infinite time turing machines. *J. Symbol. Log.* **65**, 567–604 (2000)
- [HMS] Hamkins, J.D., Miller, R., Seabold, D., Warner, S.: Infinite time computable model theory. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) *New Computational Paradigms*, pp. 521–557. Springer, New York (2008). https://doi.org/10.1007/978-0-387-68546-5_20
- [Hro] Hromkovic, J.: *Theoretische Informatik: Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie*. Springer, Wiesbaden (2011)
- [HW] Hamkins, J., Welch, P.: $P^f \neq NP^f$ for almost all f . *Math. Log. Q.* **49**(5), 536 (2003)
- [ITRM] Koepke, P., Miller, R.: An enhanced theory of infinite time register machines. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) *CiE 2008. LNCS*, vol. 5028, pp. 306–315. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69407-6_34
- [KS] Koepke, P., Seyfferth, B.: Ordinal machines and admissible recursion theory. *Ann. Pure Appl. Log.* **160**, 310–318 (2009)
- [KT] Komjath, P., Totik, V.: *Problems and Theorems in Classical Set Theory*. Springer, New York (2006). <https://doi.org/10.1007/0-387-36219-3>
- [L] Löwe, B.: Space bounds for infinitary computation. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J. (eds.) *Logical Approaches to Computational Barriers. Lecture Notes in Computer Science*, vol. 3988, pp. 319–329. Springer, Berlin (2006). https://doi.org/10.1007/11780342_34
- [ORM] Koepke, P., Siders, R.: Computing the recursive truth predicate on ordinal register machines. In: Beckmann, A., et al. (eds.) *Logical Approaches to Computational Barriers. Computer Science Report Series*, vol. 7, pp. 160–169, Swansea (2006)
- [OTM] Koepke, P.: Turing computations on ordinals. *Bull. Symb. Log.* **11**, 377–397 (2005)
- [Schindler] Schindler, R.: $P \neq NP$ for infinite time turing machines. *Monatsh. Math.* **138**, 335–340 (2003)
- [W1] Welch, P.: Eventually infinite time turing machine degrees: infinite time decidable reals. *J. Symb. Log.* **65**(3), 1193–1203 (2000)
- [wITRM] Koepke, P.: Infinite time register machines. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J.V. (eds.) *CiE 2006. LNCS*, vol. 3988, pp. 257–266. Springer, Heidelberg (2006). https://doi.org/10.1007/11780342_27
- [Wi1] Winter, J.: *Space complexity in Infinite Time Turing Machines*. Master’s thesis, Universiteit van Amsterdam (2007)
- [Wi2] Winter, J.: Is $P = PSPACE$ for infinite time turing machines? In: Archibald, M., Brattka, V., Goranko, V., Löwe, B. (eds.) *ILC 2007. LNCS (LNAI)*, vol. 5489, pp. 126–137. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03092-5_10



Taming Koepke's Zoo

Merlin Carl¹(✉), Sabrina Ouazzani², and Philip Welch³

¹ Fachbereich Mathematik und Statistik,
Universität Konstanz, Universitätsstraße 10, 78464 Konstanz, Germany
merlin.carl@uni-konstanz.de

² LACL, Université Paris-Est Créteil, 61 avenue du Général de Gaulle,
94010 Créteil, France
sabrina.ouazzani@lacl.fr

³ School of Mathematics, University of Bristol, Clifton, Bristol BS8 1TW, UK
P.Welch@bristol.ac.uk

Abstract. For ordinals α and β , Koepke defined (α, β) -Turing machines as Turing machines with tape length α and working time bounded above by β . So far, their computational strength was determined for $\alpha = \beta$ exponentially closed, $\alpha = \beta = \text{On}$ and $(\alpha, \beta) = (\omega, \text{On})$. In this paper, we determine the set of (α, β) -writable subsets of α when α is multiplicatively closed and $\beta > \alpha$ is admissible. This answers some open questions by Koepke in [5].

1 Introduction

In [2], Hamkins and Lewis introduced Turing machines which run along an ordinal time axis, while ‘keeping’ the tape length ω . Soon afterwards, the prospects of expanding the tape length as well were considered, along with the idea of restricting the working time to a certain ordinal. In this way, one obtains Koepke’s (α, β) -Turing machines, which have a tape of length α while their working time is bounded by β . We are thus facing a family of $\text{On} \times \text{On}$ many models of computation, which we label ‘Koepke’s Zoo’.

We briefly describe the mechanics of (α, β) -Turing machines; a detailed account can be found in [8].

An (α, β) -Turing machine ((α, β) -TM, (α, β) -machine) has an input tape, an output tape and a scratch tape, each of length $\alpha \in \text{On}$, each cell of which can contain 0 or 1. When α is closed under the Cantor pairing function, which will always be the case in this paper, one may grant the machine finitely many scratch tapes instead of a single one without changing any of the results in this paper; we will make silent use of this where convenient. Programs for (α, β) -TMs are regular Turing programs. We imagine the inner states occurring in a program indexed uniquely with natural numbers. The computation of such a program on an (α, β) -TM works by carrying out the usual Turing operation at successor ordinals. When the read-write-head is commanded to move to the left while currently occupying a cell indexed by a limit ordinal, it is reset to position 0. At a limit time $\delta < \beta$, the inner state is determined as the inferior limit of

the sequence of earlier inner states (which, as we recall, are labelled with natural numbers). For $\iota < \alpha$, the content of the ι th tape cell at time δ is determined as the inferior limit of the sequence of earlier contents of that cell. The head position at time δ is also the inferior limit of the sequence of earlier head positions, if this is $< \alpha$. Otherwise, the head is reset to cell 0. A computation is considered to be halting if the halting state is reached in $< \beta$ many steps. Otherwise, it is considered as non-halting. We will consider computations relative to parameters. For a machine with an α -tape, a parameter is a finite sequence $\mathbf{p} \subseteq \alpha$ which is given to the machine by marking the cells with indices in \mathbf{p} with 1.

The following cases have so far been considered:

(ω, ω) -machines are of course regular Turing machines. (ω, On) -machines were invented by Kidder and introduced by Hamkins and Lewis in [2] as ‘Infinite Time Turing Machines’ and were extensively studied e.g. in [1, 2, 11].

(α, α) -machines for admissible α were studied by Koepke and Seyfferth in [8] and turned out to compute exactly the $\Delta_1(L_\alpha)$ -subsets of α . (On, On) -machines are known as ‘Ordinal Turing Machines’, and it was shown by Koepke in [4] that, if ordinal parameters are admitted, then they compute the constructible sets of ordinals. Rin [9] considered parameter-free (α, On) -machines for arbitrary α and showed that they exhibit a somewhat extravagant behaviour; for example, there are $\alpha < \alpha'$ such that there are (α', On) -incomputable subsets of α that are nevertheless (α, On) -computable. Also, note that (α, β) -machines are equivalent to (β, β) -machines for $\alpha \geq \beta$, as no cells with index $\geq \beta$ can be reached in fewer than β many steps.

In this paper, we determine the computational strength for many of the remaining machine models; more specifically, we show that, if α is multiplicatively closed (i.e. if $\gamma, \delta < \alpha$, then $\gamma \cdot \delta < \alpha$) and $\beta > \alpha$ is admissible, then the subsets of α that are computable by an (α, β) -machine with parameters (i.e. finitely many ordinals $< \alpha$) are exactly those contained in L_β . This answers some questions left open by Koepke in Sect. 3 of [5]. The analysis can be extended to broader classes of ‘reasonably closed’ α and β , which, however, leads to some extra technical complications, which we decided to avoid mainly for the sake of readability.

2 The Computational Strength of (α, β) -Turing Machines

A crucial concept in the analysis of (α, β) -machines is that of the ‘clockability’ of an ordinal. We say that γ is α -clockable if and only if there are an α -Turing machine program P and a parameter \mathbf{p} such that $P(\mathbf{p})$ runs for exactly $\gamma + 1$ many steps.

A subset $A \subseteq \alpha$ is (α, β) -writable if and only if there are an α -Turing machine program P and a parameter \mathbf{p} such that $P(\mathbf{p})$ halts in $< \beta$ many steps with (the characteristic function of) A on the output tape. If this holds for some $\beta \in \text{On}$, then A is α -writable.

An ordinal δ is α -writable if and only if there is an α -writable $A \subseteq \alpha$ which codes a well-ordering of length δ . Here, coding of well-orderings via subsets of α

works as follows: Let p denote Cantor’s ordinal pairing function. Then $x \subseteq \alpha$ is a code for the ordinal δ if and only if there is a bijection $f : \alpha \rightarrow \delta$ such that $x = \{p(\iota, \iota') : f(\iota) \in f(\iota')\}$. Note that multiplicative closure of α implies closure under p .

In the following, when we say that a program P halts, diverges, has a certain output etc. without specifying an input, we mean that P is run on the empty input, i.e. tapes that initially contain 0s everywhere.

It is not hard to see that, for any α , the α -writable ordinals form a downwards closed set: If $c \subseteq \alpha$ codes an ordinal η and $\beta < \eta$, then there is $\iota < \alpha$ such that restricting c to elements smaller than ι in the sense of c yields a code for β ; this restriction is easily computable in the parameter ι . However, the same does not hold for the clockable ordinals: It is shown in [2] that the set of ω -clockable ordinals has ‘gaps’, i.e. there are ω -clockable ordinals β, η with $\beta + \omega < \eta$ such that no $\xi \in [\beta, \eta)$ is ω -clockable. For machines with tape length α , we call such intervals ‘ α -clockable gaps’. If in this situation (β', η) contains an α -clockable ordinal for every $\beta' < \beta$, then β is called the ‘start of the gap’. The ordinal δ ‘belongs to the gap $[\beta, \eta)$ ’ if and only if $\beta < \delta < \eta$.

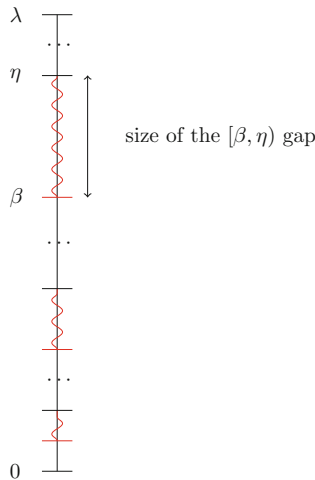


Fig. 1. A $[\beta, \eta)$ gap in the ω -clockable ordinals.

Let us observe at this point that α^ι is α -clockable for any $\iota \in \omega + 1$, which will become relevant later on.

Lemma 1. *If α is additively closed, then α^ι is α -clockable for every $\iota \in \omega + 1$.*

Proof. We show inductively that α^n is α -clockable for any $n \in \omega$. To this end, we show how to obtain, recursively in $n \in \omega$, an α -machine program Q_n that clocks α^n . It then follows that there is a program Q that runs all Q_i one after another and will thus clock α^ω .

The Q_1 case is dealt with by Rin's 'LoopAround' algorithm, see [9]: Clearly, α is α -clockable by a program that simply moves the head to the right until it is back at the starting position (which is e.g. the unique cell marked with 1).

Now suppose that Q_n is given; then Q_{n+1} works as follows: Until the end of the tape is reached, place $0 \underbrace{11 \cdots 1}_{(n+1) \times}$ to the right of the supremum ρ of all blocks

of $(n + 1)$ successive 1s on the working tape. Then run Q_n on the rest of the tape (when the head is reset to position 0 during the run of Q_n , it is moved to position ρ before Q_n is continued).

By multiplicative closure of α and the inductive assumption, this will take $\alpha^{n+1} + 1$ many steps and thus clock α^{n+1} .

Note that Q_i can easily be obtained recursively from i for $i \in \omega$. We can thus run all Q_i in succession, which will clock α^ω .

We fix some notation. If α is an ordinal, then α^+ denotes the smallest admissible ordinal $> \alpha$. If $\alpha, \beta \in \text{On}$ and β belongs to an α -clockable gap, then $\text{gap}_\alpha(\beta)$ denotes the ordinal that starts this gap; otherwise, $\text{gap}_\alpha(\beta) = \beta$. When $\alpha = \omega$, the subscript is dropped. We also fix natural enumerations $(P_i : i \in \omega)$ and $(\phi_i : i \in \omega)$ of the Turing programs and the \in -formulas, respectively.

2.1 Tape Length ω

We start by determining the computational strength of (α, β) -machines in the case that $\alpha = \omega$ and $\beta > \omega$ is admissible. This already reveals the main structure of the argument.

We will need the following facts about (ω, On) -machines, which are better known as ITTMs:

Theorem 1 [Welch, see [11]]. *There are ordinals λ, ζ, Σ such that $x \subseteq \omega$ is ITTM-writable if and only if $x \in L_\lambda$. Moreover, every ITTM will either halt in $< \lambda$ many steps or cycle from time Σ on, repeating its sequence of configurations between ζ and Σ over and again. Finally, (λ, ζ, Σ) is characterized as the lexically minimal triple such that $L_\lambda \prec_{\Sigma_1} L_\zeta \prec_{\Sigma_2} L_\Sigma$.*

Theorem 2. *No admissible ordinal is ω -clockable. [Hamkins and Lewis, see [2]] Conversely, if β starts an ω -clockable gap, then β is admissible. [Welch, see [11]]*

Theorem 3 [See [11], Lemma 48 and [1], Remark after Lemma 2]. *Suppose that α is ω -clockable. Then a code for α is ω -writable in exactly α many steps and a code c for L_α is ω -writable in $< \alpha^+$ many steps.*

Using these ingredients, we prove:

Theorem 4. *Let $\beta > \omega$ be admissible. Then $x \subseteq \omega$ is (ω, β) -writable if and only if one of the following holds:*

- (i) β does not belong to an ω -clockable gap and $x \in L_\beta$.

(ii) β does belong to an ω -clockable gap started by some β' and $x \in L_{\beta'}$. Thus, $x \subseteq \omega$ is (ω, β) -writable if and only if $x \in L_{\text{gap}_\omega(\beta)}$.

Proof. First, note that the definition of the computation of an ω -tape Turing machine is absolute between V and the constructible hierarchy; also, by an easy induction, the computation up to time γ will be contained in $L_{\gamma+\omega}$. Thus, if $x \subseteq \omega$ is (ω, β) -writable, then it is written at some time $\gamma < \beta$; the whole computation, and hence x , will thus be contained in $L_{\gamma+\omega} \subseteq L_\beta$. (This requires only that $\gamma + \omega < \beta$ for all $\gamma < \beta$, i.e. that β is a multiple of ω^2 .)

If β belongs to an ω -clockable gap started by β' , then no ITTM-program will halt between times β' and β ; the interval $[\beta', \beta)$ is ‘empty time’ as far as ITTM-Turing machine computations are concerned. Thus $x \subseteq \omega$ is (ω, β) -writable if and only if it is (ω, β') writable and (ii) reduces to (i).

Now suppose that β does not belong to an ω -clockable gap. It remains to show that every $x \in L_\beta$ is (ω, β) -writable. As β does not belong to a gap, there must be cofinally many ω -clockable ordinals below β . In particular then, there is an ω -clockable ordinal $\gamma < \beta$ such that $x \in L_\gamma$. By Theorem 3, the ω -clockability of γ implies that a code c for L_γ is writable in $<\gamma^+ \leq \beta$ many steps. In the code for L_γ , x is coded by some natural number i which can be given to the program as a parameter.¹ Now, given i and c , the code x can easily be computed in time $<\beta$, thus x is (ω, β) -writable.

2.2 The General Case

We now turn to the general case. In this section, α is always multiplicatively closed (and thus a fortiori closed under Cantor pairing) and β is always admissible and strictly greater than α . In contrast to the last section, we work in Jensen’s J -hierarchy (see [3]) rather than in the Gödel-hierarchy. In addition to allowing more direct generalizations of the relevant results from [1] to the α -tape case, this has a number of technical advantages, such as the existence of uniformly and parameter-freely Σ_1 -definable Σ_1 -Skolem functions, i.e. maps $f : \omega \times J_\gamma^{<\omega} \rightarrow J_\gamma$ that map the pair (i, \mathbf{p}) to some (in fact, the $<_L$ -minimal) witness for the i th Σ_1 -formula in the parameter \mathbf{p} (see e.g. Theorem 1.15 of [10]). Also recall from [3] that, for $\omega\alpha = \alpha$, we have $L_\alpha = J_\alpha$; thus, for reasonably closed ordinals such as α and β , but also the ordinals $\lambda(\alpha)$, $\zeta(\alpha)$ and $\Sigma(\alpha)$ defined below, this does not make a difference; in these cases, we will prefer to state our results in terms of the better-known L -hierarchy.

We start by showing that an (α, β) -machine can produce a code for L_α . To this end, we introduce two-dimensional ordinal machines as an auxiliary concept.

Definition 1. *An $\alpha \times \alpha$ -Turing machine has a two-dimensional ‘tape’ of dimension $\alpha \times \alpha$. The head is initially located at position $(0, 0)$. The head can be moved*

¹ This, of course, is unnecessary, as every natural number is ITTM-writable. We proceed in this way for the sake of analogy with the general case, in which it is no longer the case that there is a parameter-freely writable code for each ordinal below the tape length.

to the right, to the left, up and down. The programs are like regular Turing programs, with the modification that ‘up’ and ‘down’ are added to the legitimate head movements. Here, going ‘up’ from position (δ, γ) results in position $(\delta, \gamma + 1)$, while going ‘down’ from $(\delta, \gamma + 1)$ leads to (δ, γ) . If δ is a limit ordinal, moving the head to the left (resp. downwards) from position (δ, γ) (resp. (γ, δ)) results in the head location $(0, \gamma)$ (resp. $(\gamma, 0)$). At limit times, the head position is determined by coordinate-wise inferior limits (or resets to 0, if the limit is α). A set $A \subseteq \alpha$ is $(\alpha \times \alpha, \beta)$ -writable if and only if there is an $\alpha \times \alpha$ -program P that, when run on the empty ‘tape’, halts in $< \beta$ many steps with the characteristic function of A on the portion $\{(0, \iota) : \iota < \alpha\}$ of its ‘tape’.

Lemma 2. *If $A \subseteq \alpha$ is $(\alpha \times \alpha, \beta)$ -writable, then it is (α, β) -writable.*

Proof. This is proved by simulating $\alpha \times \alpha$ -machines by α -machines. The simulation works by splitting the α -tape into α many disjoint portions via Cantor pairing, each representing one ‘row’ of the two-dimensional tape. The details are omitted for the sake of brevity.

Lemma 3. *There is an (α, β) -writable code $x \subseteq \alpha$ for $L_\alpha = J_\alpha$.*

Proof. This can be seen by using an $\alpha \times \alpha$ -machine to build up the L -hierarchy iteratively, writing a code for L_γ in the $(\gamma + 1)$ th ‘row’ of the tape for $\gamma < \alpha$ and finally one for L_α on the 0th row. It can be checked that this works in $< \alpha^+$ many steps, so that the claim follows from the last lemma. The details are again omitted.

Definition 2. *Let $\alpha, \gamma \in On$. Then $\Sigma_2\text{-Th}(J_\gamma, \alpha)$ denotes the set of pairs $([\phi], \mathbf{p})$, where $[\phi]$ is (the Gödel number of) a Σ_2 -formula and $\mathbf{p} \subseteq \alpha$ is a finite sequence such that $J_\gamma \models \phi(\mathbf{p})$.*

Lemma 4. *Let $c \subseteq \alpha$ be an (α, β) -writable code for a structure (X, E) , where E is a binary relation on X . Then the elementary diagram T of (X, E) is (α, β) -writable. In particular, this holds for $\Sigma_2\text{-Th}(J_\gamma, \alpha)$ whenever J_γ has an (α, β) -writable code.*

Proof. This is an easy adaption of Lemma 4 of [8].

We now generalize Theorem 2.

Lemma 5. *For any multiplicatively closed $\alpha \in On$, there are ordinals $\lambda(\alpha)$, $\zeta(\alpha)$, $\Sigma(\alpha)$ such that a subset of α is writable by an (α, On) -Turing machine with finitely many ordinal parameters $< \alpha$ if and only if it is contained in $L_{\lambda(\alpha)}$, it is eventually writable by an (α, On) -Turing machine (i.e. every tape cell eventually stabilizes at the correct value) if and only if it is contained in $L_{\zeta(\alpha)}$ and it is accidentally writable by an (α, On) -tape Turing machine (i.e. it appears at some point on the output tape, whether it stays there or not) if and only if it is contained in $L_{\Sigma(\alpha)}$. Moreover, $\lambda(\alpha)$ is the supremum of the halting times of (α, On) -Turing machines with parameters. Finally, $(\lambda(\alpha), \zeta(\alpha), \Sigma(\alpha))$ is the lexically minimal triple of ordinals such that $\lambda(\alpha) > \alpha$ and $L_{\lambda(\alpha)} \prec_{\Sigma_1} L_{\zeta(\alpha)} \prec_{\Sigma_2} L_{\Sigma(\alpha)}$.*

Proof. The proof is analogous to that of the λ - ζ - Σ -theorem, see [11].

We use the following version of a subclaim from the proof of Lemma 1 from [1] (see also [12]). Here, $h_1^{J_\beta}$ is the canonical Σ_1 -Skolem function for J_β (see [3]); the superscript β is dropped when it is clear from the context.

Lemma 6. *Let $\alpha \in On$, and let $\delta > \alpha$ be a limit ordinal such that $\delta < \Sigma(\alpha)$. Furthermore, let $\phi(\mathbf{p}) \equiv \exists x \forall y \psi(x, y, \mathbf{p})$ be a Σ_2 -formula, where ψ is Δ_0 and $\mathbf{p} \subseteq \alpha$ is a finite sequence. Then $J_\delta \models \phi(\mathbf{p})$ if and only if there is an $n \in \omega$ such that, for all sufficiently large $\beta < \delta$, we have that the following holds in J_β : There is γ with $\gamma = 0$ or $J_\gamma \prec_{\Sigma_1} J_\beta$ such that $J_\gamma \models \phi(\mathbf{p})$ or $h_1(n, \gamma)$ exists and $J_\beta \models \forall y \psi(h_1(n, \gamma), \mathbf{p})$.*

Proof. The proof is analogous to that given in [1]; see also [12], Sect. 4.1.

Corollary 1. *There is an α -machine program P_{limit} such that the following holds:*

Let $\delta > \alpha$ be a limit ordinal such that $\delta < \Sigma(\alpha)$, and let $\widehat{T}_\delta := \{(\phi, \mathbf{p}) : \mathbf{p} \subseteq \alpha \wedge \phi \in \Sigma_2 \wedge \exists \beta < \delta \forall \gamma < \delta (\beta < \gamma \rightarrow J_\gamma \models \phi(\mathbf{p}))\}$.

Then, given a subset of α coding \widehat{T}_δ , P_{limit} writes a code for $\Sigma_2\text{-Th}(J_\delta, \alpha)$ in α many steps.

Proof. For each Σ_2 -formula ϕ and each finite sequence \mathbf{p} of ordinals in α , the question whether $L_\delta \models \phi(\mathbf{p})$ reduces, via Lemma 6, to the question whether there is $n \in \omega$ such that a certain Σ_2 -formula ϕ' with parameter \mathbf{p} and n holds in J_β for all sufficiently large $\beta < \delta$. The appropriate ϕ' given by Lemma 6 is recursive in ϕ and thus computable from ϕ in finitely many steps. Looking up (ϕ', \mathbf{p}) in \widehat{T}_δ is done by running to and reading out the appropriate cell of the input tape, which takes $< \alpha$ many steps. This is done α many times. Searching for an appropriate $n \in \omega$ requires iterating this ω many times. By multiplicative closure of α , this takes α many steps in total.

We now work towards an α -tape version of the ‘theory machine’ in [1]. Intuitively, the theory machine is a program that writes codes of the Σ_2 -theories of the levels of the J -hierarchy in a controllable amount of time.

Lemma 7. *There are α -TM programs P_{lft} (‘level from theory’) and P_{succ} such that the following holds for every $\rho < \Sigma(\alpha)$:*

- (i) *Given $T := \Sigma_2\text{-Th}(J_\rho, \alpha)$ as the input, P_{lft} halts in $< \alpha^2 \cdot \omega$ many steps and outputs a code for J_ρ .*
- (ii) *Given $T := \Sigma_2\text{-Th}(J_\rho, \alpha)$ as the input, P_{succ} halts in $< \alpha^\omega \cdot 2$ many steps and outputs $\Sigma_2\text{-Th}(J_{\rho+1}, \alpha)$.*

Proof. (i) We use the following Fact generalising Theorem 3 of [13]. For this we let $\beta_0(\alpha)$ be the least $\beta > \alpha$ with $J_\beta \models ZF^-$.

Fact (Uniform Σ_n -Skolem Functions). For every $n < \omega$ there is a single Σ_n -definition of a partial function h_n , which defines a Σ_n -Skolem function over any

$\langle J_\gamma, \in \rangle$ with $\gamma < \beta_0(\alpha)$, in the sense that for any $X \subseteq J_\gamma$ then $h_n^{J_\gamma} \omega \times (X \cup \alpha)$ is the least Σ_n elementary substructure of J_γ .

By our choice of ρ we have that $\rho < \beta_0(\alpha)$. Taking $n = 2$ we see there is a partial map $h = h_2^{J_\gamma}$ map of α onto J_γ . (Since $J_\gamma \models \forall x(|x| \leq \alpha)$ and any $\tau \in h \omega \times \alpha$ is the range of an onto map $f_\tau \in h \omega \times \alpha$ from α , we shall have that $h \omega \times \alpha$ is transitive and so is some $J_\delta \prec_{\Sigma_2} J_\gamma$ with $\delta \leq \gamma$. By our minimal choice of ρ we must have $\delta = \gamma$.) Now suppose we are given T . We have shown that h is thus a partial function from α onto J_γ . Furthermore statements such as “ $h(i, \xi) = \setminus \in h(i', \xi')$ ” are included in T . Thus given the set of (i, ξ) so that $h(i, \xi)$ is defined, define the equivalence relation \sim and equality relation E by: $h(i, \xi) \sim \setminus \in h(i', \xi')$ respectively, then this yields a model isomorphic to (J_γ, \in) and we are done.

(ii) Is an easy consequence of (i): Use P_{flt} to compute a code $c \subseteq \alpha$ for J_γ from T . Then compute a code $c' \subseteq \alpha$ for $J_{\gamma+1}$ from c by successively computing the intermediate S -levels and using Lemma 4 to evaluate the \in -relations in these levels. Finally, read off $\Sigma_2\text{-Th}(J_{\gamma+1}, \alpha)$ from c' .

The following is an α -TM version of Theorem 4 of [1], there called as the ‘theory machine’:

Lemma 8. *Let $\beta > \alpha$ be admissible. There are an α -TM program P_{theory} and an α -clockable ordinal $\gamma < \beta$ with the following property: For any $\tau < \Sigma(\alpha)$, P_{theory} contains $\Sigma_2\text{-Th}(J_\tau, \alpha)$ on its output tape at time $\gamma + \alpha^\omega \cdot 2 \cdot \tau$.*

Proof. We reserve the first two tape cells as ‘flags’. Initially, they will be set to the contents 0 and 1. By Lemma 3, a code for J_α is (α, β) -writable; let $\gamma < \beta$ be the halting time of the respective program, then γ is clearly α -clockable. Now, when $\Sigma_2\text{-Th}(J_\delta, \alpha)$ is given on the output tape at time τ , we use P_{succ} to write $\Sigma_2\text{-Th}(J_{\delta+1}, \alpha)$ on the output tape; in parallel, we clock $\alpha^\omega \cdot 2$ and continue with computing $\Sigma_2\text{-Th}(J_{\delta+2}, \alpha)$ from $\Sigma_2\text{-Th}(J_{\delta+1}, \alpha)$ only after the clocking program has halted. Whenever a new $\Sigma_2\text{-Th}(J_\delta, \alpha)$ has been written, the contents of the flag cells are flipped. The flag cells will thus have equal content if and only if the order type of the sequences of theories so far computed is a limit ordinal δ , in which case we use P_{limit} to compute $\Sigma_2\text{-Th}(J_\delta, \alpha)$ from the ‘limit theory’ currently on the output tape. Then, we use the clocking program to wait for $\alpha^\omega 2$ many steps to ensure that the theory is not overwritten ‘too early’.

Thus, we have at least the following α -version of ‘quick writing’:

Lemma 9. *If τ is a halting time of an α -program P with parameters \mathbf{p} , then a code for some J_γ with $\gamma \geq \tau$ is α -writable in time $< \tau^+$ in the parameter \mathbf{p} .*

Proof. We run the program P_{theory} . The statement that $P(\mathbf{p})$ has halted is Σ_1 in the parameter \mathbf{p} . Thus, given the parameter \mathbf{p} , it can be read off from $\Sigma_2\text{-Th}(J_\gamma, \alpha)$ whether $P(\mathbf{p})$ has halted at time γ . As soon as this happens, P_{flt} is applied to the content of the output tape (which will be $\Sigma_2\text{-Th}(J_\gamma, \alpha)$ for some γ with $\omega\gamma > \tau$) to obtain a code for some J -level with index $\geq \tau$. By the estimates given on the running time of P_{theory} and P_{flt} , this takes place in time well before τ^+ .

Theorem 5. *For any multiplicatively closed $\alpha \geq \omega$ and any admissible $\beta > \alpha$, if β does not belong to an α -clockable gap, then the subsets of α which are (α, β) -writable with finitely many ordinal parameters $< \alpha$ are exactly those contained in $L_\beta = J_\beta$. If β belongs to a gap started by γ , then it is the set of those in $L_\gamma = J_\gamma$.*

Thus, the (α, β) -writable subsets of α are exactly those in $L_{\text{gap}_\alpha(\beta)}$.

Proof. The case that β belongs to a gap is reduced to the case where it does not as in the proof of Theorem 4. So let us now assume that β does not belong to a gap, so that the α -clockable ordinals are cofinal in β .

That the (α, β) -writable subsets of α (with parameters) are contained in L_β is shown as in the proof of Theorem 4.

Thus, let $A \subseteq \alpha$ be contained in J_β . Pick $\gamma < \beta$ such that $A \in J_\gamma$. By Lemma 9, a code c for some J_δ with $\delta \geq \gamma$ can be written in time $< \beta$. From c , one can write A , as it is coded in the code for L_γ by some ordinal $\iota < \alpha$, which in turn we can give to the machine as a parameter. Thus A is (α, β) -writable.

3 Conclusion and Further Work

By the results in this paper, the missing fields in the table given on p. 8 of [5] can be filled out (new results are written in boldface and marked with an asterisk):

	Tape length ω	Tape length α multiplicatively closed	Tape length On
Time ω	$\Delta_1(L_\omega) = \Delta_1^0$ (Folklore)	$\Delta_1(L_\omega) = \Delta_1^0$ (Folklore)	$\Delta_1(L_\omega) = \Delta_1^0$ (Folklore)
Time β admissible	$*\mathfrak{P}(\omega) \cap L_{\text{gap}_\omega(\beta)}$	$\Delta_1(L_{\min(\alpha, \beta)})$ if $\beta \leq \alpha$ and α is exponentially closed (Koepke, Seyffarth) $*L_{\text{gap}_\alpha(\beta)} \cap \mathfrak{P}(\alpha)$ if $\beta > \alpha$	$\Delta_1(L_\beta)$ (Koepke, Seyffarth)
Time On	$\mathfrak{P}(\omega) \cap L_\lambda$ (Hamkins, Lewis, Welch)	$*\mathfrak{P}(\alpha) \cap L_{\lambda(\alpha)}$	$\mathfrak{P}(\text{Ord}) \cap L$ (Koepke)

A natural next question concerns the notions of eventual and accidental writability associated with tape models of infinitary computability (see e.g. [2]). What are the sets of eventually and accidentally writable subsets of α for an (α, β) -TM?

Also, the arguments in this paper can probably be applied to work for weaker closure conditions on α than multiplicative closure: With some extra effort, the closure condition can probably be removed altogether. It might also well be that it is sufficient to require that the time bound β is exponentially closed rather than admissible.

In contrast, a question left wide open by the above is the strength of (α, β) -TMs without ordinal parameters, which are basically a time-bounded version of the machines considered in [9].

Finally, we will in further work consider the (α, β) -variants of the machine model based on register machines, such as Koepke’s Infinite Time Register

Machines (ITRMs, [6]) and Ordinal Register Machines (ORMs, [7]). We conjecture that, for appropriately closed α and admissible $\beta > \alpha$, the (α, β) -register machine computable subsets of α will be those in $L_{\min\{\alpha+\omega, \beta\}}$, where $\alpha^{+\omega}$ denotes the smallest limit of admissible ordinals $> \alpha$ (the arguments in [6] can easily be adapted to show that this is in fact an upper bound).

References

1. Friedman, S., Welch, P.: Two observations concerning infinite time Turing machines. In: Dimitriou, I. (ed.) BIWOC Report, pp. 44–47. Hausdorff Centre for Mathematics, Bonn (2007)
2. Hamkins, J.D., Lewis, A.: Infinite time turing machines. *J. Symb. Log.* **65**(2), 567–604 (2000)
3. Jensen, R.: The fine structure of the constructible hierarchy. *Ann. Math. Log.* **4**, 229–308 (1972)
4. Koepke, P.: Computing a model of set theory. In: Cooper, S.B., Löwe, B., Torenvliet, L. (eds.) CiE 2005. LNCS, vol. 3526, pp. 223–232. Springer, Heidelberg (2005). https://doi.org/10.1007/11494645_28
5. Koepke, P.: Ordinal computability. In: Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) CiE 2009. LNCS, vol. 5635, pp. 280–289. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03073-4_29
6. Koepke, P., Miller, R.: An enhanced theory of infinite time register machines. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 306–315. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69407-6_34
7. Koepke, P., Syders, R.: Computing the recursive truth predicate on ordinal register machines. In: Beckmann, A., et al. (eds.) Logical Approaches to Computational Barriers. Computer Science Report Series, Swansea, vol. 7, pp. 160–169 (2006)
8. Koepke, P., Seyfferth, B.: Ordinal machines and admissible recursion theory. *Ann. Pure Appl. Log.* **160**, 310–318 (2009)
9. Rin, B.: The computational strengths of α -tape infinite time turing machines. *Ann. Pure Appl. Log.* **165**(9), 1501–1511 (2014)
10. Schindler, R., Zeman, M.: Fine structure. In: Foreman, M., Kanamori, A. (eds.) Handbook of Set Theory, pp. 605–656. Springer, Dordrecht (2010). https://doi.org/10.1007/978-1-4020-5764-9_10
11. Welch, P.: Characteristics of discrete transfinite time turing machine models: halting times, stabilization times, and normal form theorems. *Theoret. Comput. Sci.* **410**, 426–442 (2009)
12. Welch, P.: Some observations on truth hierarchies. *Rev. Symb. Log.* **7**(1), 1–30 (2014)
13. Welch, P.: Weak systems of determinacy and arithmetical quasi-inductive definitions. *J. Symb. Log.* **76**(2), 418–436 (2011)



Online Computability and Differentiation in the Cantor Space

Douglas Cenzer¹(✉) and Diego A. Rojas²

¹ Department of Mathematics, University of Florida, P.O. Box 118105, Gainesville,
FL 32611-8105, USA

cenzer@ufl.edu

² Department of Mathematics, Iowa State University, 411 Morrill Road,
396 Carver Hall, Ames, IA 50011-2104, USA

darojas@iastate.edu

Abstract. This paper investigates a notion of differentiation for functions on the Cantor space. We study the existence and complexity of this derivative, particularly for online and computable online functions. It is shown that a random online function has no derivative at any computable point. It is shown that if a computable online function F has derivative $m > 0$ at a weakly 1-random point, then F has derivative m on a set of positive measure. We also explore the family of online functions on the Cantor space which represent real-valued functions.

1 Introduction

The interaction between computable analysis and algorithmic randomness has been very productive in recent years. Following the introduction of the concept of algorithmically random sequences by Martin-Löf [10], notions have been developed for random sets and functions. Barmpalias et al. [1] defined the random closed sets on the Cantor space $2^{\mathbb{N}}$, and later extended their work to random continuous functions [2] on $2^{\mathbb{N}}$. The connection between random closed sets and capacities has been developed in [4, 5]. The latter paper examined the notions of online functions on $2^{\mathbb{N}}$, and in particular of computable and random online functions. Brattka et al. [3] showed in 2016 that a real number z is computably random if and only if each nondecreasing computable function from $[0, 1]$ to \mathbb{R} is differentiable at z .

The computability and complexity of differentiability has been studied going back to Myhill [11], who constructed in 1971 a computable real function with a derivative which is continuous but not computable. Cenzer and Remmel [6] gave an enumeration F_0, F_1, \dots of the (partial) computable real functions and showed that the index set of the computably differentiable functions is Σ_3^0 complete, the index set of the continuously differentiable functions is Π_3^0 complete, and the index set of the everywhere differentiable functions is Π_1^1 complete. Pour-El and Richards [12] defined a computable ordinary differential equation which possess no computable solution. Heinonen wrote a survey [8] on *nonsmooth calculus* concerned with differentiation and integration on general “nonsmooth” spaces.

Cenzer and Porter discussed integrals of random functions on $2^{\mathbb{N}}$ in [5]. In the present paper, we develop a notion of differentiation for online continuous functions on $2^{\mathbb{N}}$ and make a connection with algorithmic randomness. In particular, we show that a random online function is not differentiable at any computable point, and we show that if an online computable function F has derivative $F'(A) = m > 0$ at any weakly one-random point, then $F'(X) = m$ on a set of positive measure.

We outline the paper in the following way:

Section 2 introduces the necessary definitions and background that will be used throughout the paper. A function F on $2^{\mathbb{N}}$ is said to be *online* continuous if the first n values of the output $Y = F(X)$ are determined by the first n values of the input X ; F is online computable if there is a program to compute $Y(n)$ from $X(0), \dots, X(n)$. In addition, we define reducibility notions in the arithmetical hierarchy and the effective Wadge hierarchy of sets that will be used in the discussion on computability and complexity of derivatives in $2^{\mathbb{N}}$.

Section 3 gives some basic results about online functions and introduces the notion of the derivative of a function on $2^{\mathbb{N}}$ which we are exploring in the paper. Given the standard distance $d(A, B) = 2^{-n-1}$ where n is the least such that $A(n) \neq B(n)$, we let

$$F'(A) = \lim_{X \rightarrow A} \frac{d(F(X), F(A))}{d(X, A)},$$

if this exists. We also look at the connections between functions on $2^{\mathbb{N}}$ and functions on \mathbb{R} . We show that certain classes of real-valued functions can be represented by online functions. If an online function F on $2^{\mathbb{N}}$ represents a real function $\Phi : [0, 1] \rightarrow [0, 1]$, then Φ satisfies a Lipschitz condition and is therefore differentiable almost everywhere.

Section 4 contains a discussion on the complexity of sets in relation to the derivatives of computable online functions on $2^{\mathbb{N}}$. We also give insight as to the computability of the derivative of a computable online function. We show in particular that for any online computable function F , $\{X : F'(X) = 0\}$ is a Π_3^0 class and that for any computable $A \in 2^{\mathbb{N}}$, the set of (representations for) online continuous functions F such that $F'(A) = 0$ is lightface Π_3^0 complete. We give examples to show that the derivative of an online computable function F need not be continuous even when F' is defined everywhere.

Section 5 considers the existence of the derivative for random functions and for random inputs. We prove in particular that a random online function is not differentiable at any computable point.

In Sect. 6, we consider further the representation of real functions by online functions on $2^{\mathbb{N}}$. We show that a linear function $\Phi(r) = mr + b$ on $[0, 1]$ with $m \neq 0$ may be represented by an online function if there exists n such that $m = \pm 2^{-n}$ and there exists k such that $b = k/2^{n+1}$. For the converse direction, we show that if $\Phi = mr + b$ is represented by an online function, then $m = \pm 2^{-n}$ for some n and $b = k/2^{2n}$ for some k . We show that if Φ is represented by an online function, then $|\Phi(r) - \Phi(s)| \leq |r - s|$ for all reals r, s so that Φ is differentiable almost everywhere and, furthermore, $|\Phi'(r)| \leq 1$ whenever it exists.

Many proofs are omitted due to page limitations. Details will be given in a planned full version of the paper.

2 Background

Some definitions are needed for the discussion of closed and open sets and continuous functions on the space $2^{\mathbb{N}}$. The space $2^{\mathbb{N}}$ may be viewed as a metric space with the distance $d(X, Y)$ between two distinct elements given by 2^{-n-1} , where n is the least such that $X(n) \neq Y(n)$. Then for any X and any n , the basic open set $\{Y : d(X, Y) < 2^{-n}\}$ is the interval $[[X \upharpoonright n] = \{Y : Y \upharpoonright n = X \upharpoonright n\}$. An alternate metric may be given by letting $d_2(X, Y) = \sum \{2^{-n-1} : X(n) \neq Y(n)\}$. Note that $d(X, Y) \leq d_2(X, Y) \leq 2d(X, Y)$, so that d and d_2 induce the same topology.

A continuous function F on $2^{\mathbb{N}}$ has a representation $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that

- (a) For all strings $\sigma \sqsubset \tau$, $f(\sigma) \subseteq f(\tau)$;
- (b) $(\forall n)(\exists m)(\forall \sigma \in \{0, 1\}^m) |f(\sigma)| \geq n$.

Then for $Y = F(X)$, we have $Y = \bigcup_m f(X \upharpoonright m)$.

A function F is said to be *online continuous* if it has a representation g with $|g(\sigma)| = |\sigma|$ for all strings σ ; F is said to be *online computable* if the function g is computable. Alternatively, we can represent F by the function $f : \{0, 1\}^* \rightarrow \{0, 1\}$, where $f(\sigma) = g(\sigma)(|\sigma| - 1)$. Then $g(\sigma \smallfrown 0) = g(\sigma) \smallfrown f(\sigma \smallfrown 0)$. We will often use this function f to represent an online function F . Since there is a computable enumeration of $\{0, 1\}^*$ in order first by length and then lexicographically, it follows that an online continuous function has a representation in $2^{\mathbb{N}}$. The online function F is said to be *online 1-random* if this representation is a 1-random real, and similarly for other notions of randomness. Online random functions were studied by Cenzer and Porter in [5]. The general notions of online random continuous functions and closed sets were developed by Barmpalias et al. in [1, 2].

A real $X \in 2^{\mathbb{N}}$ is said to be *weakly 1-random*, or *Kurtz random*, if it does not belong to any Π_1^0 class of measure 0. X is said to be Martin-Löf random if for every effective sequence S_1, S_2, \dots of c.e. open sets with $\mu(S_n) \leq 2^{-n}$, $x \notin \bigcap_n S_n$ (where μ is the uniform measure on $2^{\mathbb{N}}$). This can be straightforwardly extended to the space $\{0, 1, 2\}^{\mathbb{N}}$ or $k^{\mathbb{N}}$ for any finite k .

We want to consider the arithmetical complexity of subsets of \mathbb{N} as well as the effective Wadge complexity of subsets of $2^{\mathbb{N}}$. For subsets A and B of \mathbb{N} , we say that A is *many-one reducible* to B , written $A \leq_m B$, if there is a computable function ϕ such that, for all n , $n \in A \Leftrightarrow \phi(n) \in B$. Then we say that B is Π_2^0 complete if $A \leq_m B$ for every Π_2^0 set. Similar definitions apply for other complexity classes. It is well-known that $\{e : W_e \text{ is infinite}\}$ is Π_2^0 complete and that $\{e : W_e \text{ is cofinite}\}$ is Σ_3^0 complete. See [14] for these and other so-called index set results.

For subsets \mathcal{C}, \mathcal{D} of $2^{\mathbb{N}}$, we say that \mathcal{C} is *effectively Wadge reducible* to \mathcal{D} , written $\mathcal{C} \leq_w \mathcal{D}$, if there is a computable continuous function Φ such that, for all X , $X \in \mathcal{C} \Leftrightarrow \Phi(X) \in \mathcal{D}$. Then we say that \mathcal{D} is *lightface Π_2^0 complete* if $\mathcal{C} \leq_w \mathcal{D}$ for every Π_2^0 class. Similar definitions apply for other complexity classes.

Here are results from the folklore of this area that we will need. These can be found in [9], pp. 178–179. First, the set $LI = \{X \in \mathbb{N}^{\mathbb{N}} : \lim_n X(n) = \infty\}$ is Π_3^0 complete. Second, the class \mathcal{D} of all $X \in 2^{\mathbb{N}}$ such that $\{n : X(n) = 1\}$ is infinite, is lightface Π_2^0 Wadge complete. Finally, the class $\{X \in 2^{\mathbb{N}} : X \text{ is cofinite}\}$ is Σ_2^0 complete.

3 Continuity and Differentiability on $2^{\mathbb{N}}$

Here we want to consider a notion of differentiability for functions on $2^{\mathbb{N}}$. We define our derivative as follows:

Definition 1. Let $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ be a function. Then the derivative $F'(A)$ at a point $A \in 2^{\mathbb{N}}$ is the limit

$$\lim_{X \rightarrow A} \frac{d(F(X), F(A))}{d(X, A)}$$

if this exists, and then we say that F is *differentiable at A* . If F is differentiable for all A , we say F is *differentiable*.

If this definition were applied to real functions Φ , one would obtain $|F'(A)|$, since on the real line $d(X, A) = |X - A|$.

Note that for the standard metric d , the distance $d(U, V)$ between distinct $U, V \in 2^{\mathbb{N}}$ can only be of the form 2^{-k-1} for some k . Also, for a function F on $2^{\mathbb{N}}$, $d(F(X), F(Y)) \leq d(X, Y)$, so that $F'(A)$ can only equal 0, 1, or 2^{-k-1} for some k . When $F'(A) = 2^{-k}$, this means that in the limit, when X begins to disagree with A , it takes exactly k steps until $F(X)$ begins to disagree with $F(A)$.

Example 2. Let $F(X) = (0, X(0), X(1), \dots)$. It is clear that $d(F(X), F(A)) = \frac{1}{2}d(X, A)$, and so $F'(A) = \frac{1}{2}$ for all $A \in 2^{\mathbb{N}}$. Now we may consider the Choquet integral $G : 2^{\mathbb{N}} \rightarrow [0, 1]$ of the constant function $\frac{1}{2}$ mapping $2^{\mathbb{N}}$ to $[0, 1]$ which will be $G(X) = \frac{1}{2}r(X)$ and may be represented by F .

A key result for real continuous functions Φ is that Φ is constant if and only if $\Phi'(r) = 0$ for all reals r . Certainly a constant function on $2^{\mathbb{N}}$ will have derivative zero at all points. In the Cantor space, however, we can have a continuous function with a range of exactly k points for any finite k . Thus, we consider the following definition.

Definition 3. The function $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ is *almost constant* if there is a finite set $S \subseteq 2^{\mathbb{N}}$ such that for all X , $F(X) \in S$. If F is an online function with representative f as above, this means that there is some n such that for all σ of length $|\sigma| \geq n$, $f(\sigma \smallfrown 0) = f(\sigma \smallfrown 1)$. Note that F is *constant* if for all σ , $f(\sigma \smallfrown 0) = f(\sigma \smallfrown 1)$.

Proposition 4. *If the online function $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ is almost constant, then $F'(A) = 0$ for all $A \in 2^{\mathbb{N}}$.*

Somewhat surprisingly, this result is not reversible, as seen by the following example.

Example 5. Define the online function F by $F(X) = 0X(0)0X(1)0X(2)0\dots$. It is easy to check that if $d(X, A) = 2^{-n-1}$, then $d(F(X), F(A)) = 2^{-2n-1}$, so that

$$\frac{d(F(X), F(A))}{d(X, A)} = 2^{-n},$$

and it follows that $F'(A) = 0$. However, F is not almost constant and is in fact one-to-one.

Here is an online computable function which is nowhere differentiable.

Example 6. Let $F(X) = (X(0), 0, X(2), 0, X(4), 0, \dots)$ and fix a point $A \in 2^{\mathbb{N}}$. For each n , let $X_n = (A(0), A(1), \dots, A(2n - 1), 1 - A(2n), 0, 0, \dots)$ and let $Y_n = (A(0), A(1), \dots, A(2n - 2), 1 - A(2n - 1), 1 - A(2n), 0, 0, \dots)$. Then for all n , $d(F(X_n), F(A))/d(X_n, A) = 1$, whereas $d(F(Y_n), F(A))/d(Y_n, A) = \frac{1}{2}$. It follows that $F'(A)$ does not exist for any $A \in 2^{\mathbb{N}}$.

Every real number $r \in [0, 1]$ has a dyadic representation $X \in 2^{\mathbb{N}}$ so that $r = r_X = \sum_n X(n)2^{-n-1}$. r is said to be a dyadic rational if it has a finite representation $r = \sum_{i=0}^n e_i 2^{-i-1}$ for some finite string (e_0, \dots, e_n) and each dyadic rational r has two representations. For example $\frac{1}{2}$ is represented by $(1, 0, 0, \dots)$ and $(0, 1, 1, \dots)$. We will say that a function F on $2^{\mathbb{N}}$ represents a real function Φ , defined by $\Phi(r_X) = r_{F(X)}$, if whenever $r_X = r_Y$, then $r_{F(X)} = r_{F(Y)}$. The function F from Example 2 above represents the real function $\Phi(r) = \frac{1}{2}r$.

More generally, various piecewise linear functions on $[0, 1]$ with dyadic rational slopes may have online representations. Here is an example of a piecewise linear function with infinitely many segments.

Example 7. The function Φ on $[0, 1]$ will consist of line segments connected at the values $\Phi(2^{-n-1}) = \frac{1}{3}(1 + 2^{-2n-1})$, that is, $\Phi(1) = 1$, $\Phi(\frac{1}{2}) = \frac{1}{2}$, $\Phi(\frac{1}{4}) = \frac{3}{8}$, $\Phi(\frac{1}{8}) = \frac{11}{32}$, and so on. Finally, we let $\Phi(0) = \frac{1}{3}$. The representing function has $F(0^n 1 X) = (01)^n 1 X$.

We note that for $x = 2^{-n-1}$, we have $\Phi(x) = \frac{1}{3}(1 + 2x^2)$, so that the overall graph of Φ approximates a quadratic near $x = 0$.

4 Computability and Complexity

In this section, we determine the complexity of certain classes related to our notion of differentiability.

We will let the complexity of a given class of online functions to be given by the corresponding class $\mathcal{C} \subseteq 2^{\mathbb{N}}$ of representing functions. Given $r \in 2^{\mathbb{N}}$, let $f_r : \{0, 1\}^* \setminus \{\epsilon\} \rightarrow \{0, 1\}$ be defined by $f_r(\sigma_n) = r(n)$ and let $F_r \in \mathcal{F}(2^{\mathbb{N}})$ be defined so that $Y = F_r(X)$ is given by $Y(n) = f_r(X \upharpoonright n)$.

Theorem 8.

1. For any computable online function F , $\{X : F'(X) = 1\}$ is a Σ_2^0 class.

2. For any computable $A \in 2^{\mathbb{N}}$, $\mathcal{B} = \{r : F'_r(A) = 1\}$ is a lightface Σ_2^0 Wadge-complete subset of $2^{\mathbb{N}}$.
3. $\mathcal{AC} = \{r : F_r \text{ is almost constant}\}$ is a lightface Σ_2^0 Wadge-complete subset of $2^{\mathbb{N}}$.

Proof. The following lemma is needed.

Lemma. For any online function $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ with representing function $f : \{0, 1\}^* \rightarrow \{0, 1\}$, and any $X \in 2^{\mathbb{N}}$, $F'(X) = 1$ if and only if there exists n such that for all $m \geq n$, $f(X \upharpoonright m + 1) \neq f(X \upharpoonright m(1 - X(m)))$.

The fact that these first two classes are Σ_2^0 follows immediately from the preceding lemma. For the completeness of \mathcal{B} , we use the dual result from Sect. 2 that the class of finite subsets of \mathbb{N} is Σ_2^0 complete. Fix a computable $A \in 2^{\mathbb{N}}$. We will define a computable map Φ such that for $\Phi(Z) = r$, we have $Z \notin \mathcal{D} \Leftrightarrow F'_r(A) = 1$. We define r so that $f_r(\sigma \frown 0) = 0$ for all σ and such that $f_r(A \upharpoonright n \frown 1) = 1$ for all n such that $n \notin Z$ and $f_r(\sigma) = 0$ for all other strings σ . Thus for all $n \notin Z$, we have $F_r(A) \upharpoonright n(1 - A(n)) \frown X = 0^n 10^\omega$ and $F_R(Y) = 0^\omega$ for all other $Y \in 2^{\mathbb{N}}$. By the preceding lemma, we know that $F'_r(A) = 1$ if and only if $\{m : f_r(X \upharpoonright m + 1) \neq f_r(X \upharpoonright m(1 - X(m)))\}$ is finite, which will be if and only if Z is finite.

The fact that \mathcal{AC} is Σ_2^0 is immediate from the definition. For the completeness, we observe that for the function Φ given above, if $r = \Phi(Z)$, Z is cofinite if and only if F_r is almost constant. \square

Notice that the set of points X in which $F'(X) = 0$ for any computable online function F on $2^{\mathbb{N}}$ are more complicated than the set of X in which $F'(X) = 1$, as seen in the result below.

Theorem 9.

1. For any computable online function F , $\{X : F'(X) = 0\}$ is a Π_3^0 class.
2. For any computable $A \in 2^{\mathbb{N}}$, $\mathcal{A} = \{r : F'_r(A) = 0\}$ is a lightface Π_3^0 Wadge-complete subset of $2^{\mathbb{N}}$.

Proof. These are Π_3^0 classes, since $F'(A) = 0$ if and only if

$$(\forall k)(\exists m)(\forall n > m)(\forall X)[d(X, A) < 2^{-n} \rightarrow d(F(X), F(A)) < 2^{-n-k}].$$

This statement is equivalent to

$$(\forall k)(\exists m)(\forall n > m)(\forall \sigma \in \{0, 1\}^k)[f(A \upharpoonright n \frown \sigma) = f(A \upharpoonright n + k)].$$

Since the $(\forall \sigma)$ quantifier is bounded, this becomes a Π_3^0 statement.

For the completeness of the class \mathcal{A} , we let $A = 0^\omega$ (for the sake of simplicity) and define a reduction Ψ of the class LI from Sect. 2 as follows. Given $Z \in \mathbb{N}^{\mathbb{N}}$, let $r = \Psi(Z)$ be defined so that $F(0^\omega) = 0^\omega$ and, for each n , for each σ of length $Z(n)$, and for all $X \in 2^{\mathbb{N}}$, $F_r(0^n 1 \sigma X) = 0^{n+Z(n)+1} 1^\omega$. The representing function f_r maps 0^n to 0^n , maps $0^n 1 \sigma$ to $0^{n+|\sigma|}$ whenever $|\sigma| \leq Z(n) + 1$, and maps $0^n 1 \sigma \tau$ to $0^{n+Z(n)+1+|\tau|}$ whenever $|\sigma| = Z(n)$.

For any n , it follows from the construction there that for $r = \Psi(Z)$, and any extension Y_n of $0^n 1$, we have the following:

1. $d(Y_n, A) = 2^{-n-1}$;
2. $d(F(Y_n), F(A)) = 2^{-n-Z(n)-3}$;
3. $\frac{d(F(Y_n), F(A))}{d(Y_n, A)} = \frac{2^{-n-Z(n)-3}}{2^{-n-1}} = 2^{-Z(n)-2}$.

Now assume that $\lim_n Z(N) = \infty$. Then $F'(A) = \lim_{n \rightarrow \infty} 2^{-Z(n)-2} = 0$. On the other hand, if $\lim_n Z(n) \neq \infty$, then there exists some k such that $Z(n) = k$ for infinitely many n , say $n_1 < n_2 < \dots$. Now, for each i , let $Y_i = 0^{n_i}1^\omega$. It follows that $d(F(Y_i), F(A))/d(Y_i, A) = 2^{-k-2}$ for each i , so that $F'(A) \neq 0$. \square

Given a computable online function F , the complexity of the sets in Theorems 8 and 9 allows us to determine the computability of F' as shown below.

Proposition 10. *For any computable online function F , F' is (partial) Δ_3^0 computable.*

Proof. Consider F' as mapping $2^{\mathbb{N}}$ to the space $\{0\} \cup \{2^{-n} : n \in \mathbb{N}\} \subset [0, 1]$. Then we can let 0^ω represent the real number 0, and $0^n 10^\omega$ represent the real number 2^{-n} to get this. It follows from Theorems 8 and 9 that the graph of F' is a Π_3^0 class and therefore F' is Δ_3^0 computable. \square

As one would expect, the derivative of an online function need not be continuous, and the derivative of a computable online function need not be computable.

Example 11. Let $F(0^\omega) = 0^\omega$ and, for each n , each σ of length n and each $X \in 2^{\mathbb{N}}$, let $F(0^n 1 \sigma X) = 0^{2n} 1 X$. It is easy to see that $F'(0) = 0$ and that for any other element Y , $F'(Y) = 1$.

Example 12. Let E be any noncomputable c.e. set and define F so that again $F(0^\omega) = 0^\omega$ but now for each n , each σ of length n , and each $X \in 2^{\mathbb{N}}$, there are two cases. If $n \notin E$, then $F(0^n 1 \sigma X) = 0^{2n} 1 X$, as before. However, if n comes into E at stage s , then for τ of length s , we let $F(0^n 1 \sigma \tau X) = 0^{2n} 1 \tau 0^\omega$. It follows that, for any extension Y of $0^n 1$, if $n \notin E$, then $F'(Y) = 1$, but if $n \in E$, then $F'(Y) = 0$.

5 Randomness

In this section, we use the ideas from Sect. 4 to prove some results about randomness and the derivative of online functions. It is not hard to prove that a weakly 1-random online function cannot be differentiable at any computable point. In fact, we can show the following stronger result.

Theorem 13. *If F is a weakly 1-random online function, then F is not differentiable at any $A \in 2^{\mathbb{N}}$.*

Here is a sketch of the case involving computable points. Suppose that $A \in 2^{\mathbb{N}}$ is a computable point and suppose f represents an online function F . We will show that $F'(A) \neq 1$. Observe that $F'(A) = 1$ if and only if, for some n and all k ,

$f(A\upharpoonright(n+k)0) \neq f(A\upharpoonright(n+k)1)$. Fixing n , we note that for each k , the probability that $f(A\upharpoonright(n+k)0) \neq f(A\upharpoonright(n+k)1)$ in the space of online functions is $\frac{1}{2}$. Thus the Π_1^0 set $V = \{f : (\forall k)f(A\upharpoonright(n+k)0) \neq f(A\upharpoonright(n+k)1)\}$ has measure zero. It follows that if f is weakly 1-random, then $f \notin V$ and therefore $F'(A) \neq 1$.

Next we consider the weaker dual version of this result.

Theorem 14. *Let F is a computable online function on $2^{\mathbb{N}}$.*

- (a) *If A is weakly 1-random and $F'(A) = 2^{-p}$ for some $p \in \mathbb{N}$, then the set $\{X : F'(X) = 2^{-p}\}$ has positive measure.*
- (b) *If A is weakly 1-random and $F'(A) = 0$, then $\mu(\{X : F'(X) \leq 2^{-p}\}) > 0$ for any $p \in \mathbb{N}$.*
- (c) *If A is weakly 2-random and $F'(A) = 0$, then $\mu(\{X : F'(X) = 0\}) > 0$.*

Here we sketch the argument in the case that $F'(A) = 1$. Fix a computable online function F with representative f . As above, we can see that the Π_1^0 set $V = \{X : (\forall k)f(X\upharpoonright(n+k)0) \neq f(X\upharpoonright(n+k)1)\}$ has measure zero. It follows that if A is weakly 1-random, then $A \notin V$ and thus $F'(A) \neq 1$.

6 Representation of Real Functions

In this section, we present further results regarding the representation of certain classes of real-valued functions by online functions. We will want to compare the discrete distance function that we are using on $2^{\mathbb{N}}$ with the usual distance function on real numbers.

A key fact is that for an online function F on $2^{\mathbb{N}}$, we know that for any A, B , $d(F(A), F(B)) \leq d(A, B)$ since $A\upharpoonright n = B\upharpoonright n$ implies that $F(A)\upharpoonright n = F(B)\upharpoonright n$. Recall that $r_X = \sum_i X(i)2^{-i-1}$. It is not hard to see that $|r_A - r_B| \leq 2d(A, B)$. It follows that the function from $2^{\mathbb{N}}$ to $[0, 1]$ mapping X to r_X is continuous.

On the other hand, note that for $A = 10^\omega$ and $B = 01^\omega$, we have $d(A, B) = \frac{1}{2}$ whereas $r_A = r_B$. So for a continuous function F which represents a function Φ on the real interval, the comparison between $|\Phi(r) - \Phi(s)|$ and $|r - s|$ is not immediately clear.

Theorem 15. *Suppose that the function F on $2^{\mathbb{N}}$ represents a function Φ on $[0, 1]$. Then for any reals $r, s \in [0, 1]$, $|\Phi(r) - \Phi(s)| \leq |r - s|$.*

Now the property that $|\Phi(r) - \Phi(s)| \leq |r - s|$ is an example of a Lipschitz condition on the function Φ . This condition clearly implies that Φ is continuous. In addition, we may now conclude the following from well-known properties of functions satisfying the Lipschitz condition (for which we refer the reader to Royden [13]):

Theorem 16. *If the online function $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ represents a function Φ on $[0, 1]$, then Φ is differentiable almost everywhere. Furthermore, for all $x \in [0, 1]$, $|\Phi'(x)| \leq 1$ if it exists.*

The next result follows from Theorem 13.

Corollary 17. *No weakly random online function F can represent a real function.*

We next revisit the problem of functions with derivative zero.

Proposition 18. *Suppose that the function F on $2^{\mathbb{N}}$ represents a function Φ on $[0, 1]$ and that $F'(A) = 0$ and $\Phi'(r_A)$ is defined. Then $\Phi'(r_A) = 0$.*

Proof. Consider the sequence $A_k = (A \lceil k(1 - A(k)), A(k + 1), \dots)$ and $r_k = r_{A_k}$. Then $d(A, A_k) = 2^{-k} = |r_A - r_k|$. Fixing n , we have, since $F'(A) = 0$, that for sufficiently large k , $d(F(A), F(A_k)) \leq 2^{-n-k}$. From this, we obtain that $|\Phi(r_A) - \Phi(r_k)| \leq 2^{1-n-k}$, and so

$$\frac{|\Phi(r_A) - \Phi(r_k)|}{|r_A - r_k|} \leq 2^{1-n}.$$

Since we are assuming that $\Phi'(r_A)$ exists, it follows that $\Phi'(r_A) = 0$. □

Now we can revisit the situation where $F'(A) = 0$ for every A .

Proposition 19. *Suppose that the function F on $2^{\mathbb{N}}$ represents a function Φ on $[0, 1]$ and that $F'(A) = 0$ for all A and $\Phi'(r)$ exists for all r . Then Φ is a constant function and F is almost constant.*

Proof. It follows from Proposition 18 that $\Phi'(r) = 0$ for all $r \in [0, 1]$ and hence Φ is constant. If the constant is not a dyadic rational, then F is constant, and otherwise F may take on at most two values, and hence is almost constant. □

Certainly any constant function on $[0, 1]$ may be represented by an online function. Now we consider which other linear functions may be represented.

Proposition 20. *For any $n > 0$ and $m = 2^{-n}$, and any $b = \frac{k}{2^{n+1}}$ such that $m + b \leq 1$, the function $\Phi(X) = mX + b$ on $[0, 1]$ may be represented by an online function F with $F'(X) = m$ for all X .*

Proof. First, let $m = \frac{1}{2}$. The function $\frac{1}{2}X$ may be represented by $F(X) = 0X$. Similarly, the function $\frac{1}{2}X + \frac{1}{2}$ may be represented by $F(X) = 1X$. Finally, the function $\frac{1}{2}X + \frac{1}{4}$ may be represented by $F(0X) = (01X(1)X(2) \dots)$ and $F(1X) = (10X(1)X(2) \dots)$.

Here we see that $f(0\sigma) = 01\sigma(1)\sigma(2) \dots$ and $f(1\sigma) = 10\sigma(1)\sigma(2) \dots$. Thus if $|\sigma| = n + 1 \geq 2$, then $f(\sigma 0) = f(\sigma 1) = f(\sigma)\sigma(n)$, while $f(\sigma 0i) = f(\sigma)\sigma(n)0$ but $f(\sigma 1i) = f(\sigma)\sigma(n)1$. It follows that $F'(X) = \frac{1}{2}$ for all X .

More generally, let $n > 1$ be given, let $m = 2^{-n}$ and let $b = k/2^{n+1}$ so that $m + b \leq 1$. Then $mX + b$ may be represented by the sum $2^{-n}X$ with the constant function b . The former is represented by 0^nX and the latter is represented by a string δ_k of length $n + 1$. These can be added together in cases, depending on the first bit of X , and the sum will not overflow since $m + b \leq 1$. That is, $m + b \leq 1$ implies that $k + 1 \leq 2^{n+1}$. So we can define $F(0X) = \delta_kX$ and $F(1X) = \delta_{k+1}X$.

It can be checked as above that $F'(X) = 2^{-n}$ for all X . □

Using the map that switches 1s and 0s, we see that if F is represented by f , then $G(X) = 1 - F(X)$ may be represented by $g(\sigma) = 1 - f(\sigma)$, that is if $\tau = f(\sigma)$, then $g(\sigma) = (1 - \tau(0), 1 - \tau(1), \dots)$. Thus we may represent $G(X) = 1 - (mX + b)$ for m and b as above.

For the converse direction, we weaken the $n+1$ to $2n$. Note that $\Phi(r) = \frac{1}{4}r + \frac{1}{16}$ may be represented by a function F on $2^{\mathbb{N}}$, but $\Psi(r) = \frac{1}{4}r + \frac{7}{16}$ may not be so represented.

Theorem 21. *If a linear function $\Phi(X) = mX + b$ (with $m \neq 0$) maps $[0, 1]$ into $[0, 1]$ and is represented by an online function, then $m = \pm 2^{-n}$ for some n and b is a dyadic rational of the form $\frac{k}{2^{2n}}$ for some k .*

We just note here that, of course, $m \leq 1$ since the function would otherwise not map $[0, 1]$ into $[0, 1]$. Functions on finite strings (and in particular on natural numbers) that are computable by transducers (finite automata) are a special case of online functions. Such functions were studied by Frougny and Sakarovitch [7].

References

1. Barmpalias, G., Brodhead, P., Cenzer, D., Dashti, S., Weber, R.: Algorithmic randomness of closed sets. *J. Log. Comput.* **17**, 1041–1062 (2007)
2. Barmpalias, G., Cenzer, D., Remmel, J., Weber, R.: k -triviality of closed sets and continuous functions. *J. Log. Comput.* **19**, 3–16 (2009)
3. Brattka, V., Miller, J., Nies, A.: Randomness and differentiability. *Trans. Am. Math. Soc.* **368**, 581–605 (2016)
4. Brodhead, P., Cenzer, D., Toska, F., Wyman, S.: Algorithmic randomness and capacity of closed sets. *Log. Methods Comput. Sci.* **6**, 1–16 (2011)
5. Cenzer, D., Porter, C.P.: Algorithmically random functions and effective capacities. In: Jain, R., Jain, S., Stephan, F. (eds.) TAMC 2015. LNCS, vol. 9076, pp. 23–37. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17142-5_4
6. Cenzer, D., Remmel, J.: Index sets for computable differential equations. *Math. Log. Q.* **50**, 329–344 (2004)
7. Frougny, C., Sakarovitch, J.: Number representation and finite automata. In: Combinatorics, Automata and Number Theory. *Encyclopedia of Mathematics and its Applications*, vol. 135, pp. 34–107. Cambridge University Press (2010)
8. Heinonen, J.: Nonsmooth calculus. *Bull. Am. Math. Soc.* **44**, 163–232 (2007)
9. Kechris, A.: *Classical Descriptive Set Theory*. Graduate Texts in Mathematics, vol. 156. Springer, New York (1995). <https://doi.org/10.1007/978-1-4612-4190-4>
10. Martin-Löf, P.: The definition of random sequences. *Inf. Control* **9**, 602–619 (1966)
11. Myhill, J.: A recursive function, defined on a compact interval, and having a continuous derivative that is not recursive. *Mich. Math. J.* **18**, 97–98 (1971)
12. Pour-El, M., Richards, J.: A computable ordinary differential equation which possesses not computable solution. *Ann. Math. Log.* **17**, 61–90 (1979)
13. Royden, H.L.: *Real Analysis*, 3rd edn. Macmillan Publishing Company, New York (1988)
14. Soare, R.I.: *Recursively Enumerable Sets and Degrees*. Springer, Heidelberg (1987)



Turing's Vision and Deep Learning

Martin Davis^(✉)

Courant Institute, New York University, New York, USA
martin@eipye.com

Abstract. The success of the AlphaGo computer program in playing world class Go is examined in connection with what Turing had envisioned more than 70 years ago. A critical discussion is given of Searle's comments on intelligent computer programs and of what Copeland had to say about this.

Keywords: Turing · AlphaGo · Searle · Copeland
Convolutional neural network

An issue of the British newspaper *The Guardian* in October 2017 featured an article on the achievements of DeepMind's most recent incarnation of their Go-playing machine. Their AlphaGo program had previously learned to play this deep ancient Chinese game well enough to defeat human grandmasters. It accomplished this after being trained using thousands of games that had been played between grandmasters. The newer version "AlphaGo Zero" learned by playing thousands of games against itself thereby steadily improving. Today it beats all comers including the original AlphaGo. Both versions made use of multi-layer neural networks. These are imagined as consisting of a large number of individual elements called "neurons" interconnected so that signals output from each became the input to others. One may imagine a tangle of interconnected wires resembling the tangle of neural connections in a brain. However, entering a room in which a device like AlphaGo is housed, one would see only ordinary-looking computer equipment.

In this paper, I will discuss how Turing imagined the future of computation at a time when the first computers were yet to be built, and how this relates to contemporary developments.

1 Alan Turing in 1945

The Second World War was over, but the contributions Turing had made to victory at Bletchley Park were to remain secret for a long time. But he still had knowledge he had acquired there of the use of vacuum tube (British: valves) circuits in carrying out logical operations. And he understood the relevance of the new expansive view of the nature of computation that had emerged from his theoretical investigation of computability before the war. This led to his

ACE report outlining the construction and use of a general purpose digital computer [13]. A comparison of this report with von Neumann's Edvac report is revealing of Turing's very different outlook. Whereas, along with addition and multiplication, von Neumann would provide circuitry for arithmetic division as well as, astonishingly, even for obtaining square roots, the ACE report suggests performing division via software, but did specify circuitry for basic digit-by-digit Boolean logical operations. Also two of the ten problems Turing discusses as possibly appropriate for the ACE, solving simple jigsaw puzzles and playing chess, are quite remote from numerical computation.

In 1947 in concluding an address on the ACE to the London Mathematical Society [14], Turing said:

It has been said that computing machines can only carry out the processes that they are instructed to do. This is certainly true in the sense that if they do something other than what they were instructed then they have just made some mistake. It is also true that the intention in constructing these machines in the first instance is to treat them as slaves, giving them only jobs which have been thought out in detail, jobs such that the user of the machine fully understands what in principle is going on all the time. Up till the present machines have only been used in this way. But is it necessary that they should always be used in such a manner? Let us suppose we have set up a machine with certain initial instruction tables,¹ so constructed that these tables might on occasion, if good reason arose, modify those tables. One can imagine that after the machine had been operating for some time, the instructions would have altered out of all recognition, but nevertheless still be such that one would have to admit that the machine was still doing very worthwhile calculations. Possibly it might still be getting results of the type desired when the machine was first set up, but in a much more efficient manner. In such a case one would have to admit that the progress of the machine had not been foreseen when its original instructions were put in. It would be like a pupil who had learnt much from his master, but had added much more by his own work. When this happens I feel that one is obliged to regard the machine as showing intelligence. As soon as one can provide a reasonably large memory capacity it should be possible to begin to experiment on these lines. The memory capacity of the human brain is probably of the order of ten thousand million binary digits. But most of this is probably used in remembering visual impressions, and other comparatively wasteful ways. One might reasonably hope to be able to make some real progress with a few million digits, especially if one confined one's investigations to some rather limited field such as the game of chess. It would probably be quite easy to find instruction tables which would enable the ACE to win against an average player. Indeed Shannon of Bell Telephone laboratories tells me that he has won games, playing by rule of thumb: the skill of his opponents

¹ Turing had introduced the term *instruction table* for what came to be called a computer program. As I write, one usually speaks of an *app*.

is not stated. But I would not consider such a victory very significant. What we want is a machine that can learn from experience. The possibility of letting the machine alter its own instructions provides the mechanism for this, but this of course does not get us very far.

2 Turing and Machine Intelligence

Famously Turing proposed as a goal, the ability of a machine to carry on a conversation with a person so effectively that it would be difficult or impossible for the person to tell whether the conversation had been with a person or a machine [15]. This has led to a huge discussion of “Turing’s test,” and a proliferation of claims that various programs had “passed” the test, mainly based on conversations severely restricted in subject matter. In a brief forward to the Turing Centenary reprint of Sara Turing’s biography of her son [18], I wrote, “The criterion he chose was the ability of such a machine to carry on a conversation that could not reasonably be distinguished from one by a person. He predicted that this would be achieved by the end of the twentieth century, but was far too optimistic about the task of programming computers to achieve a command of natural language equivalent to that of every normal person”. I’m grateful to Jack Copeland for calling my attention to the fact that Turing’s prediction of what would be accomplished by the end of the century was far more modest.²

I believe that in about fifty years’ time it will be possible to programme computers . . . to play the imitation game so well that an average interrogator will have not more than a 70 per cent chance of making the right identification after five minutes of questioning. . . . I believe that at the end of the century the use of words and generally educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.

I would claim that even this modest claim was too optimistic if the machine is to exhibit anything like the verbal fluency of an adolescent or adult. Reading what Turing had to say about teaching a “child computer” with rewards and punishments, that is not at all surprising. Very young children seem to acquire language with an ease that adults seeking to learn a “second” language can only envy. Referring to my error, Copeland ([4] p. 272) refers to me as “one of Turing’s critics”, and to what I said as “misguided criticism”. It wasn’t criticism at all, it was an error, evidence of sloppy scholarship on my part. As Turing wisely said at the end of his provocative essay, “We can only see a short distance ahead”; I could never imagine criticizing Turing concerning his skill as a prophet.

I one of Turing’s critics? In 1987, Turing was still being totally ignored in discussions of the early history of computers, discussions that revolved around the question of whether von Neumann had improperly denied the credit for their

² I was embarrassed to find that I had made the same error in my book [6, 7]. Fortunately I’ve been able to correct it for the third edition, in press as I write.

contribution to the “stored program concept” that the engineers Eckert and Mauchly purportedly deserved. My essay [5], published in that year, explained the significance of Turing’s theoretical investigations contributions as providing the conceptual basis for modern computers. David Leavitt, in his biography of Turing [10], mentioned “von Neumann often being given credit for ideas originating with Turing”, and wrote: “Martin Davis must be credited with setting the record straight on this account”.

Although I cannot fairly be called a critic of Turing, and although I very much admire Copeland’s extensive and enthusiastic work in presenting and advocating for Turing’s crucial contributions, I certainly have been a critic of Copeland. His advocacy has at times led him to farfetched notions, as when he proclaimed that the “the search is on” for the construction of a Turing oracle that could provide answers to uncomputable problems [1,9].

I was astonished to find that Copeland takes seriously Searles fanciful Chinese room as having anything significant to say about Turing’s ideas. Copeland takes Searle seriously as “one of AI’s greatest critics”. He seems to believe that the Chinese room provides a serious challenge with “a subtle fallacy” ([4], pp. 274–275). Copeland responds with a clever verbal riposte such as one can hear in the discussion following an academic philosophy seminar talk. I prefer to discuss what Searle has to say in terms of a domain in which computers can already perform quite creditably: chess. Fortunately, Searle has also provided us with a chess room [12]:

Imagine that a man who does not know how to play chess is locked inside a room, and there he is given a set of, to him, meaningless symbols. Unknown to him, these represent positions on a chessboard. He looks up in a book what he is supposed to do, and he passes back more meaningless symbols. We can suppose that if the rule book, i.e., the program, is skillfully written, he will win chess games. People outside the room will say, “This man understands chess, and in fact he is a good chess player because he wins”. They will be totally mistaken. The man understands nothing of chess, he is just a computer. And the point of the parable is this: if the man does not understand chess on the basis of running the chess-playing program, neither does any other computer solely on that basis.

As with the Chinese room, Searle and Copeland can assure us that “the individual steps” are “simple binary operations that a human being can easily carry out . . . given enough time”. Because chess playing programs (and very good ones at that) exist, one can calculate how much time is “enough time”. As I write, I have no doubt that chess playing programs exist that are a lot better than Deep Blue, the program that created a sensation when it defeated Kasparov. And Deep Blue was a better player than its predecessor Deep Thought. After obtaining some data from a member of the team that had designed Deep Thought, I calculated that it would require an average of more than a year of the man’s time to carry out enough of those “simple steps” for making a single move. Searle nevertheless insists [12]:

Here is what is going on inside Deep Blue. The computer has a bunch of meaningless symbols that the programmers use to represent the positions of the pieces on the board. It has a bunch of equally meaningless symbols that the programmers use to represent options for possible moves. The computer does not know that the symbols represent chess pieces and chess moves, because it does not know anything.

Some academic philosophers like Searle may enjoy regarding computers as knowing nothing, but a human chess player who has lost his queen to a fiendishly clever trap, will retort, “Well it certainly *knows* how to play chess”. I have provided the following version of Searle’s parable [6–8]:

A precocious child whose mother is passionate about chess becomes tired of watching her play and demands that he be allowed to play her opponent. His mother agrees on the condition that he move the pieces only when she tells him to and exactly where she says. He does as requested and doing what his mother whispers in his ear achieves a checkmate. Observing the scene, Searle tells us that the child doesn’t know anything about chess, and is certainly not playing chess. Who could disagree?

And here is my analogous parable for the famous Chinese room itself, a spy story:

A double agent Nak Moon, who is Korean, is pretending to be fluent in Chinese. Actually he doesn’t understand a word of it, but is an excellent mimic. He is being questioned in Chinese by his interlocutor via a video connection. Amy Chung, who was born and grew up in Shanghai, is in an adjacent room. She hears everything, and can speak to Nak without his interlocutor knowing, by means of a hidden earphone. She tells Nak what to reply to the questions being thrown at him. The dialog ends with the interlocutor satisfied and Nak Moon still totally ignorant of the meaning of what he has been saying.

Searle tells us that Nak Moon knows no Chinese. Who could disagree? Copeland concludes his discussion of Searle’s Chinese room, by writing:

Alan Turing’s test has been attacked by some of the sharpest minds in the business. To date, however, it stands unrefuted. In fact, it is the only viable proposal on the table for testing whether a computer is capable of thought ([4] p. 275).

I cannot claim to share Copeland’s apparent ability to assess the relative “sharpness” of the minds of those who engage in discussions regarding computer thought. However, I will note that multi-level neural network software together with the sheer power of contemporary hardware has enabled computer systems to perform as well as and even better than people, in doing things that, when people do them, certainly require thought. I don’t believe that one can have a meaningful discussion of whether what these devices are doing constitutes thinking in the human sense until neuroscientists have some real understanding what

it is that we do, at the level of individual neurons, when we think. In the meantime, we can safely leave discussions of whether these remarkable systems are *really* thinking to those philosophers who regard this as worthwhile.

3 Neural Networks and AlphaGo

The idea of, a “neural network” that sought to emulate a brain by imagining an interconnected network of objects, each having some of the characteristics of biological neurons, occurred to a number of researchers. The foundational article [11] by McCulloch and Pitts was the first to elaborate a mathematical theory of such structures. Turing himself in his 1948 memo to his boss at the National Physical Laboratory, [17]³ considered several models of such neural networks. Other early work in this area were done by Marvin Minsky, Frank Rosenblatt, and Michael Arbib. The neurons considered in contemporary networks have several input channels that receive numerical signals from other neurons or from their external environment and one output channel that transmits such signals. Each input channel has a number associated with it called its *weight*; at each stage the signal transmitted by a neuron is the weighted average of the input values. Although weights are already present in [11], there were no weights in Turing's neurons.

Contemporary neural networks usually consist of at least three interconnected layers. In a three-layer network, the neurons in the first layer receive input signals and transmit their output to neurons in the second layer. These neurons in turn transmit their output to the third layer which generates the output signal. In order to obtain worthwhile results, at least some of the neurons must apply a suitable non-linear *activation function* to the weighted average of its input signals to produce its output. The choice of an appropriate activation function in the design is crucial. The hyperbolic tangent

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

is an example of an activation function with desirable properties. Learning takes place by *training* the network with a collection of examples. A *back propagation* algorithm is applied after each example. This algorithm is designed to reduce the error between the desired output and the actual output. This process is what is meant by *deep learning*.

Such deep learning techniques have been very successful in training neural networks to identify specific objects in a video image. Video images exist in the form of an array of pixels stored in a computer. Neural networks have been

³ Jack Copeland, in a careful detailed introduction to this essay, pointed out that its previous publication, [16] pp.107–132, contained serious errors. I should also mention that in [1], the article I have criticized for its embrace of hypercomputation, information about Turing's early comments on neural nets was brought to public attention.

trained to distinguish a bird from a squirrel and to identify individual human faces. The networks that accomplish such feats are usually designed to be “convolutional”: what people and animals accomplish by scanning a visual field through eye movements, a convolutional neural network accomplishes by carrying out the enormous number of computations needed to recognize that objects in different parts of a video image are the same kind of thing. These computations are greatly facilitated by having special “graphical processing unit” chips as part of the hardware. These GPU chips were originally designed and manufactured for machines intended for playing computer games, and make very fast complex numerical computations feasible. It needs to be emphasized that these networks are algorithms, typically implemented, not as a large box of wriggling interconnected little gadgets, but rather as lines of code written in a modern high level programming language. The network can be thought of as a kind of dynamic data structure.

When I was preparing the third edition of my *The Universal Computer* [6–8], just published as I write, there was general astonishment over a remarkable achievement by AlphaGo, a machine for playing the ancient game of Go. Go is much more complex than chess. Where chess is played on a board with 64 squares on which the pieces can be placed, Go is played on a 19×19 board with 361 places where pieces, called stones, can be placed. Like chess the winner is entirely determined by the moves the players make, taking alternate turns placing a stone in an unoccupied place. An algorithm to play Go must at each stage choose a branch from a tree that is very long and extremely wide. Playing against the Go master Lee Sedol in a five game match, AlphaGo won four of the games. The prevailing opinion had been that a Go machine capable of beating human master players was still many years in the future. I wanted to write something about this, but I was ignorant. I had been a skeptic about neural networks and had never learned much about them. Fortunately Thore Graepelof, one of the computer engineers in the DeepMind group that had developed AlphaGo, agreed to an hour-long interview from London by video connection, courtesy of Google. He turned out to be a clear and patient explainer.

AlphaGo uses a Monte Carlo tree search algorithm taking advice from two auxiliary convolutional neural networks: a *policy* network and a *value* network. Both networks were trained by giving them access to a huge library of games played between expert players over many years. The policy network estimates how likely various possible moves are to be played by an expert Go player starting from a particular arrangement of pieces on the Go board. Thore told us, “This policy network was already a pretty good Go player. I am a pretty good amateur player myself. On my first day at DeepMind, David Silver invited me to play against this network, and I was sure I would win. But it beat me! Then I was sure I wanted to be part of the AlphaGo team”. The other network, the value network, starting with any Go position, estimates for one of the players, the probability that player will win. Convolution plays a role similar to that in object recognition: just as a bird in one corner of a video image needs to be recognized as still a bird when it’s in a different part of the image, a certain

configuration of the stones in one part of the Go board needs to be recognized as similar to one in a different part. It should be emphasized that the success of AlphaGo is due not only to the effectiveness of the software design, but also to the power of contemporary hardware. In particular, AlphaGo uses GPUs for intensive computation.

In 2016 AlphaGo was brought to Korea to challenge Lee Sedol to a five game match. Thore said, “Although we had great confidence in AlphaGo, you never know what chance will come up with in a tournament. Fortunately, AlphaGo won the first three games. By the fourth game, our team was actually rooting for Lee Sedol, and was happy for him when he succeeded”. Then AlphaGo won the fifth game. In May 2017 at the Future of Go Summit in China, AlphaGo won all three games against world champion Ke Jie. Thore continued, “Go players have started to pick up patterns of play that AlphaGo invented. In training for the May competition, we let AlphaGo play against itself and create new games of higher quality, resulting in a set of training data for a stronger version of AlphaGo. With this kind of bootstrapping, a machine learning system can continue to grow”.

4 Turing's Vision

When Turing began writing about the capabilities of computers in the 1940s, there were still no working computers. When the first computers did come on line, they had severe limitations. Comparing them to the objects we call phones and carry in our pockets, we might say metaphorically, would be like comparing an earthworm to a monkey. Programming them had to be done in terms of binary machine code. Yet his audacious comments spoke of far-reaching accomplishments to come:

In his 1945 detailed proposal for his Automatic Computing Engine (ACE) [13], Turing listed chess as one of ten problems that might be appropriate for the machine. He suggested that it could be programmed to “display intelligence” and to play “very good chess”. We have already mentioned Turing's 1947 lecture on the ACE to the London Mathematical Society [14]. He suggested that given the possibility of changing its own program, the machine might be able to improve its performance. He asserted, “When this happens I feel that one is obliged to regard the machine as showing intelligence”. He ended his address mentioning chess as an appropriate arena for the machine to learn from exposure to human opponents.

In 1948, Turing presented an essay entitled *Intelligent Machinery* as an internal memo to his superior at The National Physical Laboratory [17]. The far-sighted comments are particularly astonishing considering the state of computer hardware, with the first crude true stored program computers just being built. As problems appropriate for machines like the ACE, Turing mentioned, games like chess or bridge, learning languages, translation, cryptography. He discussed several kinds of simple networks constructed from neuron-like elements, and pointed out that the networks could be fully realized by programming a machine like the

ACE. He envisioned programming a computer to carry out formal deductions in a system like *Principia Mathematica*, and using it to search for solutions of mathematical problems. He suggested both the possibility of a computer modifying its own program and also that it might be useful to introduce a random element. It can't be said that Turing imagined something quite like AlphaGo. However, with its two neural nets and its Monte Carlo search algorithm, one can find an echo of Turing's imaginings

Given what has already been achieved it is exciting to imagine the further developments that will now be coming. May we hope that human society will find itself able to deal intelligently and equitably with them as they arrive?

References

1. Copeland, B.J., Proudfoot, D.: Alan Turing's forgotten ideas in computer science. *Sci. Am.* **253**(4), 98–103 (1999)
2. Copeland, B.J. (ed.): *The Essential Turing*. Oxford University Press, New York (2004)
3. Copeland, B.J., et al.: *Alan Turing's Electronic Brain: The Struggle to Build the ACE, the World's Fastest Computer*. Oxford University Press, Oxford (2005)
4. Copeland, B.J., et al.: *The Turing Guide*. Oxford University Press, Oxford (2017)
5. Davis, M.: Mathematical logic and the origin of modern computers. In: *Studies in the History of Mathematics*, pp. 137–165. Mathematical Association of America (1987). Reprinted in *The Universal Turing Machine - A Half-Century Survey*, Herken, R. (ed.), pp. 149–174. Verlag Kemmerer & Unverzagt/Oxford University Press, Hamburg/Oxford (1988)
6. Davis, M.: *The Universal Computer: The Road from Leibniz to Turing*. W.W. Norton, New York City (2000). Second (Turing Centenary) Edition. CRC Press, Taylor & Francis (2012)
7. Davis, M.: *Engines of Logic: Mathematicians and the Origin of the Computer*. W.W. Norton, New York City (2001). Paperpack edition of [6]
8. Davis, M.: *The Universal Computer: The Road from Leibniz to Turing*, 3rd edn. Taylor & Francis/CRC Press, Boca Raton (2018)
9. Davis, M.: The myth of hypercomputation. In: Teuscher, C. (ed.) *Alan Turing: Life and Legacy of a Great Thinker*, pp. 195–212. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-662-05642-4_8
10. Leavitt, D.: *The Man Who Knew Too Much: Alan Turing and the Invention of the Computer*. W.W. Norton, New York City (2006)
11. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133 (1943). Reprinted in McCulloch, W.S., *Embodiments of Mind*, pp. 19–39. M.I.T. Press, Cambridge (1965)
12. Searle, J.R.: I married a computer. *N. Y. Rev. Books* **46**, 34–38 (1999)
13. Turing, A.: Proposed Electronic Calculator. Proposal submitted to the Mathematics Division of the National Physical Laboratory (1945). Reprinted in [16], pp. 1–86. Reprinted in [3], pp. 369–454
14. Turing, A.: The Automatic Computing Engine. Lecture delivered to the London Mathematical Society, 20 February 1947. Reprinted in [16], pp. 87–105. Reprinted in [2], pp. 378–394
15. Turing, A.: Computing machinery and intelligence. *Mind* **LIX**, 433–460 (1950). Reprinted in [16], pp. 133–160. Reprinted in [2], pp. 433–464

16. Turing, A.: *Collected Works: Mechanical Intelligence*, Ince, D.C. (ed.) North-Holland, Amsterdam (1992)
17. Turing, A.: *Intelligent Machinery*. [2], pp. 410–432. [16], pp. 107–127
18. Turing, S.: *Alan M. Turing, Centenary Edition*. Cambridge University Press, Cambridge (2012)



Computing and Scheduling with Explorable Uncertainty

Thomas Erlebach^(✉) 

Department of Informatics, University of Leicester, Leicester, England
t.erlebach@leicester.ac.uk

Abstract. Explorable uncertainty refers to settings where parts of the input data are initially unknown, but can be obtained at a certain cost using queries. In a typical setting, initially only intervals that contain the exact input values are known, and queries can be made to obtain exact values. An algorithm must make queries one by one until it has obtained sufficient information to solve the given problem. We discuss two lines of work in this area: In the area of query-competitive algorithms, one compares the number of queries made by the algorithm with the best possible number of queries for the given input. In the area of scheduling with explorable uncertainty, queries may correspond to tests that can reduce the running-time of a job by an a priori unknown amount and are executed on the machine that also schedules the jobs, thus contributing directly to the objective value of the resulting schedule.

1 Introduction

In many real-world settings, parts of the input data of a problem may not be known precisely. For example, instead of having the exact values of some input parameters, only intervals containing the exact values may be known. If the inputs are points in a higher-dimensional space, only regions containing the exact point locations may be known. We are interested in settings where it is possible to obtain the exact value of such an uncertain input element using a *query*. As this means that an algorithm can obtain additional information about the uncertain elements of the input data, such settings are referred to as *explorable uncertainty* (or *queryable uncertainty*). Typically, queries are associated with a cost and one wants to make as few queries as possible until one has obtained sufficient information to solve the given problem. Using competitive analysis [1], one can compare the number of queries made by an algorithm with the smallest possible number of queries that are sufficient to solve the problem. An algorithm is called ρ -query-competitive, or simply ρ -competitive, if it makes at most ρ times as many queries as an optimal solution for each given instance. Another direction is to consider scheduling problems where the execution of queries is part of the schedule for the actual jobs, and hence the time spent on queries contributes naturally to the objective value of the schedule produced.

Some examples of application settings that motivate the study of explorable uncertainty are as follows. If the input to a computation consists of the locations

of moving objects (e.g., planes or mobile wireless devices) and the exact locations of those objects were known at some point in the past, the current locations of these objects are uncertain, but it is possible to obtain an object's exact location, e.g., using radio communication with a plane or a message exchange with a mobile device [8]. In settings with a central master database and distributed database caches that are updated less frequently than the central database, a local database cache may provide approximate values as the input to a computation, but exact values can be obtained using a query to the master database [11]. In repair or maintenance work, the time required for a specific repair job may be uncertain, but can be determined using fault diagnosis [9,12].

The remainder of this paper is structured as follows. Section 2 discusses selected work on query-competitive algorithms for explorable uncertainty, while Sect. 3 covers work on a scheduling problem where the execution of queries is part of the produced schedule itself. Section 4 suggests some directions for future work.

2 Query-Competitive Algorithms

In a problem with explorable uncertainty, for some input elements u only an uncertainty set A_u that is guaranteed to contain the exact value w_u is given, but the algorithm must query u in order to obtain w_u . For example, the weight of an edge e in a weighted graph might be given as an uncertainty set in the form of an open interval, while the exact weight of the edge can be an arbitrary value in that interval.

Query-competitive algorithms for problems with explorable uncertainty were first considered by Kahan [8]. He studies problems where the input consists of n uncertain values, each represented as a closed interval or a singleton set. For the problem of identifying all elements whose value is equal to the maximum of the n values, and for the problem of identifying all elements whose value is equal to the median of the n values, he presents algorithms using at most $OPT + 1$ queries, where OPT is the minimum possible number of queries that suffice to solve the problem. Bruce et al. [2] study geometric problems where the input consists of n uncertain points in the plane, each represented by a singleton set or the closure of a connected, open set. For the problem of computing the set of maximal points or the set of points that lie on the convex hull, they present 3-competitive algorithms and show that this is best possible for deterministic algorithms. Their algorithms are based on the concept of witness sets: A *witness* set is a set of input elements with the property that it is impossible to solve the problem without querying at least one element of the set. A witness set algorithm repeatedly identifies a small witness set and queries all elements of that witness set, until the problem can be solved without further queries. If every witness set has size at most k , the resulting algorithm is k -competitive. Erlebach et al. [7] present witness set algorithms for the minimum spanning tree problem with uncertainty. For the case of uncertain edge weights with uncertainty sets that are singleton sets or open intervals, they present a 2-competitive deterministic

algorithm and show a matching lower bound for deterministic algorithms. A randomized algorithm achieving competitive ratio $1 + 1/\sqrt{2} \approx 1.707$ for the same problem was presented by Megow et al. [10], exploiting a connection to the bipartite vertex cover problem (a more restricted version of that connection had been used to determine the off-line complexity of computing the optimal query set in [4]).

Query-competitive algorithms for cheapest set problems were presented in [6]. Here, the input consists of a set E of elements whose weights are uncertain, and a family \mathcal{S} of subsets of E . The uncertainty set A_e of an element $e \in E$ can be an open interval or a singleton set, and the exact weight of e is denoted by w_e . The goal is to identify a set $S \in \mathcal{S}$ of minimum weight using a minimum number of queries. For the case that each set in \mathcal{S} has cardinality at most d , a straightforward application of the witness set algorithm approach yields a $2d$ -competitive algorithm. In [6] it was shown that bounds with a better multiplicative factor can be achieved using algorithms that do not always query witness sets: For arbitrary sets with cardinality at most d , their algorithm makes at most $dOPT + d$ queries, and for sets that represent minimal cuts in a tree with d terminal pairs, their algorithm makes at most $dOPT + 1$ queries. For the problem of computing a minimum-weight matroid base, a generalisation of the minimum spanning tree problem with edge uncertainty, they obtain a 2-competitive algorithm.

We refer to [5] for a more comprehensive survey of work on computing with explorable uncertainty.

3 Scheduling with Explorable Uncertainty

In this section we review some of the results for a scheduling problem with explorable uncertainty that were recently presented by Dürr et al. [3]. They consider the classical scheduling problem $1 \parallel \sum C_j$ of minimizing the sum of completion times for n given jobs without preemption on a single machine, but with a new twist: For each job J_j , initially only an upper limit \bar{p}_j on its processing time is known. If the job is not queried, its execution time is \bar{p}_j . If the job is queried, its execution time changes to some value p_j , $0 \leq p_j \leq \bar{p}_j$, that is revealed only as a result of the query. As motivation for this model one can consider, e.g., the case that jobs are file transmissions and querying a file corresponds to compressing it (where the amount of compression is not known before executing the compression). Another setting that motivates the model is that jobs can be executed in a safe mode and a faster, alternative mode, and an analysis of the job is necessary to determine whether the alternative mode can be used and what the resulting running-time will be.

It is assumed that executing a query takes one unit of time, and that the machine that executes the queries is the same as the machine that executes jobs. An algorithm must at each time decide whether to query a job, to execute a previously queried job, or to execute a job without querying (unqueried). This adds a novel flavour to the scheduling problem. Algorithms are analyzed using competitive analysis, comparing the objective value of the schedule produced

by the algorithm with the objective value of the optimal schedule. The optimal schedule also needs to query a job in order to be able to execute it with time p_j instead of \bar{p}_j , and it will do so only if $1 + p_j < \bar{p}_j$.

A 2-competitive deterministic algorithm can be obtained as follows: First, schedule all jobs with upper limit at most 2 unqueried in order of non-decreasing upper limit. Then, query all remaining jobs in arbitrary order. If querying a job J_j yields $p_j \leq 2$, the job is executed immediately. Otherwise, the job is deferred. At the end of the schedule, all deferred jobs are executed in order of non-decreasing p_j . Dürr et al. [3] also show that no deterministic algorithm can be better than 1.8546-competitive. The lower bound instance consists of n jobs with equal upper limit \bar{p} . For each of the first δn jobs that the algorithm queries or executes unqueried, the adversary sets p_j to be equal to \bar{p} if the algorithm queries the job and equal to 0 otherwise. Optimising the parameters δ and \bar{p} yields the lower bound.

The lower bound exploits the fact that the algorithm cannot distinguish between jobs with $p_j = 0$ and with $p_j = \bar{p}$ without querying them, and the adversary can ensure that jobs with $p_j = 0$ are queried last. A natural idea for a randomized algorithm is therefore to query jobs in random order. Dürr et al. [3] show that such an approach can be used to achieve competitive ratio 1.7453 against an oblivious adversary. They also give a lower bound of 1.6257 for randomized algorithms.

4 Future Directions

In the area of query-competitive algorithms for computing with explorable uncertainty discussed in Sect. 2, a limitation of witness set algorithms is that they cannot achieve a good competitive ratio for problems where it is not possible to determine small witness sets. More generally, there are a number of fundamental optimisation problems (e.g., the shortest-path problem) where one can construct instances where a single query suffices to solve the problem while any deterministic algorithm can be forced to make a large number of queries (e.g., $\Omega(n)$ queries for the shortest-path problem in a graph with n nodes and uncertain edge weights). It would be very interesting to model such problems in a way that limits the power of the adversary, so that algorithms with good competitive ratio become possible. For example, assuming that the uncertain weights are drawn from some probability distribution (but still allowing queries to reveal the exact weights) may be helpful [14].

Another interesting direction is the consideration of settings where a certain number of queries can be made in parallel and the goal is to minimize the number of query rounds that are needed to solve the problem. It is not straightforward to adapt witness set algorithms to this setting, as it is often necessary to query one witness set before another can be determined, and so it seems difficult to query several witness sets in parallel.

For the problem of scheduling with explorable uncertainty discussed in Sect. 3, an interesting question is whether there is a deterministic algorithm that

achieves competitive ratio strictly smaller than 2. For the special case where all jobs have equal upper limits, such algorithms are presented in [3]. The crucial ingredient seems to be balancing the time spent on querying and the time spent on executing jobs. It would also be interesting to study other scheduling problems in the model where jobs can become shorter after querying. More generally, studying other optimization problems with explorable uncertainty where the cost for the queries is part of the overall objective function could be a worthwhile direction [13].

References

1. Borodin, A., El-Yaniv, R.: *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge (1998)
2. Bruce, R., Hoffmann, M., Krizanc, D., Raman, R.: Efficient update strategies for geometric computing with uncertainty. *Theory Comput. Syst.* **38**(4), 411–423 (2005)
3. Dürr, C., Erlebach, T., Megow, N., Meißner, J.: Scheduling with explorable uncertainty. In: Karlin, A.R. (ed.) *ITCS 2018 9th Innovations in Theoretical Computer Science Conference*, 11–14 January 2018, Cambridge, MA, USA. LIPIcs, vol. 94, pp. 30:1–30:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). <https://doi.org/10.4230/LIPIcs.ITCS.2018.30>
4. Erlebach, T., Hoffmann, M.: Minimum spanning tree verification under uncertainty. In: Kratsch, D., Todinca, I. (eds.) *WG 2014*. LNCS, vol. 8747, pp. 164–175. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12340-0_14
5. Erlebach, T., Hoffmann, M.: Query-competitive algorithms for computing with uncertainty. *Bull. EATCS* **116**, 22–39 (2015)
6. Erlebach, T., Hoffmann, M., Kammer, F.: Query-competitive algorithms for cheapest set problems under uncertainty. *Theor. Comput. Sci.* **613**, 51–64 (2016)
7. Erlebach, T., Hoffmann, M., Krizanc, D., Mihalák, M., Raman, R.: Computing minimum spanning trees with uncertainty. In: *25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008)*. LIPIcs, vol. 1, pp. 277–288 (2008)
8. Kahan, S.: A model for data in motion. In: *23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pp. 267–277 (1991)
9. Levi, R.: Practice driven scheduling models. Talk at Dagstuhl Seminar 16081: Scheduling (2016)
10. Megow, N., Meißner, J., Skutella, M.: Randomization helps computing a minimum spanning tree under uncertainty. *SIAM J. Comput.* **46**(4), 1217–1240 (2017). <https://doi.org/10.1137/16M1088375>
11. Olston, C., Widom, J.: Offering a precision-performance tradeoff for aggregation queries over replicated data. In: *26th International Conference on Very Large Data Bases (VLDB 2000)*, pp. 144–155 (2000)
12. Shaposhnik, Y.: Exploration vs. Exploitation: reducing uncertainty in operational problems. Ph.D. thesis, Sloan School of Management, MIT (2016)
13. Singla, S.: The price of information in combinatorial optimization. In: Czumaj, A. (ed.) *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pp. 2523–2532 (2018)
14. Yamaguchi, Y., Maehara, T.: Stochastic packing integer programs with few queries. In: Czumaj, A. (ed.) *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pp. 293–310 (2018)



Diminishable Parameterized Problems and Strict Polynomial Kernelization

Henning Fernau¹, Till Fluschnik²(✉), Danny Hermelin³, Andreas Krebs⁴,
Hendrik Molter², and Rolf Niedermeier²

¹ Fachbereich 4 – Abteilung Informatik, Universität Trier, Trier, Germany
`fernau@uni-trier.de`

² Institut für Softwaretechnik und Theoretische Informatik,
TU Berlin, Berlin, Germany

`{till.fluschnik,h.molter,rolf.niedermeier}@tu-berlin.de`

³ Ben Gurion University of the Negev, Beersheba, Israel
`hermelin@bgu.ac.il`

⁴ Wilhelm-Schickard-Institut für Informatik, Universität Tübingen,
Tübingen, Germany
`krebs@informatik.uni-tuebingen.de`

Abstract. Kernelization—a mathematical key concept for provably effective polynomial-time preprocessing of NP-hard problems—plays a central role in parameterized complexity and has triggered an extensive line of research. In this paper we consider a restricted yet natural variant of kernelization, namely *strict kernelization*, where one is not allowed to increase the parameter of the reduced instance (the kernel) by more than an additive constant. Building on earlier work of Chen, Flum, and Müller [CiE 2009, Theory Comput. Syst. 2011], we underline the applicability of their framework by showing that a variety of fixed-parameter tractable problems, including graph problems and Turing machine computation problems, does not admit strict polynomial kernels under the weaker assumption of $P \neq NP$. Finally, we study a relaxation of the notion of strict kernels.

Work initiated by the research retreat of the Theoretical Computer Science group of the Universität of Tübingen in Sulz (Neckar), September 2016.

T. Fluschnik—Supported by the DFG, project DAMM (NI 369/13) and project TORE (NI 369/18).

D. Hermelin—Supported by the People Programme (Marie Curie Actions) of the European Union’s Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, and by the ISRAEL SCIENCE FOUNDATION (grant No. 551145/14). Also supported by a DFG Mercator fellowship, project DAMM (NI 369/13) while staying at TU Berlin (August 2016).

A. Krebs—Supported by the DFG Emmy Noether program (KR 4042/2).

H. Molter—Partially supported by the DFG, project DAPA (NI 369/12).

1 Introduction

Kernelization is one of the most fundamental concepts in the field of parameterized complexity analysis. Given an instance $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ of some parameterized problem L (we assume the parameter to be encoded in unary), a *kernelization* for L produces in polynomial time an instance (x', k') satisfying: $(x', k') \in L \iff (x, k) \in L$ and $|x'| + k' \leq f(k)$ for some fixed computable function $f(k)$. In this way, kernelization can be thought of as a preprocessing procedure that reduces an instance to its “computationally hard core” (*i.e.*, the *kernel*). The function $f(k)$ is accordingly called the *size* of the kernel, and it is typically the measure that one wishes to minimize. Kernelization is a central concept in parameterized complexity not only as an important algorithmic tool, but also because it provides an alternative definition of *fixed-parameter tractability* (FPT) [6]. An algorithm with running time $f(k) \cdot |x|^{O(1)}$ for a parameterized problem L implies that L has a kernel of size $f(k)$, but in the converse direction one cannot always take the same function f . The goal of minimizing the size of the problem kernel leads to the question of what is the kernel with the smallest size one can obtain in polynomial time for a given problem. In particular, do all fixed-parameter tractable problems have small kernels, say, of linear or polynomial size?

The latter question was answered negatively [5, 14] by proving that various problems in FPT do not admit a polynomial-size kernel (*polynomial kernel* for short) under the assumption that $\text{NP} \not\subseteq \text{coNP/poly}$. The framework has been extended in several directions [18]. Regardless, all of these frameworks rely on the assumption that $\text{NP} \not\subseteq \text{coNP/poly}$, an assumption that, while widely believed in the community, is a much stronger assumption than $\text{P} \neq \text{NP}$.

Throughout the years, researchers have considered different variants of kernelization such as *fidelity-preserving preprocessing* [11], *partial kernelization* [3], *lossy kernelization* [20], and *Turing kernelization* [4, 21]. In this paper, we consider a variant which has been considered previously quite a bit, which is called *proper kernelization* [1] or *strict kernelization* [8]:

Definition 1 (Strict Kernel). *A strict kernelization for a parameterized problem L is a polynomial-time algorithm that on input $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ outputs $(x', k') \in \{0, 1\}^* \times \mathbb{N}$, the strict kernel, satisfying: (i) $(x, k) \in L \iff (x', k') \in L$, (ii) $|x'| \leq f(k)$, for some function f , and (iii) $k' \leq k + c$, for some constant c . We say that L admits a strict polynomial kernelization if $f(k) \in k^{O(1)}$.*

Thus, a strict kernelization is a kernelization that does not increase the output parameter k' by more than an additive constant. While the term “strict” in the definition above makes sense mathematically, it is actually quite harsh from a practical perspective. Indeed, most of the early work on kernelization involved applying so-called *data reduction rules* that rarely ever increase the parameter value (see *e.g.* the surveys [15, 18]). Furthermore, strict kernelization is clearly preferable to kernelizations that increase the parameter value in a dramatic way:

Often a fixed-parameter algorithm on the resulting problem kernel is applied, whose running time highly depends on the value of the parameter, and so a kernelization that substantially increases the parameter value might in fact be useless. Finally, the equivalence with FPT is preserved: A parameterized problem is solvable in $f(k) \cdot |x|^{O(1)}$ time if and only if it has a strict kernel of size $g(k)$ (where f and g are some computable functions).

Chen *et al.* [8] showed that ROOTED PATH, the problem of finding a simple path of length k in a graph that starts from a prespecified root vertex, parameterized by k has no strict polynomial kernel unless $P = NP$. They also showed a similar result for CNF-SAT parameterized by the number of variables. Both of these results seemingly are the only known polynomial kernel lower bounds that rely on the assumption of $P \neq NP$ (see Chen *et al.* [7] for a few linear lower bounds that also rely on $P \neq NP$). The goal of our work is to show that Chen *et al.*'s framework applies for more problems and is easy to extend.

Our Results. We build on the work of Chen *et al.* [8], and further develop and widen the framework they presented for excluding strict polynomial kernels. Using this extended framework, we show that several natural parameterized problems in FPT have no strict polynomial kernels under the assumption that $P \neq NP$. The main result of our work is given in Theorem 2 below. Note that we use the brackets in the problem names to denote the parameter under consideration.¹

Theorem 2. *Unless $P = NP$, each of the following fixed-parameter tractable problems does not admit a strict polynomial kernel:*

- MULTICOLORED PATH(k) and MULTICOLORED PATH($k \log n$);
- CLIQUE(Δ), CLIQUE(tw), CLIQUE(bw), and CLIQUE(cw);
- BICLIQUE(Δ), BICLIQUE(tw), BICLIQUE(bw), and BICLIQUE(cw);
- COLORFUL GRAPH MOTIF(k) and TERMINAL STEINER TREE($k + |T|$);
- MULTI-COMPONENT ANNOTATED DEFENSIVE ALLIANCE(k) and MULTI-COMPONENT ANNOTATED VERTEX COVER(k);
- SHORT NTM COMPUTATION($k + |\Sigma|$), SHORT NTM COMPUTATION($k + |Q|$), and SHORT BINARY NTM COMPUTATION(k).

(Herein, k denotes the solution size, n , Δ , tw , bw , and cw denote the number of vertices, the maximum vertex degree, the treewidth, bandwidth, and cutwidth of the graph, respectively, T denotes the set of terminals, $|\Sigma|$ denotes the alphabet size, and $|Q|$ denotes the number of states.)

Finally, we also explore how “tight” the concept of strict polynomial kernels is. We modify the framework for “less” strict kernels, and, employing the Exponential Time Hypothesis (ETH), conclude that we often cannot hope for significantly relaxing the concept of strict kernelization to achieve a comparable list of such analogous kernel lower bounds under $P \neq NP$. Notably, our

¹ For a complete list of problem definitions we refer to a long version [12] of the paper.

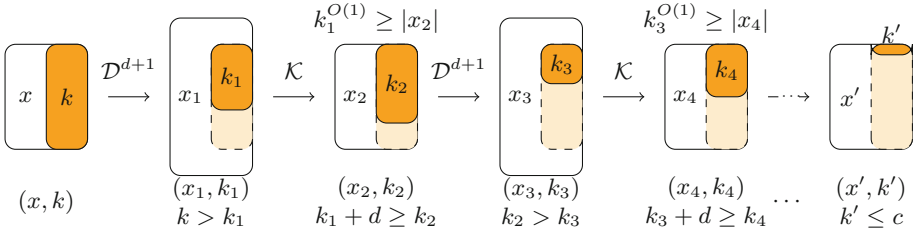


Fig. 1. Illustration to the proof of Theorem 4 for an input instance (x, k) . Herein, \mathcal{K} denotes the strict kernelization with additive constant d and \mathcal{D} denotes the parameter diminisher. We represent each instance by boxes: the size of a box symbolizes the size of the instance or the value of the parameter (each dashed box refers to k).

modified framework herein was recently applied [13] for proving the first direct kernelization lower bounds for polynomial-time solvable problems.

We remark that for the problems we discuss in this paper, one can exclude polynomial kernels under the assumption that $\text{NP} \not\subseteq \text{coNP/poly}$ using the existing frameworks [5, 18]. In contrast, our results base on a weaker assumption, but exclude a more restricted version of polynomial kernels. Hence, our results are incomparable with the existing no-polynomial-kernel results.

Notation. We use basic notation from parameterized complexity [10] and graph theory [9]. Let $G = (V, E)$ be an undirected graph. For $W \subseteq V$, let $G - W := (V \setminus W, \{e \in E \mid e \cap W = \emptyset\})$. If $W = \{v\}$, then we write $G - v$. For $v \in V$, we denote by $N_G(v) := \{w \in V \mid \{v, w\} \in E\}$ the neighborhood of v in G . We denote by $[\ell]$, $\ell \in \mathbb{N}$, the set $\{1, \dots, \ell\}$ and by \log the logarithm with base two.

2 Framework

In this section we present the general framework used throughout the paper. Firstly, we define the central notion of a *parameter diminisher* referring to *parameter decreasing polynomial reduction* introduced by Chen *et al.* [8].

Definition 3 (Parameter Diminisher). A parameter diminisher for a parameterized problem L is a polynomial-time algorithm that maps instances $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ to instances $(x', k') \in \{0, 1\}^* \times \mathbb{N}$ such that (i) $(x, k) \in L$ if and only if $(x', k') \in L$ and (ii) $k' < k$.

We call a parameterized problem L *diminishable* if there is a parameter diminisher for L . The following theorem was proved initially by Chen *et al.* [8], albeit for slightly weaker forms of diminisher and strict polynomial kernels.

Theorem 4 ([8]). Let L be a parameterized problem such that its unparameterized version is NP-hard and $\{(x, k) \in L \mid k \leq c\} \in \text{P}$, for some constant c . If L is diminishable and admits a strict polynomial kernel, then $\text{P} = \text{NP}$.

The idea behind Theorem 4 is to repeat the following two procedures until the parameter value drops below c (see Fig. 1 for an illustration). First, apply the parameter diminisher a constant number of times such that when, second, the strict polynomial kernelization is applied, the parameter value is decreased. The strict polynomial kernelization keeps the instances small, hence the whole process runs in polynomial time.

Reductions transfer diminishability from one parameterized problem to another if they do not increase the parameter value and run in polynomial time. Formally, given two parameterized problems L with parameter k and L' with parameter k' , a *parameter-non-increasing reduction* from L to L' is an algorithm that maps each instance (x, k) of L to an equivalent instance (x', k') of L' in $\text{poly}(|x| + k)$ time such that $k' \leq k$. Note that to transfer diminishability, we need parameter-non-increasing reductions between two parameterized problems in both directions—a crucial difference to other reduction-based hardness results.

Lemma 5 (\star^2). *Let L_1 and L_2 be two parameterized problems such that there are parameter-non-increasing reductions from L_1 to L_2 and from L_2 to L_1 . Then we have that L_1 is diminishable if and only if L_2 is diminishable.*

Parameter-Decreasing Branching and Strict Composition. To construct parameter diminishers, it is useful to follow a “branch and compose” technique: Herein, first *branch* into several subinstances while decreasing the parameter value in each, and then *compose* the subinstances into one instance without increasing the parameter value by more than an additive constant. We first give the definitions and then show that both combined form a parameter diminisher.

A *parameter-decreasing branching rule* for a parameterized problem L is a polynomial-time algorithm that on input $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ outputs a sequence of instances $(y_1, k'), \dots, (y_t, k') \in \{0, 1\}^* \times \mathbb{N}$ such that $(x, k) \in L \iff (y_i, k') \in L$ for at least one $i \in [t]$ and $k' < k$. Composition is the core concept behind the standard kernelization lower bound framework introduced by Bodlaender *et al.* [5]. Here we use a more restrictive notion of this concept: A *strict composition* for a parameterized problem L is an algorithm that receives as input t instances $(x_1, k), \dots, (x_t, k) \in \{0, 1\}^* \times \mathbb{N}$, and outputs in polynomial time a single instance $(y, k') \in \{0, 1\}^* \times \mathbb{N}$ such that (i) $(y, k') \in L \iff (x_i, k) \in L$ for some $i \in [t]$ and (ii) $k' \leq k + c$ for some constant c . If we now combine (multiple applications of) a parameter-decreasing branching rule with a strict composition, then we get a parameter diminisher. We remark that this also holds if we require in both definitions that the equivalence holds for all $i \in [t]$.

Lemma 6 (\star). *Let L be a parameterized problem. If L admits a parameter-decreasing branching rule and a strict composition, then it is diminishable.*

² Full proofs of results marked with (\star) are deferred to a long version [12] of the paper.

On the Exclusion of Non-Uniform Kernelization Algorithms. The presented framework can be easily adapted to exclude different forms of strict kernelizations. As our example, we show that the framework can be used to exclude strict polynomial kernels computed in *non-uniform* polynomial time (the corresponding complexity class is called P/poly) under the assumption that $\text{NP} \not\subseteq \text{P/poly}$. A *non-uniform strict kernelization* for a parameterized problem L is a *non-uniform* polynomial-time algorithm that on input $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ outputs $(x', k') \in \{0, 1\}^* \times \mathbb{N}$, the *strict kernel*, satisfying: (i) $(x, k) \in L \iff (x', k') \in L$, (ii) $|x'| \leq f(k)$, for some function f , and (iii) $k' \leq k + c$, for some constant c . We say that L admits a non-uniform strict *polynomial* kernelization if $f(k) \in k^{O(1)}$.

Proposition 7 (\star). *Let L be a parameterized problem such that its unparameterized version is NP-hard and we have that $\{(x, k) \in L \mid k \leq c\} \in \text{P/poly}$, for some constant c . If L is diminishable and admits a non-uniform strict polynomial kernel, then $\text{NP} \subseteq \text{P/poly}$.*

We remark that if $\text{NP} \subseteq \text{P/poly}$, then the Polynomial Hierarchy collapses to its second level [17] (note that $\text{NP} \subseteq \text{coNP/poly}$ implies a collapse in the Polynomial Hierarchy to its third level).

3 Problems Without Strict Polynomial Kernels

In this section, we exemplify the proof of Theorem 2 on two selected graph problems. The complete proof of Theorem 2 can be found in a long version [12].

First, we present a diminisher for the MULTICOLORED PATH(k) problem: Given an undirected graph $G = (V, E)$ with a vertex coloring function $\text{col} : V \rightarrow [k]$, determine whether there exists a simple path of length k containing one vertex of each color. This problem is NP-complete as it generalizes HAMILTONIAN PATH. Furthermore, MULTICOLORED PATH(k) is fixed-parameter tractable as it can be solved in $2^{O(k)}n^2$ time [2]. The idea used in the parameter diminisher for MULTICOLORED PATH(k) can also be applied for the COLORFUL GRAPH MOTIF problem, a problem with applications in bioinformatics.

Proposition 8. MULTICOLORED PATH(k) is diminishable.

For graph problems, a vertex-coloring seems to help to construct diminishers. As an example, the diminishability of the (uncolored) PATH(k) problem, asking whether a given graph contains a simple path of length k , remains open.

Proof. We give a parameter-decreasing branching rule and a strict composition for MULTICOLORED PATH(k). The result then follows from Lemma 6. Let $(G = (V, E), \text{col})$ be an instance of MULTICOLORED PATH(k). Our parameter-decreasing branching rule for $(G = (V, E), \text{col})$ creates a graph $G_{(v_1, v_2, v_3)}$ for each ordered triplet (v_1, v_2, v_3) of vertices of V such that v_1, v_2, v_3 is a multicolored path in G . The graph $G_{(v_1, v_2, v_3)}$ is constructed from G as follows: Delete from G all vertices $w \in V \setminus \{v_2, v_3\}$ with $\text{col}(w) \in \{\text{col}(v_1), \text{col}(v_2), \text{col}(v_3)\}$.

Following this, only vertices of $k - 1$ colors remain, and v_2 and v_3 are the only vertices colored $\text{col}(v_2)$ and $\text{col}(v_3)$, respectively. Then delete all edges incident with v_2 , apart from $\{v_2, v_3\}$, and relabel all colors so that the image of col for $G_{(v_1, v_2, v_3)}$ is $[k - 1]$.

Clearly our parameter-decreasing branching rule can be performed in polynomial time. Furthermore, the parameter decreases in each output instance. We show that the first requirement holds as well: Indeed, suppose that G has a multicolored path v_1, v_2, \dots, v_k of length k . Then v_2, \dots, v_k is a multicolored path of length $k - 1$ in $G_{(v_1, v_2, v_3)}$ by construction. Conversely, suppose that there is a multicolored path u_2, \dots, u_k of length $k - 1$ in some $G_{(v_1, v_2, v_3)}$. Then since v_2 is the only vertex of color $\text{col}(v_2)$ in $G_{(v_1, v_2, v_3)}$, and since v_2 is only adjacent to v_3 , it must be without loss of generality that $u_2 = v_2$ and $u_3 = v_3$. Hence, since v_1 is adjacent to v_2 in G , and no vertices of u_2, \dots, u_k have color $\text{col}(v_1)$ in G , the sequence of v_1, u_2, \dots, u_k forms a multicolored path of length k in G .

Our strict composition for MULTICOLORED PATH(k) is as follows. Given a sequence of inputs $(G_1, \text{col}_1), \dots, (G_t, \text{col}_t)$, the strict composition constructs the disjoint union G and the coloring col of all graphs G_i and coloring functions col_i , $1 \leq i \leq t$. Clearly, (G, col) contains a multicolored path of length k if and only if there is a multicolored path of length k in some (G_i, col_i) . The result thus follows directly from Lemma 6. \square

Proposition 9 (\star). *Unless $P = NP$, MULTICOLORED PATH($k \log n$) has no strict polynomial kernel.*

We next consider the NP-complete TERMINAL STEINER TREE (TST) [19] problem: given an undirected graph $G = (V = N \uplus T, E)$ (T is called the terminal set) and a positive integer k , decide whether there is a subgraph $H \subseteq G$ with at most $k + |T|$ vertices such that H is a tree and T is a subset of the set of leaves of H . TST forms a variant of the well-known STEINER TREE problem. When parameterized by $k + |T|$, TST is fixed-parameter tractable (see long version [12]).

Proposition 10 (\star). *TERMINAL STEINER TREE($k + |T|$) is diminishable.*

Proof (Diminisher Construction). We present a parameter-decreasing branching rule and a strict composition for TERMINAL STEINER TREE($k + |T|$). Together with Lemma 6, the claim then follows. Let $(G = (N \uplus T, E), k)$ be an instance of TST($k + |T|$) (we can assume that G has a connected component containing T). We make several assumptions first. We can assume that $|T| \geq 3$ (otherwise a shortest path is the optimal solution) and additionally that for all terminals $t \in T$ it holds that $N_G(t) \not\subseteq T$ (as otherwise the instance is a no-instance). Moreover, we can assume that there is no vertex $v \in N$ such that $T \subseteq N_G(v)$, as otherwise we immediately output whether $k \geq 1$.

For the parameter decreasing branching rule, select a terminal $t^* \in T$, and let v_1, \dots, v_d denote the neighbors of t^* in $G - (T \setminus \{t^*\})$. We create d instances $(G_1, k - 1), \dots, (G_d, k - 1)$ as follows. Define G_i , $i \in [d]$, by $G_i := G - v_i$. Turn the vertices in $N_G(v_i)$ in G_i into a clique, that is, for each distinct vertices $v, w \in$

$N_G(v_i)$ add the edge $\{v, w\}$ if not yet present. This finishes the construction of G_i . It is not hard to see that the construction can be done in polynomial time.

Next, we describe the strict composition for $\text{TST}(k+|T|)$. Given the instances $(G_1, k), \dots, (G_d, k)$, we create an instance (G', k) as follows. Let G' be initially the disjoint union of G_1, \dots, G_d . For each $t \in T$, identify its copies in G_1, \dots, G_d , say t_1, \dots, t_d , with one vertex t' corresponding to t . This finishes the construction of G' . Note that for every $i, j \in [d]$, $i \neq j$, any path between a vertex in G_i and a vertex in G_j contains a terminal vertex. Hence, any terminal Steiner tree in G' contains non-terminal vertices only in G_i for exactly one $i \in [d]$. It is not difficult to see that (G', k) is a yes-instance if and only if one of the instances $(G_1, k), \dots, (G_d, k)$ is a yes-instance. \square

4 Problems Without Semi-strict Polynomial Kernels

As strict kernels only allow an increase of the parameter value by an additive constant (Definition 1), one may ask whether one can exclude less restrictive versions of strict kernels for parameterized problems using the concept of parameter diminishers. Targeting this question, in this section we study scenarios with a multiplicative (instead of additive) parameter increase by a constant. That is, property (iii) in Definition 1 is replaced by $k' \leq c \cdot k$, for some constant c . We refer to this as *semi-strict kernels*.

Note that Theorem 4 does not imply that the problems mentioned in Theorem 2 do not admit semi-strict polynomial kernelizations unless $\text{P} = \text{NP}$. Intuitively, the parameter diminisher is constantly often applied to decrease the parameter, while dealing only with a constant additive blow-up of the parameter caused by the strict kernelization. When dealing with a constant multiplicative blow-up of the parameter caused by the semi-strict kernelization, the parameter diminisher is required to be applied a non-constant number of times. Hence, to deal with semi-strict kernelization, we introduce a stronger version of our parameter diminisher: Formally, we replace property (ii) in Definition 3 by $k' \leq k/c$, for some constant $c > 1$. We refer to this as *strong parameter diminishers*.

Next, we show an analogue of Theorem 4 for semi-strict polynomial kernelizations and strong parameter diminishers.

Theorem 11 (\star). *Let L be a parameterized problem such that its unparameterized version is NP-hard and $\{(x, k) \in L \mid k \leq c\} \in \text{P}$, for some constant $c \geq 1$. If L is strongly diminishable and admits a semi-strict polynomial kernel, then $\text{P} = \text{NP}$.*

By Theorem 11, if we can prove a strong diminisher for a parameterized problem, then it does not admit a semi-strict polynomial kernel, unless $\text{P} = \text{NP}$. We give a strong diminisher for the SET COVER problem: Given a set U called the universe, a family $\mathcal{F} \subseteq 2^U$ of subsets of U , and an integer k , the question is whether there are k sets in the family \mathcal{F} that cover the whole universe. We show that SET COVER parameterized by $k \log n$, where $n = |U|$, is strongly diminishable.

Theorem 12 (★). *Unless $P = NP$, SET COVER($k \log n$) and HITTING SET($k \log m$) do not admit a semi-strict polynomial kernel.*

Proof (Strong Diminisher Construction). Let $(U, \mathcal{F} = \{F_1, \dots, F_m\}, k)$ be an instance of SET COVER($k \log n$) and assume that $k \geq 2$ and $n \geq 5$. If k is odd, then we add a unique element to U , a unique set containing only this element to \mathcal{F} , and we set $k = k + 1$. Hence, we assume that k is even. The following procedure is a strong parameter diminisher for the problem parameterized by $k \log n$. Let $U' = U$ and for all F_i, F_j create $F'_{\{i,j\}} = F_i \cup F_j$. Let $\mathcal{F}' = \{F'_{\{i,j\}} \mid i \neq j\}$ and set $k' = k/2$. This yields the instance (U', \mathcal{F}', k') of SET COVER($k \log n$) in polynomial time. The proof of correctness is deferred to a long version [12]. \square

Seeking for parameter diminishers to exclude strict polynomial kernelizations raises the question whether there are parameterized problems that are not (strongly) diminishable. In the following, we prove that under the *Exponential Time Hypothesis*, or ETH for short [16], there are natural problems that do not admit strong parameter diminishers. Here we restrict ourselves to problems where we have a parameter diminisher. The Exponential Time Hypothesis states that there is no algorithm for 3-CNF-SAT running in $2^{o(n)}$ poly($n + m$) time, where n and m denote the number of variables and clauses, respectively.

Theorem 13 (★). *Assuming ETH, none of the following is strongly diminishable: CNF-SAT(n), ROOTED PATH(k), CLIQUE(Δ), CLIQUE(tw), and CLIQUE(bw).*

The following lemma is the key tool for excluding strong parameter diminishers under ETH. Roughly, it can be understood as saying that a strong parameter diminisher can improve the running time of existing algorithms.

Lemma 14 (★). *Let L be a parameterized problem. If there is an algorithm \mathcal{A} that solves any instance $(x, k) \in L$ in $2^{O(k)} \cdot |x|^{O(1)}$ time and L is strongly diminishable, then there is an algorithm \mathcal{B} that solves L in $2^{O(k/f(x,k))} \cdot |x|^{f(x,k)^{O(1)}}$ time, where $f : L \rightarrow \mathbb{N}$ is a function mapping instances of L to the natural numbers with the following property: For every constant c there is a natural number n such that for all instances $(x, k) \in L$ we have that $|x| \geq n$ implies that $f(x, k) \geq c$.*

Intuitively, we apply Lemma 14 to exclude the existence of strong parameter diminishers under ETH as follows. Consider a problem where we know a running time lower bound based on the ETH and we also know an algorithm that matches this lower bound. Then, due to Lemma 14, for many problems a strong parameter diminisher and a suitable choice for the function f would imply the existence of an algorithm whose running time breaks the lower bound.

5 Conclusion

We showed that for several natural problems a strict polynomial-size problem kernel is as likely as $P = NP$. Since basically all observed (natural and practically

relevant) polynomial kernels are strict, this reveals that the existence of valuable kernels may be tighter connected to the P vs. NP problem than previously expected (in almost all previous work a connection is drawn to a collapse of the polynomial hierarchy to its third level, and the conceptual framework used there seems more technical than the one used here). Our work is based on results of Chen *et al.* [8] and shows that their basic ideas can be extended to a larger class of problems than dealt with in their work.

The diminisher framework leaves several challenges for future work. Are there natural problems where the presented framework is able to refute strict polynomial kernels while the composition framework [5] is not? It is not clear whether a framework based on a weaker assumption is even able to produce results that a framework based on a stronger assumption is not able to produce. This possibly also ties in with the question whether there are “natural” parameterized problems that admit a polynomial kernel but no strict polynomial kernel.³ We close with two concrete open problems:

- We proved that $\text{MULTICOLORED PATH}(k)$ is diminishable (and thus refutes a strict polynomial kernel unless $P = NP$). Can this result be extended to the uncolored version of the problem? This is also open for the directed case.
- $\text{CLIQUE}(\Delta)$, $\text{CLIQUE}(\text{tw})$, $\text{CLIQUE}(\text{bw})$ do not have strong diminishers under the ETH (Sect. 4). Is this also true for $\text{CLIQUE}(\text{cw})$?

References

1. Abu-Khzam, F.N., Fernau, H.: Kernels: annotated, proper and induced. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 264–275. Springer, Heidelberg (2006). https://doi.org/10.1007/11847250_24
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. *J. ACM* **42**(4), 844–856 (1995)
3. Betzler, N., Guo, J., Komusiewicz, C., Niedermeier, R.: Average parameterization and partial kernelization for computing medians. *J. Comput. Syst. Sci.* **77**(4), 774–789 (2011)
4. Binkele-Raible, D., Fernau, H., Fomin, F.V., Lokshtanov, D., Saurabh, S., Villanger, Y.: Kernel(s) for problems with no kernel: on out-trees with many leaves. *ACM Trans. Algorithms* **8**(4), 38 (2012)
5. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *J. Comput. Syst. Sci.* **75**(8), 423–434 (2009)
6. Cai, L., Chen, J., Downey, R.G., Fellows, M.R.: Advice classes of parameterized tractability. *Ann. Pure Appl. Logic* **84**(1), 119–138 (1997)
7. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: lower bounds and upper bounds on kernel size. *SIAM J. Comput.* **37**(4), 1077–1106 (2007)
8. Chen, Y., Flum, J., Müller, M.: Lower bounds for kernelizations and other preprocessing procedures. *Theory Comput. Syst.* **48**(4), 803–839 (2011)
9. Diestel, R.: *Graph Theory*, 4th edn. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-662-53622-3>

³ Note that Chen *et al.* [8, Proposition 3.3] presented an artificial parameterized problem admitting a polynomial kernel but no strict polynomial kernel.

10. Downey, R.G., Fellows, M.R.: *Fundamentals of Parameterized Complexity*. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-1-4471-5559-1>
11. Fellows, M.R., Kulik, A., Rosamond, F.A., Shachnai, H.: Parameterized approximation via fidelity preserving transformations. *J. Comput. Syst. Sci.* **93**, 30–40 (2018)
12. Fernau, H., Fluschnik, T., Hermelin, D., Krebs, A., Molter, H., Niedermeier, R.: Diminishable parameterized problems and strict polynomial kernelization. *CoRR abs/1611.03739* (2018). <http://arxiv.org/abs/1611.03739>
13. Fluschnik, T., Mertzios, G.B., Nichterlein, A.: Kernelization lower bounds for finding constant size subgraphs. *CoRR abs/1710.07601* (2017). <http://arxiv.org/abs/1710.07601>
14. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.* **77**(1), 91–106 (2011)
15. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. *ACM SIGACT News* **38**(1), 31–45 (2007)
16. Impagliazzo, R., Paturi, R.: On the complexity of k -SAT. *J. Comput. Syst. Sci.* **62**(2), 367–375 (2001)
17. Karp, R.M., Lipton, R.: Turing machines that take advice. *L'Enseignement Mathématique* **28**(2), 191–209 (1982)
18. Kratsch, S.: Recent developments in kernelization: a survey. In: *Bulletin of the EATCS*, no. 113 (2014)
19. Lin, G., Xue, G.: On the terminal Steiner tree problem. *Inf. Process. Lett.* **84**(2), 103–107 (2002)
20. Lokshantov, D., Panolan, F., Ramanujan, M.S., Saurabh, S.: Lossy kernelization. In: *Proceedings of 49th STOC*, pp. 224–237. ACM (2017)
21. Schäfer, A., Komusiewicz, C., Moser, H., Niedermeier, R.: Parameterized computational complexity of finding small-diameter subgraphs. *Optim. Lett.* **6**(5), 883–891 (2012)



New Nonterminal Complexity Results for Semi-conditional Grammars

Henning Fernau^{1(✉)}, Lakshmanan Kuppusamy², and Rufus O. Oladele³

¹ CIRT, Fachbereich 4, Universität Trier, 54286 Trier, Germany
fernau@uni-trier.de

² School of Computer Science and Engineering, VIT, Vellore 632 014, India

³ Department of Computer Science, University of Ilorin, P.M.B.1515, Ilorin, Nigeria

Abstract. A semi-conditional grammar is a form of regulated rewriting system. Each rule consists of a context-free core rule $A \rightarrow w$ and two strings w_+, w_- ; the rule is applicable if w_+ (the positive condition) occurs as a substring of the current sentential form, but w_- (the negative condition) does not. The maximum lengths i, j of the positive or negative conditional strings, respectively, give a natural measure of descriptiveness, known as the *degree* of such grammars. Employing several normal form results on phrase-structure grammars as derived by Geffert, we improve on previously obtained results by reducing the number of nonterminals of semi-conditional grammars of a given degree (i, j) while maintaining computational completeness of the said mechanisms.

1 Introduction

One of the key areas in modern formal language theory is descriptiveness. In a nutshell, this area investigates how succinct can a grammatical device, automaton, or rewriting system represent a formal language class (with respect to a certain complexity measure). In this paper, we focus on the nonterminal complexity of semi-conditional grammars of a certain degree so that they still achieve computational completeness, i.e., they still characterize the class RE of recursively enumerable languages. This includes questions like: Does there exist a semi-conditional grammar G_L for any RE language L such that G_L has degree $(2, 1)$ and uses only six nonterminals? We will answer this question affirmatively in this paper, while it is left as an open question if five nonterminals suffice.

To arrive at such results, it is often helpful to make use of other similar results and to simulate the corresponding devices. Among the most often used ones are the normal forms for type-0 grammars provided by Geffert in [3], because they also use very few nonterminals to characterize RE.

To underline the importance and history of nonterminal complexity in the context of regulated rewriting, we refer to [1, 2, 9]. In the field of semi-conditional grammars, we would like to point to [4–6, 8, 9]. A further descriptiveness measure more special to semi-conditional grammars is that of the *degree* which limits the length of the conditional strings that can be tested. It is known that

Table 1. Survey on nonterminal complexity results for SCG grammars

Degree (i, j)	# Nonterminals	# Conditional rules	Reference
(2,1)	8	7	[6]
(2,1)	7	6	[8]
(2,1)	6	$7 + P_{cf} $	Theorem 1
(2,2)	6	6	Theorem 2
(3,1)	5	$9 + 2(T + 1) + P_{cf} $	Theorem 3
(3,1)	6	13	Theorem 4

even semi-conditional grammars of degree $(1, 1)$ are computationally complete, i.e., they characterize the family RE of recursively enumerable languages; see [5]. Nothing is known on the nonterminal complexity in that case, though.

For semi-conditional grammars of degree $(2, 1)$, we improve on the previous result (due to [8]) of seven nonterminals, bringing it down to six. For semi-conditional grammars of degree $(3, 1)$, also nothing better than the sufficiency of seven nonterminals is known. We reduce this bound to five by a non-trivial use of another normal form result of Geffert. We survey the previously known and new results in Table 1, also including a discussion of the number of conditional rules. In Table 1, T refers to the terminal alphabet and P_{cf} to a subset of the rules of the given type-0 grammar; see Footnote 1. Notice that if we neglect the number of conditional rules as a descriptonal complexity aspects, our results improve on all previously published ones, sometimes trading off degree parameters and number of nonterminals.

In order to show our results, we make use of several normal forms for type-0 grammars that were presented in [3]. This is somehow special, as previously (mostly) the normal form with non-context-free erasing rules $AB \rightarrow \lambda$ and $CD \rightarrow \lambda$ was used. We also derive several useful properties of the said normal forms and we use them later while proving our results.

2 Preliminaries and Definitions

It is assumed in this paper that the reader is familiar with the fundamentals of language theory and mathematics in general. Let \mathbb{N} denote the set of non-negative integers. Let Σ^* denote the free monoid generated by a finite set Σ called alphabet under an operation termed concatenation, where λ denotes the unit of Σ^* , also called the empty string. Any element of Σ^* is called a word or string (over Σ). Any subset of Σ^* is called a language. A word v is a subword of $x \in \Sigma^*$ if there are words u, w such that $x = uvw$. Let $sub(x) \subseteq \Sigma^*$ denote the set of all subwords of $x \in \Sigma^*$. Clearly, $sub(x)$ is a finite language.

2.1 Semi-conditional Grammars

A *semi-conditional grammar* is a quadruple $G = (V, T, P, S)$, where V is the total alphabet, $T \subset V$ is the terminal alphabet, $S \in V \setminus T$ is the starting symbol, P is a finite set of productions of the form $(A \rightarrow x, \alpha, \beta)$ with $A \in V \setminus T$, $x \in V^*$, $\alpha, \beta \in V^+ \cup \{0\}$, where $0 \notin V$ is a special symbol, intuitively meaning that the condition is missing.

The production $l : (A \rightarrow x, \alpha, \beta) \in P$ (with label l) is said to be *conditional* if $\alpha \neq 0$ or $\beta \neq 0$. String α is permitting, while β is forbidden, as formally explained now. The production labeled l can be applied on a string $u \in V^*(V \setminus T)V^*$ if and only if $A \in \text{sub}(u)$ and $(\alpha \in \text{sub}(u)$ or $\alpha = 0)$ and $(\beta \notin \text{sub}(u)$ or $\beta = 0)$. Under these conditions, $u = u_1Au_2$ will be transformed into $v = u_1xu_2$, which is denoted by $u \Rightarrow_l v$.

When no confusion exists on the rule being applied we avoid mentioning the label and we simply write $u \Rightarrow v$. The language of G is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$, where \Rightarrow^* denotes the reflexive and transitive closure of \Rightarrow .

G is said to be of *degree* (i, j) , where $i, j \in \mathbb{N}$, if in every rule $(A \rightarrow x, \alpha, \beta)$ of P we have $|\alpha| \leq i$ and $|\beta| \leq j$ whenever $\alpha, \beta \neq 0$. For brevity, we write $\text{SCP}(i, j)$ to refer to semi-conditional grammars of degree (i, j) .

2.2 Geffert Normal Forms

In [3], quite a number of normal forms for type-0 grammars have been derived. They all differ by the number of nonterminals that are used and also by the number of non-context-free rules. We will hence speak of (n, r) -GNF to refer to a Geffert normal form with n nonterminals and r non-context-free rules. However, all these normal forms characterize the class RE.

The best known such normal form is the $(5, 2)$ -GNF with nonterminals S (the start symbol) and A, B, C, D that uses context-free rules with S as its left-hand side in phase one; in a second phase, non-context-free erasing rules $AB \rightarrow \lambda$ and $CD \rightarrow \lambda$ are applied to finally derive a terminal string.

In [3], it was also proved that every RE language is generated by a type-0 grammar $G = (N, T, P, S)$ with three nonterminals only, i.e., $N = \{S, A, B\}$ is the set of non-terminals, T is the set of terminal symbols, S is the start symbol and P contains the context-free productions of the form (i) $S \rightarrow w$ and the only non-context-free rule of the form (ii) $ABBBA \rightarrow \lambda$. ($(3, 1)$ -GNF)

We need the following *properties* of $(3, 1)$ -GNF that follow from its construction from the better-known $(5, 2)$ -GNF with nonterminals S, A, B, C, D ; the rules of this normal form are transformed by applying the morphism $A \mapsto AB, B \mapsto BBA, C \mapsto ABB$ and $D \mapsto BA$ (otherwise behaving like the identity) to obtain a grammar in $(3, 1)$ -GNF.

1. If $S \Rightarrow^* w$, then $w \in \{AB, ABB\}^* \{S, \lambda\} \{BA, BBA\}^* T^*$.
2. No sentential form w that is derivable in G contains a substring $BBBBB$.
3. If $S \Rightarrow^* w$, then w contains at most one of the three substrings $ABBBA$ or $ABBA$ or $ABBBBA$, called the *central part*. In the latter two cases, the derivation is stuck and the former case will only yield a fruitful derivation.

4. If $S \Rightarrow^* w$ and w contains no substring BBB , then either $w \in T^*$ or there is no way to terminate this derivation.
5. A derivation $S \Rightarrow^* x$ with $x \in T^*$ works in two phases. Phase one actually splits in two stages. In the first stage, rules of the form $S \rightarrow uSa$ are applied, with $u \in \{AB, ABB\}^*$ and $a \in T$, while in the second stage, rules of the form $S \rightarrow uSv$ are applied¹, where $u \in \{AB, ABB\}^*$ and $v \in \{BA, BBA\}^*$, with $uv \neq \lambda$. The first phase ends with applying a rule of the form $S \rightarrow uv$, where u, v are as in the previous phrase. In phase two, the erasing rule $ABBBA \rightarrow \lambda$ is applied.
6. In view of Property 1, we can assume that in any sentential form that is derivable according to G , there is a symbol X occurring to the right of a substring $ABBBA$ and X will satisfy $X \in T \cup \{B\}$.

We are now considering another normal form due to Geffert [3, Theorem 4], where we have four nonterminals S, A, B, C , and non-context-free erasing rules of the form $AB \rightarrow \lambda$ and $CC \rightarrow \lambda$; we call this (4, 2)-GNF for short. This normal form is obtained from the more classical one (using nonterminals S, A, B, C, D) by applying the morphism $A \mapsto CAA, B \mapsto BBC, C \mapsto CA$ and $D \mapsto BC$ to all rules. This already implies that the following properties hold for this normal form.

1. If $S \Rightarrow^* w$, then $w \in \{CA, CAA\}^* \{S, CC, \lambda\} \{BC, BBC\}^* T^*$.
2. No sentential form derivable in the grammar contains any of the substrings BA, AAA, BBB, CCC .
3. If $S \Rightarrow^* w$, then w contains at most one of the two substrings CC, AB (called the *central part*) at most once. This also rules out $ABCC$ and $ABAB$ as substrings of sentential forms.
4. Again, the derivation proceeds in two phases, the first one split into two stages. Only in phase two, the central part (either AB or CC) will appear.

3 Main Results

In this section, the main results of this paper (see Table 1) are presented.

1: $(A \rightarrow \$S, AB, S)$	2: $(B \rightarrow \#, \$S, \#)$
3: $(S \rightarrow \$, S\#, 0)$	4: $(C \rightarrow \$\$, CC, \$)$
5: $(C \rightarrow \#, \$\$, \#)$	6: $(\$ \rightarrow \lambda, \$\#, 0)$
7: $(\# \rightarrow \lambda, 0, \$)$	$w: (S \rightarrow w, 0, \$)$

Fig. 1. Semi-conditional rules simulating $AB \rightarrow \lambda$ and $CC \rightarrow \lambda$ and $S \rightarrow w \in P_{cf}$.

¹ We collect all rules $S \rightarrow w$ with $S \notin \text{sub}(w)$ plus the only rule $S \rightarrow uS$ within P_{cf} .

Theorem 1. *For each RE language L , there is an SCG(2, 1) with only six non-terminals that describes L .*

Proof. Consider some RE language L represented by some type-0 grammar $G = (\{S, A, B, C\}, T, P, S)$ in (4, 2)-GNF. G is simulated by a semi-conditional grammar of degree (2, 1) $G' = (V, T, P', S)$, where $V = N \cup \{\$, \#\} \cup T$.

The particular non-context-free erasing rules $AB \rightarrow \lambda$ and $CC \rightarrow \lambda$ of G are simulated by the semi-conditional rules given in Fig. 1 where the last rule is considered for each context-free rule $S \rightarrow w$ of the type-0 grammar G .

The intended derivations of phase one of G can be simulated by literally the same rules (where only the forbidden context $\$$ was added that has no influence here as $\$$ is not present until end of phase one). Observe that it is not possible to start such a simulation when still being in the phase one simulation, as AB or CC must be present as a substring, which can only happen after finishing with the simulation of phase one.

In phase two, we consider a sentential form uvt , where $u \in \{CA, CAA\}^*$, $v \in \{BC, BBC\}^*$ and $t \in T^*$ with $S \Rightarrow^* uvt$. If u ends with A and v starts with B , i.e., $u = u'A$ and $v = Bv'$, $AB \rightarrow \lambda$ can be applied. In the simulating semi-conditional grammar, we find

$$uvt = u'ABv't \Rightarrow_1 u'\$SBv't \Rightarrow_2 u'\$S\#v't \Rightarrow_3 u'\$\$\#v't \Rightarrow_6 u'\#v't \Rightarrow_7 u'v't.$$

If instead CC is a substring of the current sentential form uvt , this means that $u = u'C$ and $v = Cv'$, and now $CC \rightarrow \lambda$ is simulated as follows:

$$uvt = u'CCv't \Rightarrow_4 u'\$\$Cv't \Rightarrow_5 u'\$\$\#v't \Rightarrow_6 u'\#v't \Rightarrow_7 u'v't.$$

By induction, this argument shows that $L(G) \subseteq L(G')$.

Conversely, consider a sentential form w derivable by the simulating grammar G' that is also derivable by the (4, 2)-GNF grammar G . Assume first that w contains some occurrence of S . As w is derivable in G , neither $\$$ nor $\#$ occur in w only rules of the type $(S \rightarrow x, 0, \beta)$ apply, $\beta \in \{0, \$\}$. In particular, Rule 3 is not applicable. Hence, for the string w' derivable from w in one step we also find $w \Rightarrow_{G'} w'$. This means that we are simulating phase one of G faithfully with G' .

We are left with the case that w does not contain S . As w is also derivable in G , we are in (the simulation of) phase two. Still, w does not contain any occurrences of $\$$ and $\#$ either. If w is a terminal string, nothing remains to be shown. So, w contains some occurrences from $\{A, B, C\}$ and satisfies Property 1 of (4, 2)-GNF. Hence, $w \in \{CA, CAA\}^* \{S, CC, \lambda\} \{BC, BBC\}^* T^*$. All rules but Rule 1 and Rule 4 require that $sub(w) \cap \{S, \#, \$\} \neq \emptyset$. If the central part AB is a substring of w , then Rule 1 applies, starting an intended simulation cycle as discussed above. We will look into this intended derivation later.

If AB is a substring of w , then $w = u'ABv't$, with $u' \in \{CA, CAA\}^* \{C, CA\}$, $v' \in \{C, BC\} \{BC, BBC\}^*$ and $t \in T^*$ (by induction), so that only Rule 1 is applicable. This leads us to w_1 , replacing any occurrence of A by $\$S$. As $\{\#, \$\} \cap sub(w_1) = \emptyset$, only Rule 2 is applicable on w_1 . The resulting string w_2 is obtained from w_1 by replacing some occurrence of B by $\#$. As $\$, \#, S \in sub(w_2)$,

only Rules 3 or 6 apply. As w_2 will have S occurring to the right of the only occurrence of $\$$, Rule 6 cannot be applied, leaving us with Rule 3 as the only possibility. In order to have $S\# \in \text{sub}(w_2)$, $w_2 = u'\$S\#v't$ is enforced due to Properties 2 and 3 of (4,2)-GNF, as there are no other occurrences of the substring AB in w but the one that we singled out. Hence, with $w_2 \Rightarrow w_3$, we know that $w_3 = u'\$\#v't$. Due to the forbidden contexts, only Rules 3 or 6 might apply, with Rule 3 being disabled, as $S \notin \text{sub}(w_3)$. In order to apply Rule 6, $\#\# \in \text{sub}(w_3)$ is required. Hence, $w_3 \Rightarrow w_4 = u'\#\#v't$ follows. As $S \notin \text{sub}(w_4)$, Rules 1, 2, 3 are not applicable (and of course, no rule from phase one). As $\$ \notin \text{sub}(w_4)$, Rules 5 and 6 do not apply, either. Hence, we are left with either applying Rule 4 or Rule 7 next. If we applied Rule 4, any occurrence of C is replaced by $\#\#$, leading to some string w_5 . With AB , S and $\#$ being absent in w_5 but $\#\#$ present as substrings, Rule 5 is the only potentially applicable rule. However, this would require CC to appear as a substring in w_4 , which would mean that CC appears twice as a substring in w , contradicting Properties 2 and 3 of (4,2)-GNF. Hence, this derivation gets stuck. Rule 7 would be the intended rule to be applied on w_4 , leading us to some w' that satisfies $w \Rightarrow_G w'$ by applying the rule $AB \rightarrow \lambda$ from G , this way finishing our argument in this case.

The argument for the case that CC is a substring of w is along similar lines. We can also rule out using rules $(S \rightarrow x, 0, 0)$.

By induction, we see that $L(G') \subseteq L(G)$. □

By replacing the use of symbol S by a new nonterminal \dagger , we can avoid using an unbounded number of conditional rules. This would yield a result nearly matching the one of Okubo [8].

It is open if the number of nonterminals can be further reduced for semi-conditional grammars of degree (2,1). In the following result, we show that if we increase the (maximum) forbidden context length to two, then with 6 conditional rules and 6 nonterminals we can obtain a computational completeness result. Notice that we are not aware of any previous results for SCG($i, 2$) for any i (Table 1).

Theorem 2. *For each RE language L , there is an SCG(2,2) with only six non-terminals and six conditional rules that describes L .*

- | | |
|--|---|
| 1: $(A \rightarrow \$, AB, \$)$ | 2: $(B \rightarrow \# , \$B, \#)$ |
| 3: $(C \rightarrow \#\$, CC, \#)$ | 4: $(C \rightarrow \#\#, \$C, \#\#)$ |
| 5: $(\$ \rightarrow \lambda , \#\#, AB)$ | 6: $(\# \rightarrow \lambda , 0 , \$)$ |

Fig. 2. Semi-conditional rules for simulating $AB \rightarrow \lambda$ and $CC \rightarrow \lambda$.

For reasons of space, we only provide an explanation of how the simulation of the non-context-free erasing rules of a type-0 grammar in (4,2)-GNF should work; cf. Fig. 1.

1: $(B \rightarrow \$, BBB , \$)$	2: $(B \rightarrow \# , B\$B , \#)$
3.A: $(\$ \rightarrow \lambda , \#\$A , S)$	3.B: $(\$ \rightarrow \lambda , \#\$B , S)$
4: $(B \rightarrow S\$, A\#B , \$)$	5: $(A \rightarrow \$\$, A\#A , \$)$
6: $(\# \rightarrow \lambda , \$\$ \# , S)$	7: $(A \rightarrow \# , \$\$A , \#)$
8.X: $(\$ \rightarrow \lambda , \$\#X , S)$	9.X: $(\# \rightarrow \lambda , \#X , \$)$
$\delta S: (S \rightarrow \lambda , S\$A , 0)$	$w: (S \rightarrow w , 0 , \$)$

Fig. 3. Semi-conditional rules for simulating $ABBBA \rightarrow \lambda$, with $X \in T \cup \{B\}$, and for simulating $S \rightarrow w \in P_{cf}$.

$$wvt = u'ABv't \Rightarrow_1 u'\$Bv't \Rightarrow_2 u'\#\$v't \Rightarrow_5 u'\#v't \Rightarrow_6 u'v't,$$

$$wvt = u'CCv't \Rightarrow_3 u'\#\$Cv't \Rightarrow_4 u'\#\#\$v't \Rightarrow_5 u'\#\#\#v't \Rightarrow_6^3 u'v't.$$

Aiming at further trade-off results, trading the number of nonterminals for degree conditions, we consider semi-conditional grammars of degree (3, 1). In [8], it is shown that 7 nonterminals with degree (2, 1) suffice to describe any RE language. Below, we improve on the nonterminal number, but with degree (3, 1).

Theorem 3. *For each RE language L, there is an SCG(3, 1) with only five nonterminals that describes L.*

Proof. We start out with a representation of a recursively enumerable language by a type-0 grammar $G = (N, T, P, S)$ in (3, 1)-GNF that has only three nonterminals S, A, B . We are going to construct an equivalent an SCG(3, 1) G' with additional nonterminals $\$$ and $\#$. G has context-free rules of the form $S \rightarrow w$ that we simply also incorporate in the semi-conditional grammar G' that we produce without any conditions (apart from the rules that replace S by a terminal string), plus one non-context-free deletion rule, namely $ABBBA \rightarrow \lambda$. This particular rule is simulated by the semi-conditional rules listed in Fig. 3. The intended simulation is as follows, basically executing the given rules in order.

$$\begin{aligned}
 & ABBBAX \Rightarrow_1 AB\$BAX \Rightarrow_2 A\#\$BAX \Rightarrow_{3.B} A\#BAX \Rightarrow_4 A\#S\$AX \Rightarrow_{\delta S} \\
 & A\#\$AX \Rightarrow_{3.A} A\#AX \Rightarrow_5 \$\#\$AX \Rightarrow_6 \$\$AX \Rightarrow_7 \$\#\#X \Rightarrow_8 \#\#X \Rightarrow_9 \#X
 \end{aligned}$$

Notice that we might assume that in any sentential form that is derivable according to G , a symbol X occurs to the right of the substring $ABBBA$ that satisfies $X \in T \cup \{B\}$ according to the last (3, 1)-GNF property. So, we can assume that the symbol X that we are testing in Rules 9 and 10 does exist. This shows that $L(G) \subseteq L(G')$. Before we continue with proving the converse inclusion $L(G') \subseteq L(G)$, we observe the following.

Observation: If $w \in (T \cup \{A, B\})^*$ and if $w \Rightarrow w'$ in G' , then there is only at most one applicable rule in G' , which is Rule 1. Furthermore, this requires that $BBB \in sub(w)$. This claim is easily understood by inspecting the rules listed in Fig. 3, as all other rules require that $\{\$, \#\} \cap sub(w) \neq \emptyset$. This nicely fits with Property 4 of (3, 1)-GNF grammars.

Now we prove the converse inclusion $L(G') \subseteq L(G)$. Consider some sentential form w of G' . By induction, we can assume that w is also a sentential form of G . We are going to argue that after some further derivation steps of G' (starting from w), either the derivation is stuck, or some sentential form is produced that is also a valid sentential form of G . By induction, this shows $L(G') \subseteq L(G)$.

As assumed above, w is a valid sentential form in G also, so it is possibly part of a derivation taking place in phase one in this (3, 1)-GNF grammar only. Hence, only context-free rules have been used, and this could be also assumed with G' , i.e., the simulation was the intended one. In particular, none of the nonterminals $S, \$, \#$ occurs in w .

Hence, we are now in phase two of the (3, 1)-GNF grammar. According to the properties of (3, 1)-GNF listed above, the interesting cases that remain to be discussed is when (i) $w = uABBBAvt$ or (ii) $w = uABBBBAvt$ or (iii) $w = uABBAvt$ with $u \in \{AB, ABB\}^*$, $v \in \{BA, BBA\}^*$, where in the latter two cases (ii) and (iii), there is no way to terminate the derivation in G according to Properties 3 and 4 of (3, 1)-GNF grammars. According to Property 5 of (3, 1)-GNF grammars the only rule applicable in G would be the non-context-free deletion rule. In G' , we also find that at most some rules from Fig. 3 might apply. By our Observation, Case (iii) is ruled out. In G' , as can be easily checked, the only applicable rule would then be the first rule $B \rightarrow \$$. This results in a string w_1 with $w \Rightarrow w_1$ where one occurrence of B is replaced by $\$$. Now, due to the fact that $\$$ is a forbidden context in the first rule, it cannot be applied again. We can check that rules numbered 3, 4, 5, 6, 8, 9 are not applicable, as $\#$ is not contained in w_1 . As $\$\$ \notin \text{sub}(w_1)$, neither Rule 7 is applicable. So, we have to employ Rule 2. In order to continue, we can conclude that (1) $w_1 = uAB\$BAvt$ or (2) $w_1 = uAB\$BBAvt$ or (3) $w_1 = uABB\$BAvt$, as in the given normal form grammar G , no strings with a sequence of more than four B 's can be produced, see Property 2 of (3, 1)-GNF grammars, and there is only one position where one could find a sequence of three or four B 's according to Property 3 of (3, 1)-GNF. It is then clear that for $w_1 \Rightarrow w_2$ we have that (a) $w_2 = uA\#\$BAvt$ or (b) $w_2 = uAB\#\$Avt$ (both from $w_1 = uAB\$BAvt$), or (c) $w_2 = uA\#\$BBAvt$ or (d) $w_2 = uAB\#\$BAvt$ or (e) $w_2 = uAB\$B\#Avt$ (all three from $w_1 = uAB\$BBAvt$), or (f) $w_2 = uA\#B\$BAvt$ or (g) $w_2 = uAB\#\$BAvt$ or (h) $w_2 = uABB\#\$Avt$ (all three from $w_1 = uABB\$BAvt$), or (j) any other occurrence of B within u or v is replaced, leading to w_2 where the subword between the u - and the v -part is identical to w_1 .

In each case, both $\$$ and $\#$ are present in the string w_2 , which immediately rules out all but Rules 3, 6, 8 due to the forbidden strings in the other rules. Rule 6 requires two occurrences of $\$$, which is not satisfied with any of the w_2 . We are now discussing possibilities for $w_2 \Rightarrow w_3$ with the two remaining rules.

If we try to apply Rule 8, this requires the substring $\#\#X$ for $X \in T \cup \{B\}$, which leaves us with Case (d). In that case, $w_2 \Rightarrow uAB\#BAvt = w_3$. Now, Rule 1 is excluded, as substring BBB is missing; Rule 2 is excluded by the presence of $\#$; Rules 3 and 8 are not possible, as $\$$ is not present; Rule 4 is not applicable, as the substring $A\#B$ is not present; similarly, Rule 5 is not

applicable, as the substring $A\#A$ is missing; Rules 6 and 7 require two occurrences of $\$$ in the string, which is not the case here. Applying Rule 9 leads us to $w_4 = uABBAvt$. By our Observation, there is no possible continuation.

If we want to apply Rule 3 instead to the string w_2 , which in fact has the same effect as Rule 8, namely, to delete the only occurrence of $\$$ in w_2 , this means that $\#\$Y \in \text{sub}(w_2)$ for $Y = A$ or $Y = B$. This leaves us with Cases (a) or (c) or (g). Some further case analysis shows that only Case (a) leads to continuation, forcing us step-by-step along the intended derivation, this way arriving at $w_6 = uA\#Avt$.²

As $\{BBB, B\$B, \$, A\#B, \#X \mid X \in T \cup \{B\}\} \cap \text{sub}(w_6) = \emptyset$, the only applicable rule is Rule 5. Hence, (A) $w_7 = u\$\Avt , or (B) $w_7 = uA\#\$\vt , or (C) $w_7 = u'A\#Av't$, where $u = u'$ or $v = v'$ and exactly one occurrence of A in uv was replaced by $\$\$$ in order to get $u'v'$. As $\{\$\$, \#\} \subseteq \text{sub}(w_7)$, only Rules 3, 6, or 8 might be applicable. In Case (A), only Rule 6 is applicable by checking the permitting conditions, i.e., $w_8 = u\$\Avt . In Case (B), Rule 3 is not applicable, as the symbol to the right of $\#\$$ is not A or B . The permitting conditions of Rules 6 and 8 require the substring $\#\$$, which is not present in this case. The same reasoning shows that Case (C) knows no continuation. To summarize, we are forced to $w_8 = u\$\Avt . As $\$$ is present in w_8 , we cannot apply Rules 1, 4, 5, or 9. As $\{B\$B, \#\$A, \#\$B, \#\} \cap \text{sub}(w_8) = \emptyset$, the only applicable rule is Rule 7. This leads us to $w_9 = u\$\vt or $u'\$\$Av't$, where $u' = u$ or $v' = v$, and either v' is obtained from v by replacing one occurrence of A by $\#$, or u' is derived from u in a similar fashion. Anyways, the presence of $\$$ and $\#$ in the new string restricts us to using Rules 3, 6, 8 in the next derivation step, which always requires $\$$ and $\#$ to be neighboring symbols, leaving us with $w_9 = u\$\vt as the only possibility, as u cannot end with A (and hence u' cannot end with $\#$) and v cannot start with A (and hence v' cannot start with $\#$).

As $\{\$\$, \#\} \subseteq \text{sub}(w_9)$, only Rules 3, 6 or 8 might be applicable to w_9 . The permitting condition of Rule 3 is not satisfied. If we apply Rule 6, we arrive at $u\$\vt . As can be checked by a case-by-case analysis, there is no rule that applies to transform $u\$\vt any further in G' . If we use Rule 8 instead, we get $w_{10} = u\#\$vt$. Again, only Rules 3, 6 or 8 might be applicable. The required permitting strings of Rules 3 and 6 are not subwords of w_{10} . Hence, we have to apply Rule 8 once more, leading to $w_{11} = u\#vt$. Now, recall that u ends with a B and v starts with a B , unless they are empty. Hence, Rules 4 and 5 do not apply. The presence of $\#$ prevents applying Rules 2 and 7. The absence of $\$$ does not allow us to apply Rules 3, 6, 7 or 8, either. As $BBB \notin \text{sub}(w_{11})$, Rule 1 cannot be applied. Hence, only Rule 9 is applicable, yielding $w_{12} = uvt$. As $w \Rightarrow w_{12}$ in G is valid (recall that we are still considering Case (i) from the beginning of this analysis), the induction step is shown.

There is one issue that needs to be finally addressed. As S is introduced as an auxiliary symbol in the conditional rules in Fig. 3, within G' , a new simulation of a derivation $S \Rightarrow^* x$ in G might start. This possibility is stopped by the fact that rules of G that are terminating the first phase, i.e., rules of the form $S \rightarrow w$

² Some details are suppressed due to space constraints.

for $S \notin \text{sub}(w)$, are now guarded by the condition checking if $\$$ is absent. So, if such a restart is attempted, then this could never lead to a terminal string. \square

We can get a better bound on the number of conditional rules at the expense of one additional nonterminal \dagger in Fig. 4.

1: $(B \rightarrow \$, BBB , \$)$	2: $(B \rightarrow \# , B\$B , \#)$
3.A: $(\$ \rightarrow \lambda , \#\$A , \dagger)$	3.B: $(\$ \rightarrow \lambda , \#\$B , \dagger)$
4: $(B \rightarrow \dagger\$, A\#B , \$)$	5: $(A \rightarrow \$\$, A\#A , \$)$
6: $(\# \rightarrow \lambda , \$\#\# , \dagger)$	7: $(A \rightarrow \# , \$\$A , \#)$
8: $(\$ \rightarrow \lambda , \$\#B , \dagger)$	8': $(\$ \rightarrow \lambda , 0 , A)$
9: $(\# \rightarrow \lambda , \#B , \$)$	9': $(\# \rightarrow \lambda , 0 , A)$
$\delta\dagger$: $(\dagger \rightarrow \lambda , \dagger\$A , 0)$	w : $(S \rightarrow w , 0 , 0)$

Fig. 4. Semi-conditional rules simulating $ABBBA \rightarrow \lambda$ and $S \rightarrow w$

Theorem 4. *For each recursively enumerable language L , there is an SCG(3, 1) with only six nonterminals and 13 conditional rules that describes L .*

Future Research Questions. The interested reader might have wondered why we restricted our attention to (simple) semi-conditional grammars of degree (i, j) with $i > 1$, although also and even degree $(1, 1)$ is known to suffice for computational completeness. However, the construction of [5] (based on [7]) is not helpful to get any bounds on the nonterminal complexity of such systems; even if we start out with a matrix grammar with few nonterminals (as done in the construction of Mayer), and notice that few nonterminals suffice in matrix grammars, as shown in [2], we arrive at a semi-conditional grammar with quite a number of nonterminals, as the matrix grammars of [7] are binary to start with, while in [2], the matrix grammars that are constructed contain arbitrarily many rule in their matrices. We pose it as an open question to construct a (simple) semi-conditional grammars of degree $(1, 1)$ using only (any) fixed amount of nonterminals. Clearly, other open questions concern the optimality of our nonterminal complexity results for degrees $(2, 1)$, $(2, 2)$ or $(3, 1)$.

References

1. Dassow, J., Păun, G.: Regulated Rewriting in Formal Language Theory: EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (1989)
2. Fernau, H., Freund, R., Oswald, M., Reinhardt, K.: Refining the nonterminal complexity of graph-controlled, programmed, and matrix grammars. J. Autom. Lang. Comb. **12**(1/2), 117–138 (2007)
3. Geffert, V.: Normal forms for phrase-structure grammars. RAIRO Informatique théorique et Applications/Theor. Inform. Appl. **25**, 473–498 (1991)

4. Masopust, T.: Formal models: regulation and reduction. Ph.D. thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic (2007)
5. Masopust, T.: A note on the generative power of some simple variants of context-free grammars regulated by context conditions. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 554–565. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00982-2_47
6. Masopust, T., Meduna, A.: Descriptive complexity of semi-conditional grammars. *Inf. Process. Lett.* **104**(1), 29–31 (2007)
7. Mayer, O.: Some restrictive devices for context-free languages. *Inf. Control* (now *Inf. Comput.*) **20**, 69–92 (1972)
8. Okubo, F.: A note on the descriptive complexity of semi-conditional grammars. *Inf. Process. Lett.* **110**(1), 36–40 (2009)
9. Vaszil, G.: On the descriptive complexity of some rewriting mechanisms regulated by context conditions. *Theor. Comput. Sci.* **330**, 361–373 (2005)



Kernelization Lower Bounds for Finding Constant-Size Subgraphs

Till Fluschnik¹(✉), George B. Mertzios², and André Nichterlein¹

¹ Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany

{till.fluschnik, andre.nichterlein}@tu-berlin.de

² Department of Computer Science, Durham University, Durham, UK
george.mertzios@durham.ac.uk

Abstract. Kernelization is an important tool in parameterized algorithms. Given an input instance accompanied by a parameter, the goal is to compute in polynomial time an equivalent instance of the same problem such that the size of the reduced instance only depends on the parameter and not on the size of the original instance. In this paper, we provide a first conceptual study on limits of kernelization for several *polynomial-time* solvable problems. For instance, we consider the problem of finding a triangle with negative sum of edge weights parameterized by the maximum degree of the input graph. We prove that a linear-time computable strict kernel of truly subcubic size for this problem violates the popular APSP-conjecture.

1 Introduction

Kernelization is the main mathematical concept for provably efficient preprocessing of computationally hard problems. This concept has been extensively studied (see, e.g., [17, 21, 26, 27]) and it has great potential for delivering practically relevant algorithms [24, 31]. In a nutshell, the aim is to significantly and efficiently reduce a given instance of a parameterized problem to its “computationally hard core”. Formally, given an instance $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ of a parameterized problem L , a *kernelization* for L is an algorithm that computes in polynomial time an instance (x', k') , called kernel, such that (i) $(x, k) \in L \iff (x', k') \in L$ and (ii) $|x'| + k' \leq f(k)$, for some computable function f . Although studied mostly for NP-hard problems, it is natural to apply this concept also to polynomial-time solvable problems as done e.g. for finding maximum matchings [29]. It is thus also important to know the *limits* of this concept. In this paper we initiate a systematic approach to derive *kernelization lower bounds* for problems in P. We

T. Fluschnik—Supported by the DFG, project DAMM (NI 369/13) and project TORE (NI 369/18).

G. B. Mertzios—Partially supported by the EPSRC grant EP/P020372/1.

A. Nichterlein—Supported by a postdoc fellowship of DAAD while at Durham University.

demonstrate our techniques at the example of subgraph isomorphism problems where the sought induced subgraph has constant size and is connected.

When kernelization is studied on NP-hard problems (where polynomial running times are considered computationally “tractable”), the main point of interest becomes the *size* $f(k)$ of the kernel with respect to the parameter k . In particular, from a theoretical point of view, one typically wishes to minimize the kernel size to an—ideally—polynomial function f of small degree. As every decision problem in P admits a kernelization which simply solves the input instance and produces a kernel of size $O(1)$ (encoding the YES/ NO answer), it is crucial to investigate the *trade-off* between (i) the size of the kernel and (ii) the running time of the kernelization algorithm. The following notion captures this trade-off: An (a, b) -kernelization for a parameterized problem L is an algorithm that, given any instance $(x, k) \in \{0, 1\}^* \times \mathbb{N}$, computes in $O(a(|x|))$ time an instance (x', k') such that (i) $(x, k) \in L \iff (x', k') \in L$ and (ii) $|x'| + k' \in O(b(k))$.

Kernelization for problems in P is part of the recently introduced framework “FPT in P” [20]. This framework is recently applied to investigate parameterized algorithms and complexity for problems in P [3, 14, 16, 20, 29]. Studying lower bounds for kernelization for problems in P is—as it turns out—strongly connected to the active research field of *lower bounds* on the running times of polynomial-time solvable problems (see, e.g., [1–3, 7]). These running time lower bounds rely on popular conjectures like the Strong Exponential Time Hypothesis (SETH) [22, 23] or the 3SUM-conjecture [19], for instance.

In contrast to NP-hard problems, only little is known about kernelization lower bounds for problems in P. To the best of our knowledge all known kernelization lower bounds follow trivially from the corresponding lower bounds of the running time: For instance, assuming SETH, it is known that (i) the hyperbolicity and (ii) the diameter of a graph cannot be computed in $2^{o(k)} \cdot n^{2-\varepsilon}$ time for any $\varepsilon > 0$, where k is (i) the vertex cover number and (ii) the treewidth of the graph [3, 14]. This implies that both problems do not admit an $(n^{2-\varepsilon}, 2^{o(k)})$ -kernelization—a kernel with $2^{o(k)}$ vertices computable in $O(n^{2-\varepsilon})$ time—since such a kernelization yields an algorithm running in $O(2^{o(k)} + n^{2-\varepsilon})$ time.

In this paper we initiate a systematic approach to derive kernelization lower bounds for problems in P for a—very natural—special type of kernels.

Definition 1 (strict (a, b) -kernelization). A strict (a, b) -kernelization for a parameterized problem L is an algorithm that given any instance $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ computes in $O(a(|x|))$ time an instance (x', k') such that (i) $(x, k) \in L \iff (x', k') \in L$, (ii) $|x'| + k' \in O(b(k))$, and (iii) $k' \leq k$.

Chen et al. [8] introduced a framework to exclude strict kernels for NP-hard problems, assuming that $P \neq NP$. Fernau et al. [13] applied the framework to a wide variety of FPT problems and studied it on “less” strict kernelizations. The framework [8, 13] is based on the notion of (*strong*) *diminishers*:

Definition 2 (a -diminisher). An a -diminisher for a parameterized problem L is an algorithm that given any instance $(x, k) \in \{0, 1\}^* \times \mathbb{N}$ in $O(a(|x|))$ time either decides whether $(x, k) \in L$ or computes an instance (x', k') such that (i)

Table 1. Overview of our results. Here, k is interchangeably the order of the largest connected component, the degeneracy, or the maximum degree.

	NEGATIVE WEIGHT TRIANGLE (NWT)	TRIANGLE COLLECTION (TC)
lower bounds (Theorem 2)	No strict (n^α, k^β) -kernelization with $\alpha, \beta \geq 1$ and $\alpha \cdot \beta < 3$, assuming: the APSP-conjecture	the SETH, APSP-, or 3SUM-conjecture
kernel (Theorem 3)	Strict $(n^{(3+\varepsilon)/(1+\varepsilon)}, k^{1+\varepsilon})$ -kernelization for every $\varepsilon > 0$, e.g. strict $(n^{5/3}, k^3)$ -kernelization	

$(x, k) \in L \iff (x', k') \in L$, and (ii) $k' < k$. A strong a -diminisher for L is an a -diminisher for L with $k' < k/c$ for some constant $c > 1$.

Our Contributions. We adapt the diminisher framework [8, 13] to prove kernelization lower bounds for problems in P. Our results concern the H -SUBGRAPH ISOMORPHISM (H -SI) problem¹ for *constant-sized connected* graphs H . As a running example, we focus on the fundamental case where H is a triangle and we present diminishers (along with kernelization lower bounds) for the following weighted and colored variants of the problem:

NEGATIVE WEIGHT TRIANGLE (NWT)

Input: An undirected graph G with edge weights $w: E(G) \rightarrow \mathbb{Z}$.

Question: Is there a triangle T in G with $\sum_{e \in E(T)} w(e) < 0$?

TRIANGLE COLLECTION (TC)

Input: An undirected graph G with surjective coloring $\text{col}: V(G) \rightarrow [f]$.

Question: Does there for all color-triples $C \in \binom{[f]}{3}$ exist a triangle with vertex set $T = \{x, y, z\}$ in G such that $\text{col}(T) = C$?

NWT and TC are conditionally hard: If NWT admits a *truly* subcubic algorithm—that is, with running time $O(n^{3-\varepsilon})$, $\varepsilon > 0$ —then APSP also admits a truly subcubic algorithm, breaking the APSP-conjecture [30]. A truly subcubic algorithm for TC breaks the SETH, the 3SUM-, and the APSP-conjecture [4].

For both NWT and TC we consider three parameters (in decreasing order): (i) order (that is, the number of vertices) of the largest connected component, (ii) maximum degree, and (iii) degeneracy. We prove that both NWT and TC admit a strong linear-time diminisher for all these three parameters. Together with the conditional hardness, we then obtain lower bounds on strict kernelization. Our results are summarized in Table 1.

Complementing our lower bounds, we prove a strict $(n^{5/3}, k^3)$ -kernelization for NWT and TC (k being any of the three aforementioned parameters) and a

¹ The H -SUBGRAPH ISOMORPHISM asks, given an undirected graph $G = (V, E)$, whether G contains H as a subgraph.

strict $(n \cdot \Delta^{\lfloor c/2 \rfloor + 1}, \Delta^{\lfloor c/2 \rfloor + 1})$ -Turing kernelization for H -SUBGRAPH ISOMORPHISM when parameterized by the maximum degree Δ , where $c = |V(H)|$.

Notation and Preliminaries. We use standard notation from parameterized complexity [10] and graph theory [11]. For an integer j , we define $[j] := \{1, \dots, j\}$.

2 Frameworks to Exclude Polynomial Kernelizations

We briefly recall the existing frameworks to exclude (strict) polynomial-size kernels for NP-hard problems. We further discuss the difficulties that appear when transferring these approaches to polynomial-time solvable problems.

Composition Framework. The frequently used (cross-)composition frameworks [5, 6, 18] are the tools to exclude polynomial-size problem kernels under the assumption $\text{NP} \subseteq \text{coNP/poly}$. There are some issues when adapting these frameworks for problems in P. We discuss the issues using the H -SUBGRAPH ISOMORPHISM (H -SI) problem for constant-sized connected H .

Adapting the proofs of Bodlaender et al. [5] and Fortnow and Santhanam [18] for H -SI leads to the following: “If H -SI parameterized by the order k of the largest connected component admits an $(n^c, k^{c'})$ -kernelization, then \overline{H} -SI $\in \text{NTIME}(n^{c'(c+1)})/n^{c+1}$.” Since there exists a trivial $O(n^{|H|+1})$ -time brute-force algorithm for H -SI, there also exist trivial polynomial-time computable kernels for H -SI. Hence, we have to stick with specifically chosen c and c' (with $c \cdot c' < |H|$). Furthermore, we cannot transfer these results easily to other problems in P due to the lack of a suitable completeness theory (H -SI belongs to P).

One drawback of the composition approach for any problem L in P is the lack of clarity on the assumption’s $(\overline{L} \notin \text{NTIME}(n^{c'(c+1)})/n^{c+1})$ reasonability. Moreover, due to a missing equivalent to the NP-completeness theory, the assumption bases on specific problems and not on complexity classes.

Strict Kernelization and Diminishers. Chen et al. [8] introduced a framework to exclude *strict* kernelization, that is, kernelization that do not allow an increase in the value of the parameter in the obtained kernel instance. This framework builds on the assumption $\text{P} \neq \text{NP}$ and can be easily adapted to exclude strict kernels for polynomial-time solvable problems. Recall that for problems in P, both the size of the kernel *and* the kernelization running time are important.

Theorem 1 (\star^2). *Let L be a parameterized problem with parameter k such that each instance with parameter $k \leq c$ for some constant $c > 0$ is a trivial instance of L . If L with parameter k admits a strict (a, b) -kernelization and an a' -diminisher (a strong a' -diminisher), then any instance (x, k) is solvable in $O(k \cdot (a(a'(b(k))) + a(|x|)))$ time (in $O(\log k \cdot (a(a'(b(k))) + a(|x|)))$ time).*

² Results marked with (\star) are deferred to a long version [15] of the paper.

We point out that—in contrast to “classic” kernelization for NP-hard problems—for two parameters k and k' for a problem L such that k' is stronger [25] than k , a strict kernelization regarding k does not imply a strict kernelization regarding k' .

Reductions for Transferring Kernels. There are two issues when using the strategy of *polynomial parameter transformations* to transfer results of Theorem 1 along polynomial-time solvable problems: First, we need to require the transformation to be computable “fast” enough and that the parameter does not increase ($k' \leq k$). Second, in order to transfer a strict kernel we need to show a reverse transformation from L' to L which again is computable “quick” enough and does not increase the parameter. Hence, we essentially need to show that the two problems L and L' are *equivalent* under these restrictive transformations.

3 Kernelization Lower Bounds via Diminishers

In this section, we present diminishers for H -SUBGRAPH ISOMORPHISM (H -SI) for connected H with respect to the structural parameters (i) order ℓ of the largest connected component, (ii) maximum degree Δ , and (iii) degeneracy d . Observe that $d \leq \Delta \leq \ell$ in every graph. These lead to our following main result.

Theorem 2. *If NWT (TC) parameterized by k being the (i) order ℓ of the largest connected component, (ii) maximum degree Δ , or (iii) degeneracy d admits a strict (n^α, k^β) -kernel for constants $\alpha, \beta \geq 1$ with $\alpha \cdot \beta < 3$, then the APSP-conjecture (the SETH, the 3SUM-, and the APSP-conjecture) breaks.*

Parameter Order of the Largest Connected Component. In the following, we prove a linear-time strong diminisher regarding the parameter order of the largest connected component for problems of finding constant-size subgraphs (with some specific property). The idea behind our diminisher is depicted as follows: for each connected component, partition the connected component into small parts and then take the union of not too many parts to construct new (connected) components (see Fig. 1 for an illustration of the idea with H being a triangle).

Construction 1. Let H be an arbitrary but fixed connected constant-size graph of order $c > 1$. Let $G = (V, E)$ be a graph with the largest connected component being of order ℓ . First, compute in $O(n + m)$ time the connected components G_1, \dots, G_r of G . Then, construct a graph G' as follows.

Let G' be initially the empty graph. If $\ell \leq 4c$, then set $G' = G$. Otherwise, if $\ell > 4c$, then construct G' as follows. For each connected component $G_i = (V_i, E_i)$, do the following. If the connected component $G_i = (V_i, E_i)$ is of order at most $\ell/2$, then add G_i to G' . Otherwise, if $n_i := |V_i| > \ell/2$, then we partition V_i as follows. Without loss of generality let V_i be enumerated as $V_i = \{v_i^1, \dots, v_i^{n_i}\}$. For every $p \in \{1, \dots, 4c\}$, define $V_i^p := \{v_i^q \in V_i \mid q \bmod 4c = p - 1\}$. This defines the partition $V_i = V_i^1 \uplus \dots \uplus V_i^{4c}$. Then, for each $\{a_1, \dots, a_c\} \in \binom{[4c]}{c}$, add the graph $G[V_i^{a_1} \cup \dots \cup V_i^{a_c}]$ to G' . This completes the construction. \diamond

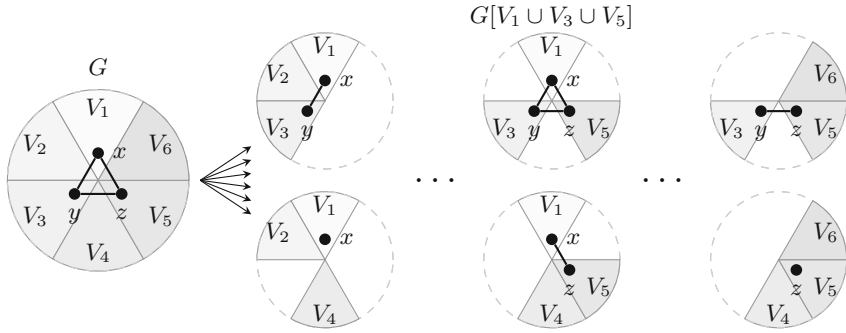


Fig. 1. Schematic illustration of the idea behind our diminisher for the parameter order of the largest connected component.

Employing Construction 1, we obtain the following.

Proposition 1 (★). *NWT and TC parameterized by the order ℓ of the largest connected component admit a strong $(n + m)$ -diminisher.*

There is a straight-forward $O(k^2 \cdot n)$ -time algorithm for NWT and TC: Check for each vertex all pairs of other vertices in the same connected component. However, under the APSP-conjecture (and SETH for TC) there are no $O(n^{3-\varepsilon})$ -time algorithms for any $\varepsilon > 0$ [4, 30]. Combining this with our diminisher in Proposition 1 we can exclude certain strict kernels as shown below.

Proof (of Theorem 2(i)). By Proposition 1, we know that NWT admits a strong $(n + m)$ -diminisher. Suppose that NWT admits a strict (n^α, k^β) -kernel for $\alpha \geq 1, \beta \geq 1$ with $\alpha \cdot \beta = 3 - \varepsilon_0, \varepsilon_0 > 0$. It follows by Theorem 1 that NWT is solvable in $t(n, k) \in O(k^{\beta \cdot \alpha} \log(k) + n^\alpha)$ time. Observe that $\log(k) \in O(k^{\varepsilon_1})$ for $0 < \varepsilon_1 < \varepsilon_0$. Together with $k \leq n$ and $\alpha \cdot \beta = 3 - \varepsilon_0$ we get $t(n, k) \in O(n^{3-\varepsilon})$ with $\varepsilon = \varepsilon_0 - \varepsilon_1 > 0$. Hence, the APSP-conjecture breaks [30]. The proof for TC works analogously. \square

Parameter Maximum Degree. The diminisher described in Construction 1 does not necessarily decrease the maximum degree of the graph. We thus adapt the diminisher to partition the edges of the given graph (using an (improper) edge-coloring) instead of its vertices. Furthermore, if H is of order c , then H can have up to $\binom{c}{2} \leq c^2$ edges. Thus, our diminisher considers all possibilities to choose c^2 (instead of c) parts of the partition. For the partitioning step, we need the following.

Lemma 1 (★). *Let $G = (V, E)$ be a graph with maximum degree Δ and let $b \in \mathbb{N}$. One can compute in $O(b(n + m))$ time an (improper) edge-coloring $\text{col}: E \rightarrow \mathbb{N}$ with less than $2b$ colors such that each vertex is incident to at most $\lceil \Delta/b \rceil$ edges of the same color.*

Construction 2. Let H be an arbitrary but fixed connected constant-size graph of order $c > 1$. Let $G = (V, E)$ be a graph with maximum degree Δ . First, employ Lemma 1 to compute an (improper) edge-coloring $\text{col}: E \rightarrow \mathbb{N}$ with $4c^2 \leq f < 8c^2$ many colors (without loss of generality we assume $\mathfrak{S}(\text{col}) = \{1, \dots, f\}$) such that each vertex is incident to at most $\lceil \Delta/(4c^2) \rceil$ edges of the same color.

Now, construct a graph G' as follows. Let G' be initially the empty graph. If $\Delta \leq 4c^2$, then set $G' = G$. Otherwise, if $\Delta > 4c^2$, then construct G' as follows. We first partition E : Let E^p be the edges of color p for every $p \in \{1, \dots, f\}$. Clearly, $E = E^1 \uplus \dots \uplus E^f$. Then, for each $\{a_1, \dots, a_{c^2}\} \in \binom{[f]}{c^2}$, add the graph $(V, E^{a_1} \cup \dots \cup E^{a_{c^2}})$ to G' . This completes the construction. \diamond

Proposition 2 (\star). *NWT and TC parameterized by maximum degree Δ admit a strong $(n + m)$ -diminisher.*

Parameter Degeneracy. The degeneracy of a graph is the smallest number d such that every induced subgraph contains a vertex of degree at most d . For parameter degeneracy, the diminisher follows the same idea as the diminisher for the parameter maximum degree (see Construction 2). The only difference between the two diminishers is how the partition of edge set is obtained.

Construction 3. Let H be an arbitrary but fixed constant-size graph of order $c > 1$. Let $G = (V, E)$ be a graph with degeneracy d . First, compute a degeneracy ordering³ σ in $O(n + m)$ time [28]. Construct a graph G' as follows.

Let G' be initially the empty graph. If $d \leq 4c^2$, then set $G' = G$. Otherwise, if $d > 4c^2$, then construct G' as follows. First, for each vertex $v \in V$, we partition the edge set $E_v := \{\{v, w\} \in E \mid \sigma(v) < \sigma(w)\}$ going to the right of v with respect to σ into $4c^2$ parts. Let E_v be enumerated as $\{e_1, \dots, e_{|E_v|}\}$. For each v , we define $E_v^p := \{e_i \in E_v \mid i \bmod 4c^2 = p - 1\}$ for every $p \in [4c^2]$. Clearly, $E_v = E_v^1 \uplus \dots \uplus E_v^{4c^2}$. Next, we define $E^p := \bigcup_{v \in V} E_v^p$ for every $p \in [4c^2]$. Clearly, $E = \biguplus_{1 \leq p \leq 4c^2} E^p = \biguplus_{1 \leq p \leq 4c^2} \biguplus_{v \in V} E_v^p$. Then, for each $\{a_1, \dots, a_{c^2}\} \in \binom{[4c^2]}{c^2}$, add the graph $(V, E^{a_1} \cup \dots \cup E^{a_{c^2}})$ to G' . This completes the construction. \diamond

Proposition 3 (\star). *NWT and TC parameterized by degeneracy admit a strong $(n + m)$ -diminisher.*

4 (Turing) Kernelization Upper Bounds

We complement our results on kernelization lower bounds by showing straightforward strict kernel results for H -SUBGRAPH ISOMORPHISM for connected constant-size H to show the limits of any approach showing kernel lower bounds.

Strict Turing Kernelization. For the parameters order of the largest connected component and maximum degree, we present strict (a, b) -Turing kernels:

³ This is an ordering of the vertices such that each vertex v has at most d neighbors ordered after v .

Definition 3. A strict (a, b) -Turing kernelization for a parameterized problem L is an algorithm that decides every input instance (x, k) in time $O(a(|x|))$ given access to an oracle that decides whether $(x', k') \in L$ for every instance (x', k') with $|x'| + k' \leq b(k)$ in constant time.

Note that the diminisher framework in its current form cannot be applied to exclude (strict) (a, b) -Turing kernelizations. In fact, it is easy to see that H -SUBGRAPH ISOMORPHISM for connected constant-size H parameterized by the order ℓ of the largest connected component admits an $(n + m, \ell^2)$ -Turing kernel, as each oracle call is on a connected component (which is of size at most $O(\ell^2)$) of the input graph. We present a strict Turing kernelization for H -SI for connected constant-size H parameterized by maximum degree Δ .

Proposition 4 (\star). H -SUBGRAPH ISOMORPHISM for connected H with $c = |V(H)|$ parameterized by maximum degree Δ admits a strict $(n \cdot \Delta \cdot (\Delta - 1)^{\lfloor c/2 \rfloor}, \Delta \cdot (\Delta - 1)^{\lfloor c/2 \rfloor})$ -Turing kernel.

Running-time Related Strict Kernelization. For NP-hard problems, it is well-known that a decidable problem is fixed-parameter tractable if and only if it admits a kernel [12]. In the proof of the *only if*-statement, one derives a kernel of size only depending on the running time of a fixed-parameter algorithm solving the problem in question. We adapt this idea to derive a strict kernel where the running time and size admit such running time dependencies.

Theorem 3 (\star). Let L be a parameterized problem admitting an algorithm solving each instance (x, k) in $k^c \cdot |x|$ time for some constant $c > 0$. Then for every $\varepsilon > 0$, each instance (x, k) admits a strict $(|x|^{1+c/(1+\varepsilon)}, k^{1+\varepsilon})$ -kernel.

NWT and TC are both solvable in $O(k^2 \cdot n)$ time (k being the order ℓ of the largest connected component, the maximum degree Δ , or the degeneracy d [9]). Together with Theorem 3 gives several kernelization results for NWT and TC, for instance, with $\varepsilon = 2$:

Corollary 1. NWT admits a strict $(n^{5/3}, d^3)$ -kernel when parameterized by the degeneracy d of the input graph.

Note that the presented kernel is a strict (n^α, d^β) -kernel with $\alpha = 5/3$ and $\beta = 3$. As $\alpha \cdot \beta = 5$ in this case, there is a gap between the above kernel and the lower bound of $\alpha \cdot \beta \geq 3$ in Theorem 2(iii). Future work could be to close this gap.

5 Conclusion

We provided the first conceptual analysis of strict kernelization lower bounds for problems solvable in polynomial time. To this end, we used and (slightly) enhanced the parameter diminisher framework [8, 13]. Our results for NEGATIVE WEIGHT TRIANGLE and TRIANGLE COLLECTION rely on the APSP-conjecture and SETH, but these assumptions can be replaced with any running-time lower

bound known for the problem at hand. Indeed the framework is not difficult to apply and we believe that developing special techniques to design diminishers is a fruitful line of further research.

We point out that the framework excludes certain *trade-offs* between kernel size and running time: the smaller the running time of the diminisher, the larger the size of the strict kernel that can be excluded. However, the framework in its current form cannot be used to exclude the existence of *any* strict kernel of polynomial size in even linear time.

In this work, we only considered parameters that we call *dispersed* parameters, defined as follows. Let G be an instance of a graph problem L , and let G_1, G_2, \dots, G_p be its connected components, where $p \geq 1$. A parameter k of G is *dispersed* if $k(G)$ (i.e. the value of the parameter k in the graph G) is equal to $k(G_i)$ for *at least one* connected subgraph G_i of G . Otherwise, if $k(G)$ is larger than $k(G_i)$ for *every* connected subgraph G_i of G , then we call k an *aggregated* parameter. In our opinion, it is of independent interest to apply the (strong) diminisher framework to graph problems with aggregated parameters. Note that such a classification into dispersed and aggregated parameters has not been studied previously.

We close with one concrete challenge: Is there a (strong) diminisher for NWT or TC with respect to the (aggregated) parameter feedback vertex number? Note that the disjoint union operation that we use in all our diminishers in Sect. 3 can increase this parameter.

Acknowledgement. We thank Holger Dell (Saarland University) for fruitful discussion on Sect. 2 and Rolf Niedermeier for discussions leading to this work.

References

1. Abboud, A., Grandoni, F., Williams, V.V.: Subcubic equivalences between graph centrality problems, APSP and diameter. In: Proceedings of 26th SODA, pp. 1681–1697. SIAM (2015)
2. Abboud, A., Williams, V.V.: Popular conjectures imply strong lower bounds for dynamic problems. In: Proceedings of 55th FOCS, pp. 434–443. IEEE Computer Society (2014)
3. Abboud, A., Williams, V.V., Wang, J.R.: Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In: Proceedings of 27th SODA, pp. 377–391. SIAM (2016)
4. Abboud, A., Williams, V.V., Yu, H.: Matching triangles and basing hardness on an extremely popular conjecture. In: Proceedings of 47th STOC, pp. 41–50. ACM (2015)
5. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. *J. Comput. System Sci.* **75**(8), 423–434 (2009)
6. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.* **28**(1), 277–305 (2014)
7. Bringmann, K.: Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In: Proceedings of 55th FOCS, pp. 661–670 (2014)

8. Chen, Y., Flum, J., Müller, M.: Lower bounds for kernelizations and other preprocessing procedures. *Theory Comput. Syst.* **48**(4), 803–839 (2011)
9. Chiba, N., Nishizeki, T.: Arboricity and subgraph listing algorithms. *SIAM J. Comput.* **14**(1), 210–223 (1985)
10. Cygan, M., Fomin, F.V., Kowalik, L., Lokshantov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: *Parameterized Algorithms*. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-21275-3>
11. Theory, G.: *Graph Theory*. Graduate Texts in Mathematics, vol. 173. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-662-53622-3>
12. Downey, R.G., Fellows, M.R.: *Fundamentals of Parameterized Complexity*. Springer, London (2013). <https://doi.org/10.1007/978-1-4471-5559-1>
13. Fernau, H., Fluschnik, T., Hermelin, D., Krebs, A., Molter, H., Niedermeier, R.: Diminishable parameterized problems and strict polynomial kernelization (2016). CoRR abs/1611.03739
14. Fluschnik, T., Komusiewicz, C., Mertzios, G.B., Nichterlein, A., Niedermeier, R., Talmon, N.: When can graph hyperbolicity be computed in linear time? *Algorithms and Data Structures*. LNCS, vol. 10389, pp. 397–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62127-2_34
15. Fluschnik, T., Mertzios, G.B., Nichterlein, A.: Kernelization lower bounds for finding constant size subgraphs. CoRR abs/1710.07601 (2017). <http://arxiv.org/abs/1710.07601>
16. Fomin, F.V., Lokshantov, D., Pilipczuk, M., Saurabh, S., Wrochna, M.: Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. In: *Proceedings of 28th SODA*, pp. 1419–1432 (2017)
17. Fomin, F.V., Saurabh, S.: Kernelization methods for fixed-parameter tractability. In: *Tractability: Practical Approaches to Hard Problems*, pp. 260–282. Cambridge University Press (2014)
18. Fortnow, L., Santhanam, R.: Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. System Sci.* **77**(1), 91–106 (2011)
19. Gajentaan, A., Overmars, M.H.: On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.* **5**, 165–185 (1995)
20. Giannopoulou, A.C., Mertzios, G.B., Niedermeier, R.: Polynomial fixed-parameter algorithms: a case study for longest path on interval graphs. *Theor. Comput. Sci.* **689**, 67–95 (2017)
21. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. *ACM SIGACT News* **38**(1), 31–45 (2007)
22. Impagliazzo, R., Paturi, R.: On the complexity of k -SAT. *J. Comput. Syst. Sci.* **62**(2), 367–375 (2001)
23. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* **63**(4), 512–530 (2001)
24. Iwata, Y.: Linear-time kernelization for feedback vertex set. In: *Proceedings of 44th ICALP*. LIPIcs, vol. 80, pp. 68:1–68:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
25. Komusiewicz, C., Niedermeier, R.: New races in parameterized algorithmics. In: *Rovan, B., Sassone, V., Widmayer, P. (eds.) MFCS 2012*. LNCS, vol. 7464, pp. 19–30. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32589-2_2

26. Kratsch, S.: Recent developments in kernelization: a survey. *Bulletin of EATCS* **113**, 58–97 (2014)
27. Lokshantov, D., Misra, N., Saurabh, S.: Kernelization – preprocessing with a guarantee. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) *The Multivariate Algorithmic Revolution and Beyond*. LNCS, vol. 7370, pp. 129–161. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30891-8_10
28. Matula, D.W., Beck, L.L.: Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM* **30**(3), 417–427 (1983)
29. Mertzios, G.B., Nichterlein, A., Niedermeier, R.: The power of data reduction for matching. In: *Proceedings of 42nd MFCS*. LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017, to appear)
30. Williams, V.V., Williams, R.: Subcubic equivalences between path, matrix and triangle problems. In: *Proceedings of 51st FOCS*, pp. 645–654. IEEE Computer Society (2010)
31. Weihe, K.: Covering trains by stations or the power of data reduction. In: *Proceedings of 1st ALEX* No publisher, pp. 1–8 (1998)



On General Sum Approximations of Irrational Numbers

Ivan Georgiev^{1(✉)}, Lars Kristiansen^{2,3}, and Frank Stephan⁴

¹ Department of Mathematics and Physics, Faculty of Natural Sciences, University “Prof. d-r Asen Zlatarov”, 8010 Burgas, Bulgaria

ivandg@yahoo.com

² Department of Mathematics, University of Oslo, Oslo, Norway

larsk@math.uio.no

³ Department of Informatics, University of Oslo, Oslo, Norway

⁴ Department of Mathematics and School of Computing, National University of Singapore, Singapore 119076, Republic of Singapore
fstephan@comp.nus.edu.sg

1 Introduction and Basic Definitions

There are numerous ways to represent real numbers. We may use, e.g., Cauchy sequences, Dedekind cuts, numerical base-10 expansions, numerical base-2 expansions and continued fractions. If we work with full Turing computability, all these representations yield the same class of real numbers. If we work with some restricted notion of computability, e.g., polynomial time computability or primitive recursive computability, they do not. This phenomenon has been investigated over the last seven decades by Specker [13], Mostowski [8], Lehman [10], Ko [3, 4], Labhalla and Lombardi [9], Georgiev [1], Kristiansen [5, 6] and quite a few more. Georgiev et al. [2] is an extended version of the current paper.

Irrational numbers can be represented by infinite sums. Sum approximations from below and above were introduced in Kristiansen [5] and studied further in Kristiansen [6]. Every irrational number α between 0 and 1 can be uniquely written as an infinite sum of the form

$$\alpha = 0 + \frac{D_1}{b^{k_1}} + \frac{D_2}{b^{k_2}} + \frac{D_3}{b^{k_3}} + \dots$$

where

- $b \in \mathbb{N} \setminus \{0, 1\}$ and $D_i \in \{1, \dots, b - 1\}$ (note that $D_i \neq 0$ for all i)
- $k_i \in \mathbb{N} \setminus \{0\}$ and $k_i < k_{i+1}$.

Let $\hat{A}_b^\alpha(i) = D_i b^{-k_i}$ for $i > 0$ (and let $\hat{A}_b^\alpha(0) = 0$). The rational number $\sum_{i=1}^n \hat{A}_b^\alpha(i)$ is an approximation of α that lies below α , and we will say that

I. Georgiev has been supported by the Bulgarian National Science Fund through the project “Models of computability”, DN-02-16/19.12.2016.

F. Stephan has been supported in part by the Singapore Ministry of Education Academic Research Fund grant MOE2016-T2-1-019/R146-000-234-112.

the function \hat{A}_b^α is the *base- b sum approximation from below* of α . The *base- b sum approximation from above* of α is a symmetric function \check{A}_b^α such that $1 - \sum_{i=1}^n \hat{A}_b^\alpha(i)$ is an approximation of α that lies above α (and we have $\sum_{i=1}^\infty \hat{A}_b^\alpha(i) + \sum_{i=1}^\infty \check{A}_b^\alpha(i) = 1$). Let \mathcal{S} be any class of subrecursive functions which is closed under primitive recursive operations. Furthermore, let $\mathcal{S}_{b\uparrow}$ denote the set of irrational numbers that have a base- b sum approximation from below in \mathcal{S} , and let $\mathcal{S}_{b\downarrow}$ denote the set of irrational numbers that have a base- b sum approximation from above in \mathcal{S} . It is proved in [6] that $\mathcal{S}_{b\uparrow}$ and $\mathcal{S}_{b\downarrow}$ are incomparable classes, that is, $\mathcal{S}_{b\uparrow} \not\subseteq \mathcal{S}_{b\downarrow}$ and $\mathcal{S}_{b\downarrow} \not\subseteq \mathcal{S}_{b\uparrow}$. Another interesting result proved in [6] is that $\mathcal{S}_{a\downarrow} \subseteq \mathcal{S}_{b\downarrow}$ iff $\mathcal{S}_{a\uparrow} \subseteq \mathcal{S}_{b\uparrow}$ iff every prime factor of b is a prime factor of a .

In this paper we prove some results on *general sum approximations*. The *general sum approximation from below* of α is the function $\check{G}^\alpha : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{Q}$ defined by $\check{G}^\alpha(b, n) = \hat{A}_b^\alpha(n)$; let $\check{G}^\alpha(b, n) = 0$ if $b < 2$. The *general sum approximation from above* of α is the function $\hat{G}^\alpha : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{Q}$ defined by $\hat{G}^\alpha(b, n) = \check{A}_b^\alpha(n)$; let $\hat{G}^\alpha(b, n) = 0$ if $b < 2$. Let \mathcal{S} be any class of subrecursive functions which is closed under primitive recursive operations. Furthermore, let $\mathcal{S}_{g\uparrow}$ denote the set of irrational numbers that have a general sum approximation from below in \mathcal{S} , and let $\mathcal{S}_{g\downarrow}$ denote the set of irrational numbers that have a general sum approximation from above in \mathcal{S} .

It was proved in [5] that $\mathcal{S}_{g\uparrow} \cap \mathcal{S}_{g\downarrow}$ contains exactly the irrational numbers that have a continued fraction in the class \mathcal{S} . In this paper we prove that $\mathcal{S}_{g\uparrow} \neq \mathcal{S}_{g\downarrow}$. Moreover, we prove that

$$\mathcal{S}_{g\downarrow} \neq \bigcap_{b=2}^\infty \mathcal{S}_{b\downarrow} \quad \text{and} \quad \mathcal{S}_{g\uparrow} \neq \bigcap_{b=2}^\infty \mathcal{S}_{b\uparrow}.$$

Some might find it interesting (at least the authors do) that we manage to complete all our proof without resorting to the standard computability-theoretic machinery involving enumerations, universal functions, diagonalizations, and so on. We prove our results by providing natural irrational numbers (the numbers are natural in the sense that they have neat and transparent definitions).

2 Preliminaries

We will restrict our attention to real numbers between 0 and 1.

A *base* is a natural number strictly greater than 1, and a *base- b digit* is a natural number in the set $\{0, 1, \dots, b - 1\}$.

Let b be a base, and let D_1, \dots, D_n be base- b digits. We will use $(0.D_1D_2 \dots D_n)_b$ to denote the rational number $\sum_{i=1}^n D_i b^{-i}$.

Let D_1, D_2, \dots be an infinite sequence of base- b digits. We say that $(0.D_1D_2 \dots)_b$ is the *base- b expansion of the real number α* if for all $n \geq 1$ we have

$$(0.D_1D_2 \dots D_n)_b \leq \alpha < (0.D_1D_2 \dots D_n)_b + b^{-n}.$$

Every real number α has a unique base- b expansion (note the strict inequality).

When $\alpha = (0.D_1D_2 \dots D_n)_b$ for some n with $D_n \neq 0$, we say that α has a *finite base- b expansion* of length n . Otherwise, we say that α has an *infinite base- b expansion*, and this infinite base- b expansion is periodic iff α is rational. More concretely, if $\alpha = cd^{-1}$ for non-zero relatively prime $c, d \in \mathbb{N}$, then the base- b expansion of α is of the form $0.D_1 \dots D_s(D_{s+1} \dots D_t)^\omega$ which we use as shorthand for the infinite sequence $0.D_1 \dots D_sD_{s+1} \dots D_tD_{s+1} \dots D_tD_{s+1} \dots D_t \dots$. The number s is the largest natural number such that p^s divides d for some prime factor p of b . The length of the period $t - s$ is the multiplicative order of b modulo d_1 where d_1 is the largest divisor of d relatively prime with b . It follows straightforwardly that $t < d$. Of course, α has a finite base- b expansion iff $d_1 = 1$, that is, iff every prime factor of d is a prime factor of b .

We assume the reader is familiar with subrecursion theory and subrecursive functions. An introduction to the subject can be found in [11] or [12].

A function ϕ is *elementary* in a function ψ , written $\phi \leq_E \psi$, if ϕ can be generated from the initial functions $\psi, 2^x, \max, 0, S$ (successor), I_i^n (projections) by composition and bounded primitive recursion. A function ϕ is *elementary* if $\phi \leq_E 0$. A function ϕ is *primitive recursive* in a function ψ , written $\phi \leq_{PR} \psi$, if ϕ can be generated from the initial functions by composition and (unbounded) primitive recursion. A function ϕ is *primitive recursive* if $\phi \leq_{PR} 0$.

Subrecursive functions in general, and elementary functions in particular, are formally functions over natural numbers (\mathbb{N}). We assume some coding of integers (\mathbb{Z}) and rational numbers (\mathbb{Q}) into the natural numbers. We consider such a coding to be trivial. Therefore we allow for subrecursive functions from rational numbers into natural numbers, from pairs of rational numbers into rational numbers, etc., with no further comment. Uniform systems for coding finite sequences of natural numbers are available inside the class of elementary functions. Hence, for any reasonable coding, basic operations on rational numbers – like e.g. addition, subtraction and multiplication – will obviously be elementary. It is also obvious that there is an elementary function $\psi(q, i, b)$ that yields the i^{th} digit in the base- b expansion of the rational number q .

A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is *honest* if it is monotonically increasing ($f(x) \leq f(x + 1)$), dominates 2^x ($f(x) \geq 2^x$) and has elementary graph (the relation $f(x) = y$ is elementary).

A class of functions \mathcal{S} is *subrecursive class* if \mathcal{S} is an efficiently enumerable class of computable total functions. For any subrecursive class \mathcal{S} there exists an honest function f such that $f \notin \mathcal{S}$ (see Sect. 8 of [5] for more details).

More on elementary functions, primitive recursive functions and honest functions can be found in Sect. 2 of [5, 7].

3 Irrational Numbers with Interesting Properties

Definition 1. Let P_i denote the i^{th} prime ($P_0 = 2, P_1 = 3, \dots$). We define the auxiliary function g by

$$g(0) = 1 \quad \text{and} \quad g(j + 1) = P_j^{2(j+2)(g(j)+1)^3}.$$

For any honest function f and any $n \in \mathbb{N}$, we define the rational number α_n^f and the irrational number α^f by

$$\alpha_n^f = \sum_{i=0}^n P_i^{-h(i)} \quad \text{and} \quad \alpha^f = \lim_{n \rightarrow \infty} \alpha_n^f$$

where $h(i) = g(f(i) + i)$ (for any $i \in \mathbb{N}$).

It is easy to see that both g and h are strictly increasing honest functions. Moreover, we have

$$P_n^{2(n+2)(h(n)+1)^3} < h(n+1) \tag{1}$$

for any $n \in \mathbb{N}$. Thus the function h possesses a growth property which f might not possess. We will need this property. This explains why we introduce the function g in the definition of α^f .

When f is a fixed honest function, we abbreviate α_j^f and α^f to α_j and α , respectively.

The next lemma is easily proven using the preliminaries on base- b expansions.

Lemma 2. For any $j \in \mathbb{N}$ and any base b , we have

- (i) if P_i divides b for all $i \leq j$, then α_j has a finite base- b expansion of length $h(j)$
- (ii) if P_i does not divide b for some $i \leq j$, then α_j has an infinite (periodic) base- b expansion.

Lemma 3. Let

- b be any base, and let $j \in \mathbb{N}$ be such that $P_j > b$
- $(0.D_1D_2\dots)_b$ be the base- b expansion of α_j
- $(0.\dot{D}_1\dot{D}_2\dots)_b$ be the base- b expansion of α_{j+1}
- $M = M(j) = P_j^{(j+1)h(j)}$ and $M' = M'(j) = h(j+1)$.

Then

- (i) there are no more than M consecutive zeros in the base- b expansion of α_j , that is, for any $k \in \mathbb{N} \setminus \{0\}$ there exists $i \in \mathbb{N}$ such that

$$k \leq i < k + M \quad \text{and} \quad D_i \neq 0$$

- (ii) the first $M' - M$ digits of the base- b expansions of α_j and α_{j+1} coincide, that is

$$i \leq M' - M \quad \Rightarrow \quad D_i = \dot{D}_i$$

and moreover, these digits also coincide with the corresponding digits of the base- b expansion of α .

Proof. By Lemma 2 (ii), α_j has an infinite periodic base- b expansion of the form $0.D_1 \dots D_s(D_{s+1} \dots D_t)^\omega$ with $s < t$. Using the preliminaries on base- b expansions we obtain

$$t - s \leq t < \prod_{i=0}^j P_i^{h(i)} \leq P_j^{(j+1)h(j)} = M. \tag{2}$$

Thus (i) holds since every M consecutive digits of α_j contain all the digits D_{s+1}, \dots, D_t of at least one period.

We turn to the proof of (ii). We have

$$\alpha_j < \alpha_{j+1} = \alpha_j + P_{j+1}^{-h(j+1)} \leq \alpha_j + b^{-M'} \tag{3}$$

since $b^{M'} < P_j^{M'} = P_j^{h(j+1)} < P_{j+1}^{h(j+1)}$. At least one digit in the period $D_{s+1} \dots D_t$ is different from $b - 1$, and the length of the period is $t - s$. Thus (3) entails

$$D_i = \dot{D}_i \text{ for any } i \leq M' - (t - s). \tag{4}$$

It follows from (2) and (4) that the first $M' - M$ digits of the base- b expansions of α_j and α_{j+1} coincide. Moreover, since $M'(j)$ is strictly increasing in j ,

$$\alpha_j < \alpha_{j+k} \leq \alpha_j + \sum_{i < k} b^{-M'(j+i)} \leq \alpha_j + b^{-M'(j)+1}$$

for any $k \geq 1$. Letting $k \rightarrow \infty$ we obtain as above that the first $M' - M$ digits of α_j and α coincide. □

Theorem 4. *Let f be any honest function, and let b be any base. The function $\hat{A}_b^{\alpha^f}$ is elementary.*

Proof. Fix the least m such that $P_m > b$. We will use the functions M and M' from Lemma 3. We will argue that we can compute the rational number $\hat{A}_b^\alpha(n)$ elementarily in n when $n \geq M(m)$. Note that $M(m)$ is a fixed number (it does not depend on n). Thus, we can compute $\hat{A}_b^\alpha(n)$ by a trivial algorithm when $n < M(m)$ (use a huge table).

Assume $n \geq M(m)$. We will now give an algorithm for computing $\hat{A}_b^\alpha(n)$ elementarily in n .

Step 1 of the algorithm: Compute (the unique) j such that

$$M(j) \leq n < M(j + 1) \tag{5}$$

(end of Step 1).

Step 1 is a computation elementary in n since M has elementary graph. So is Step 2 as M' also has elementary graph.

Step 2 of the algorithm: Check if the following relation holds:

$$n^2 + 1 < M'(j) - M(j). \tag{6}$$

If it holds, carry out Step 3A, otherwise, carry out Step 3B (*end of Step 2*).

Step 3A of the algorithm: Compute α_j . Then compute base- b digits D_1, \dots, D_{n^2+1} such that

$$(0.D_1D_2 \dots D_{n^2+1})_b \leq \alpha_j < (0.D_1D_2 \dots D_{n^2+1})_b + b^{-(n^2+1)}.$$

Find k such that D_k is the n^{th} digit different from 0 in the sequence D_1, \dots, D_{n^2+1} . Give the output $D_k b^{-k}$ (*end of Step 3A*).

Recall that $\alpha_j = \sum_{i=0}^j P_i^{-h(i)}$. We can compute α_j elementarily in n since $h(0), h(1), \dots, h(j) < M(j) \leq n$ and h is honest. Thus, we can also compute the base- b digits $D_1, D_2, \dots, D_{n^2+1}$ elementarily in n . In order to prove that our algorithm is correct, we must verify that

- (A) at least n of the digits $D_1, D_2, \dots, D_{n^2+1}$ are different from 0, and
- (B) $D_1, D_2, \dots, D_{n^2+1}$ coincide with the first $n^2 + 1$ digits of α .

By Lemma 3 (i) the sequence $D_{kM(j)+1}, D_{kM(j)+2}, \dots, D_{(k+1)M(j)}$ contains at least one non-zero digit (for any $k \in \mathbb{N}$). Thus, (A) holds since $n \geq M(j)$. Using (6) and Lemma 3 (ii) we see that (B) also holds. This proves that the output $D_k b^{-k} = \hat{A}_b^\alpha(n)$.

Step 3B of the algorithm: Compute α_{j+1} and $M(j+1)$. Then proceed as in Step 3A with α_{j+1} in place of α_j and $nM(j+1)$ in place of n^2 (*end of Step 3B*).

Step 3B is only executed when $M'(j) - M(j) \leq n^2 + 1$. Thus, we have $M'(j) = h(j+1) \leq n^2 + n + 1$. This entails that we can compute $h(j+1)$ – and also α_{j+1} and $M(j+1)$ – elementarily in n . It follows easily from (1) that

$$M(j+1)^2 + M(j+1) + 1 < M'(j+1)$$

which together with (5) imply

$$nM(j+1) + 1 < M'(j+1) - M(j+1).$$

As in Step 3A, there will be at least n non-zero digits among the first $nM(j+1)$ digits of α_{j+1} . Moreover, the first $nM(j+1)$ digits of α_{j+1} coincide with the corresponding digits of α . □

Theorem 5. *Let f be any honest function. We have $f \leq_{PR} \hat{G}^{\alpha^f}$ (f is primitive recursive in \hat{G}^{α^f}).*

Proof. Fix $n \in \mathbb{N}$, and let b be the base $b = \prod_{i=0}^n P_i$. By Lemma (2) (i), α_n has a finite base- b expansion of length $h(n)$. By the definition of α , we have

$$\alpha = \alpha_n + P_{n+1}^{-h(n+1)} + P_{n+2}^{-h(n+2)} + \dots$$

It follows that for any $j > h(n)$

$$\hat{G}^\alpha(b, j) \leq P_{n+1}^{-h(n+1)} + P_{n+2}^{-h(n+2)} + \dots,$$

which easily implies $\hat{G}^\alpha(b, j) \leq P_{n+1}^{-h(n+1)+1}$ (use that h is strictly increasing). Hence we also have $(\hat{G}^\alpha(b, j))^{-1} \geq P_{n+1}^{h(n+1)-1} > h(n+1) - 1$ for any $j > h(n)$.

The considerations above show that we can compute $h(n+1)$ by the following algorithm:

- assume that $h(n)$ is computed;
- compute $b = \prod_{i=0}^n P_i$;
- search for y such that $y < (\hat{G}^\alpha(b, h(n) + 1))^{-1} + 1$ and $h(n+1) = y$;
- give the output y .

This algorithm is primitive recursive in \hat{G}^α : The computation of b is an elementary computation. The relation $h(x) = y$ is elementary, and thus the search for y is elementary in $h(n)$ and \hat{G}^α . This proves that h is primitive recursive in \hat{G}^α . But then f will also be primitive recursive in \hat{G}^α as the graph of f is elementary and $f(n) \leq h(n)$ (for any $n \in \mathbb{N}$). This proves that $f \leq_{PR} \hat{G}^\alpha$. □

Theorem 6. *Let f be any honest function. There exists an elementary function $\check{T} : \mathbb{Q} \rightarrow \mathbb{Q}$ such that (i) $\check{T}(q) = 0$ if $q < \alpha^f$ and (ii) $q > \check{T}(q) > \alpha^f$ if $q > \alpha^f$.*

Proof. In addition to the sequence α_j we need the sequence β_j given by

$$\beta_0 = P_0^{-h(0)+1} = 2^{-h(0)+1} \quad \text{and} \quad \beta_{j+1} = \alpha_j + P_{j+1}^{-h(j+1)+1}.$$

Observe that we have $\alpha < \beta_j$ for all $j \in \mathbb{N}$.

Now we will explain an algorithm that computes a function \check{T} with the properties (i) and (ii).

Step 1 of the algorithm: The input is the rational number q . We can w.l.o.g. assume that $0 < q < 1$. Pick any $m', n \in \mathbb{N}$ such that $q = m'n^{-1}$ and $n \geq h(0)$. Find $m \in \mathbb{N}$ such that $q = m(P_0P_1 \dots P_n)^{-n}$, and compute the base b such that $b = \prod_{i=0}^n P_i$ (end of Step 1).

The rational number q has a finite base- b expansion of length s where $s \leq n$. Moreover, the rational numbers $\alpha_0, \alpha_1, \dots, \alpha_n$ and $\beta_0, \beta_1, \dots, \beta_n$ also have finite base- b expansions.

Step 2 of the algorithm: Compute (the unique) natural number $j < n$ such that

$$h(j) \leq n < h(j+1).$$

Furthermore, compute $\alpha_0, \alpha_1, \dots, \alpha_j$ and $\beta_0, \beta_1, \dots, \beta_j$ (end of Step 2).

All the numbers $h(0), h(1), \dots, h(j)$ are less than or equal to n , and h has elementary graph. This entails that Step 2 is elementary in n (and also in q).

Step 3 of the algorithm: If $q \leq \alpha_k$ for some $k \leq j$, give the output 0 and terminate. If $\beta_k < q$ for some $k \leq j$, give the output β_k and terminate (end of Step 3).

Step 3 is obviously elementary in q and gives the correct output.

If the algorithm has not yet terminated, we have $\alpha_j < q \leq \beta_j$. Now

$$q \leq \beta_{j+1} \iff q - \alpha_j \leq P_{j+1}^{-h(j+1)+1} \iff P_{j+1}^{h(j+1)} \leq (q - \alpha_j)^{-1} P_{j+1}.$$

We have determined α_j , and h is an honest function. This makes it possible to check elementarily if $q \leq \beta_{j+1}$: Search for $y < (q - \alpha_j)^{-1} P_{j+1}$ such that $h(j + 1) = y$. If no such y exists, we have $q > \beta_{j+1}$. If such an y exists, we have $q \leq \beta_{j+1}$ iff $P_{j+1}^y \leq (q - \alpha_j)^{-1} P_{j+1}$.

Step 4 of the algorithm: Search for $y < (q - \alpha_j)^{-1} P_{j+1}$ such that $y = h(j + 1)$. If the search is successful and $P_{j+1}^y \leq (q - \alpha_j)^{-1} P_{j+1}$, go to Step 5, otherwise go to Step 6B (*end of Step 4*).

Clearly, Step 4 is elementary in q . If $q \leq \beta_{j+1}$, the next step is Step 5 (and we have computed $y = h(j + 1)$). If $\beta_{j+1} < q$, the next step is Step 6B.

Step 5 of the algorithm: Compute α_{j+1} and β_{j+1} . If $q \leq \alpha_{j+1}$, give the output 0 and terminate. If $\alpha_{j+1} < q$, search for $z < (q - \alpha_{j+1})^{-1} P_{j+2}$ such that $z = h(j + 2)$. If the search is successful and $P_{j+2}^z \leq (q - \alpha_{j+1})^{-1} P_{j+2}$, give the output 0 and terminate, otherwise, go to Step 6A (*end of Step 5*).

Step 5 is elementary in q since we have computed $h(j + 1)$ in Step 4. If the algorithm terminates because $q \leq \alpha_{j+1}$, we obviously have $q < \alpha$ and the output is correct. If $q > \alpha_{j+1}$, the algorithm will not proceed to Step 6A iff $\alpha_{j+1} < q \leq \beta_{j+2}$. So assume that $\alpha_{j+1} < q \leq \beta_{j+2}$. It is not hard to show that the first $h(j + 1)$ digits of $\alpha_{j+1}, \alpha, \beta_{j+2}$ coincide, and moreover, $h(j + 1) > n \geq s$ (recall that s is the length of the base- b expansion of q). Thus, we have $q < \alpha$, and the algorithm gives the correct output, namely 0. If the algorithm proceeds with Step 6A, we have $\beta_{j+2} < q$.

Step 6A of the algorithm: Compute the least t such that $b^t > (q - \alpha_{j+1})^{-1}$. Search for $u < (q - b^{-t} - \alpha_{j+1})^{-1} P_{j+2}$ such that $u = h(j + 2)$. If the search is successful and $P_{j+2}^u < (q - b^{-t} - \alpha_{j+1})^{-1} P_{j+2}$, give the output β_{j+2} and terminate, otherwise, give the output $q - b^{-t}$ and terminate (*end of Step 6A*).

It is easy to see that Step 6A is elementary in q : we can compute t elementarily in q, b and α_{j+1} (and we have already computed b and α_{j+1} elementarily in q). When the execution of the step starts, we have $\beta_{j+2} < q$ (thus, β_{j+2} will be a correct output, but we do not yet know if we will be able to compute β_{j+2}). If the search for u is successful, we have $u = h(j + 2)$. Then, we can compute β_{j+2} elementarily in u , and give β_{j+2} as output. We also know that the search for u is successful iff $q - b^{-t} < \beta_{j+2}$. Thus, if the search for u is not successful, we have $\alpha < \beta_{j+2} \leq q - b^{-t} < q$, and we can give the correct output $q - b^{-t}$.

Step 6B of the algorithm: Exactly the same as 6A, but replace $j + 1$ and $j + 2$ by j and $j + 1$, respectively (*end of Step 6B*).

The argument for correctness of Step 6B is the same as for Step 6A, just replace $j + 1$ and $j + 2$ by j and $j + 1$, respectively, and note that we have $\beta_{j+1} < q$ when the execution of the step starts. \square

Definition 7. A function $D : \mathbb{Q} \rightarrow \{0, 1\}$ is a Dedekind cut of the real number β when $D(q) = 0$ iff $q < \beta$.

Corollary 8. *Let f be any honest function. The Dedekind cut of the real number α^f is elementary.*

Proof. By Theorem 6 there is an elementary function \tilde{T} such that $\tilde{T}(q) = 0$ iff $q < \alpha^f$. Let $D(q) = 0$ if $\tilde{T}(q) = 0$, and let $D(q) = 1$ if $\tilde{T}(q) \neq 0$. The function D is elementary since \tilde{T} is elementary. Moreover, D is the Dedekind cut of α^f . \square

4 Main Results

Theorem 9. *For any subrecursive class \mathcal{S} that is closed under primitive recursive operations, we have*

$$(i) \mathcal{S}_{g\downarrow} \subset \bigcap_b \mathcal{S}_{b\downarrow} \quad \text{and} \quad (ii) \mathcal{S}_{g\uparrow} \subset \bigcap_b \mathcal{S}_{b\uparrow}.$$

Proof. The inclusion $\mathcal{S}_{g\uparrow} \subseteq \bigcap_b \mathcal{S}_{b\uparrow}$ is trivial. Pick an honest function f such that $f \notin \mathcal{S}$. By Theorem 4, we have $\alpha^f \in \bigcap_b \mathcal{S}_{b\uparrow}$. By Theorem 5, we have $\alpha^f \notin \mathcal{S}_{g\uparrow}$. This proves that $\mathcal{S}_{g\uparrow} \subset \bigcap_b \mathcal{S}_{b\uparrow}$. The proof of (i) is symmetric. \square

Definition 10. *A function $\hat{T} : \mathbb{Q} \rightarrow \mathbb{Q}$ is a trace function from below for the irrational number α when we have $q < \hat{T}(q) < \alpha$ for any $q < \alpha$. A function $\tilde{T} : \mathbb{Q} \rightarrow \mathbb{Q}$ is a trace function from above for the irrational number α when we have $\alpha < \tilde{T}(q) < q$ for any $q > \alpha$. A function $T : \mathbb{Q} \rightarrow \mathbb{Q}$ is a trace function for the irrational number α when we have $|\alpha - q| > |\alpha - T(q)|$ for any q .*

For any subrecursive class \mathcal{S} , let \mathcal{S}_D denote the set of irrational numbers that have a Dedekind cut in \mathcal{S} ; let $\mathcal{S}_{T\uparrow}$ denote the set of irrational numbers that have a trace function from below in \mathcal{S} ; let $\mathcal{S}_{T\downarrow}$ denote the set of irrational numbers that have a trace function from above in \mathcal{S} ; let \mathcal{S}_T denote the set of irrational numbers that have a trace function in \mathcal{S} ; let $\mathcal{S}_{[\]}$ denote the set of irrational numbers that have a continued fraction in \mathcal{S} .

It is proved in [5] that we have $\mathcal{S}_{g\downarrow} \cap \mathcal{S}_{g\uparrow} = \mathcal{S}_T = \mathcal{S}_{[\]}$ for any \mathcal{S} closed under primitive recursive operations. It is conjectured in [5] that $\mathcal{S}_{g\downarrow} \neq \mathcal{S}_{g\uparrow}$. Theorem 12 shows that this conjecture holds. The following theorem will be used as a lemma in the proof of Theorem 12. Its proof can be found in Georgiev et al. [2].

Theorem 11. *For any subrecursive class \mathcal{S} that is closed under primitive recursive operations, we have*

$$(i) \mathcal{S}_{T\uparrow} \cap \mathcal{S}_D = \mathcal{S}_{g\uparrow} \quad \text{and} \quad (ii) \mathcal{S}_{T\downarrow} \cap \mathcal{S}_D = \mathcal{S}_{g\downarrow}.$$

Theorem 12. *For any subrecursive class \mathcal{S} that is closed under primitive recursive operations, there exist irrational numbers α and β such that*

$$(i) \alpha \in \mathcal{S}_{g\downarrow} \setminus \mathcal{S}_{g\uparrow} \quad \text{and} \quad (ii) \beta \in \mathcal{S}_{g\uparrow} \setminus \mathcal{S}_{g\downarrow}.$$

Proof. Pick an honest function f such that $f \notin \mathcal{S}$. We have $\alpha^f \in \mathcal{S}_{T\downarrow}$ by Theorem 6, and we have $\alpha^f \in \mathcal{S}_D$ by Corollary 8. By Theorem 11, we have $\alpha^f \in \mathcal{S}_{g\downarrow}$. By Theorem 5, we have $\alpha^f \notin \mathcal{S}_{g\uparrow}$. This proves (i). The proof of (ii) is symmetric. \square

References

1. Georgiev, I.: Continued fractions of primitive recursive real numbers. *Math. Log.* **Q.** **61**, 288–306 (2015)
2. Georgiev, I., Kristiansen, L. and Stephan, F.: Subrecursive Approximations of Irrational Numbers by Variable Base Sums (2018). ArXiv e-prints, [arXiv:1804.05330](https://arxiv.org/abs/1804.05330) [math.LO]
3. Ko, K.: On the definitions of some complexity classes of real numbers. *Math. Syst. Theory* **16**, 95–109 (1983)
4. Ko, K.: On the continued fraction representation of computable real numbers. *Theor. Comput. Sci.* **47**, 299–313 (1986)
5. Kristiansen, L.: On subrecursive representability of irrational numbers. *Computability* **6**, 249–276 (2017)
6. Kristiansen, L.: On subrecursive representability of irrational numbers, part II. *Computability* **6**, 249–276 (2018). <https://doi.org/10.3233/COM-170081>
7. Kristiansen, L., Schlage-Puchta, J.-C., Weiermann, A.: Streamlined subrecursive degree theory. *Ann. Pure Appl. Log.* **163**, 698–716 (2012)
8. Mostowski, A.: On computable sequences. *Fundamenta Mathematica* **44**, 37–51 (1957)
9. Labhalla, S., Lombardi, H.: Real numbers, continued fractions and complexity classes. *Ann. Pure Appl. Log.* **50**, 1–28 (1990)
10. Lehman, R.S.: On primitive recursive real numbers. *Fundamenta Mathematica* **49**(2), 105–118 (1961)
11. Péter, R.: *Rekursive funktionen*. Verlag der Ungarischen Akademie der Wissenschaften, Budapest (1957). English translation: Academic Press, New York, 1967
12. Rose, H.E.: *Subrecursion. Functions and hierarchies*. Clarendon Press, Oxford (1984)
13. Specker, E.: Nicht konstruktiv beweisbare Sätze der Analysis. *J. Symbol. Log.* **14**, 145–158 (1949)



Computability of Ordinary Differential Equations

Daniel S. Graça^{1,2}  and Ning Zhong³

¹ FCT, Universidade do Algarve, C. Gambelas, 8005-139 Faro, Portugal
dgraca@ualg.pt

² Instituto de Telecomunicações, Lisbon, Portugal

³ DMS, University of Cincinnati, Cincinnati, OH 45221-0025, USA

Abstract. In this paper we provide a brief review of several results about the computability of initial-value problems (IVPs) defined with ordinary differential equations (ODEs). We will consider a variety of settings and analyze how the computability of the IVP will be affected. Computational complexity results will also be presented, as well as computable versions of some classical theorems about the asymptotic behavior of ODEs.

1 Introduction

Ordinary differential equations (ODEs) appear in many applications and are used to describe a large variety of phenomena. For that reason much effort has been directed towards solving ODEs. Although one can solve exactly a few classes of ODEs such as linear ODEs, separable first-order ODEs, etc., in practice we need other methods to analyze non-linear ODEs. For example, we can numerically approximate the solution of an initial-value problem (IVP) defined with an ODE and, in some cases, we can also use some qualitative results to better understand the dynamics of the system. For instance the Poincaré-Bendixson theorem rules out chaotic behavior on dynamical systems defined by autonomous ODEs in the plane.

Numerical methods have gained a particular prominence with the advent of the digital computer. Here we will analyze the problem of finding the solution of an IVP defined with an ODE, from a computational perspective.

2 Background

In this paper we consider (autonomous) ODEs with the format

$$y' = f(y) \tag{1}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $y : \mathbb{R} \rightarrow \mathbb{R}^n$ is a solution of the ODE, with $y = (y_1, \dots, y_n)$. We notice that an initial-value problem (IVP) defined with an apparently more general non-autonomous ODE $y' = f(t, y)$ can always be

reduced to an initial-value problem defined with an autonomous ODE (1) by introducing a new component y_{n+1} satisfying $y_{n+1}(0) = 0$ and $y'_{n+1} = 1$, and by replacing t by y_{n+1} in the non-autonomous ODE. Therefore we don't lose any generality by only considering autonomous ODEs.

There are two important classical results on the existence and uniqueness of solutions for initial value problems (IVPs) involving ODEs, i.e. problems of the type

$$\begin{cases} y'(t) = f(y) \\ y(t_0) = y_0 \end{cases} \tag{2}$$

Both results are important when analyzing IVPs (2) from a computability perspective. The first one is Peano's existence theorem which asserts that if f is continuous, then (2) has at least one (local) solution (see e.g. [2] for a precise statement of this theorem). While this is a very general theorem (only continuity of f is assumed), it poses some challenges since an IVP (2) can have, under these conditions, an infinite number of solutions. Moreover, it was proved in [22] that there exists an IVP (2) defined on the plane, where f is computable and thus continuous — see Sect. 3 — but which does not admit any computable solution. This result shows that it is not enough to assume continuity of f to ensure computability of a solution of an IVP (2).

For that reason, we need the second classical result, the Picard–Lindelöf theorem, which is an existence *and* uniqueness theorem for IVPs (2). This theorem says that if f is Lipschitz continuous, then (2) has one and only one (local) solution. Moreover, this theorem is constructive and thus can be used to compute the solution of (2) when f is Lipschitz continuous. We recall that a function f is Lipschitz continuous if there is a constant $K > 0$ such that

$$\|f(x) - f(y)\| \leq K \|x - y\| \tag{3}$$

for all x, y in the domain of f .

The idea to prove the Picard–Lindelöf theorem is to first construct a very rough approximation of the solution of (2). By Picard's method one can construct a sequence of approximate solutions which will converge to the solution of (2). Moreover, by using the Lipschitz condition (3), we are able to show that the sequence uniformly converges (locally) to the solution of (2) and that the convergence rate can be expressed in terms of the Lipschitz constant K . From these facts we can conclude that the solution of (2) is locally computable if f is Lipschitz continuous and computable.

However, the previous two results are local. Thus it is natural to look also for global results. When the hypotheses of the Picard–Lindelöf theorem are satisfied, the above local existence and uniqueness can be extended globally. The idea is to apply the (local) Picard–Lindelöf theorem to (2) to show that a local solution exists and is unique in the time interval $[t_{-1}, t_1]$, with $t_0 \in [t_{-1}, t_1]$. By applying again the Picard–Lindelöf theorem to two IVPs (1) associated to the initial conditions $y(t_1) = y_1$ and $y(t_{-1}) = y_{-1}$, where y_1 and y_{-1} are the values of the local solution of (2) at times t_1 and t_{-1} , respectively, we obtain local solutions for these two IVPs. Since those solutions are unique, they must be equal to the

previously obtained solution on $[t_{-1}, t_1]$. Thus by “gluing” those solutions with the previous one, we can get a local solution in a new time interval $[t_{-2}, t_2]$, with $t_{-2} < t_{-1}$ and $t_1 < t_2$. By continuing this procedure we get a solution which is defined in a time interval $(t_{-\infty}, t_{\infty})$. It can be shown that this interval is maximal in the following sense: either t_{∞} is $+\infty$ ($t_{-\infty}$ is $-\infty$) or one of the following two cases occurs:

- the solution explodes in finite time;
- the solution leaves the domain of f .

More formally, the following result is obtained (see e.g. [13, pp. 12–13]).

Proposition 1 (*Maximal Interval of Existence*). *Let $f : E \rightarrow \mathbb{R}^n$ be continuous on E , where $E \subseteq \mathbb{R}^{n+1}$ is an open set, and let y be some solution of the IVP $y' = f(t, y), y(t_0) = y_0$ in a nonempty interval. Then y can be extended (as a solution of the IVP) over a maximal interval of existence of the form (α, β) , where $\alpha \in [-\infty, \infty)$ and $\beta \in (-\infty, \infty]$. Moreover, $y(t)$ leaves every compact subset of E , i.e. $y(t)$ tends to the boundary ∂E of E , as $t \rightarrow \alpha$ from above and as $t \rightarrow \beta$ from below.*

In the following sections, we will discuss the computability and computational complexity of solving an IVP (2) from local as well as global perspectives. But first some basic notions concerning computability with real numbers are in order.

3 Computable Analysis

In this paper, we consider computability questions involving real numbers, which require computations with infinite data. Computable analysis provides a proper setting for studying such problems. In computable analysis, real numbers are represented as infinite sequences of finite objects and computations are performed only on a portion of this input; for example, a real number can be represented by a Cauchy sequence of rational numbers [5, 6, 24]. A key feature of this approach to computing with infinite data is that the result can be computed with any guaranteed accuracy, usually by providing more computation time to read more information from the input in exchange of augmented (rigorous) accuracy of results. Computable analysis is gaining an increasing amount of interest due to several factors: (i) it allows computations with more general mathematical objects, such as real numbers, open or closed subsets of \mathbb{R}^n , and functions defined on subsets of \mathbb{R}^n or over more general spaces (function spaces, metric spaces, etc.); (ii) it makes possible to use many ideas and notions from theoretical computer science developed originally for discrete objects, including the study of non-computable problems over \mathbb{R}^n , the classification of problems to different complexity classes, etc.; (iii) it reflects the limitation that computers can only handle a finite number of bits at a time. These features of computable analysis are well suited for our purposes. In the following, we present some basic definitions. More details can be found in [5, 6, 24] and references therein .

Informally, an object is computable within the computable analysis framework if it can be approximated by computer-generated approximations with an arbitrarily high precision. To formalize this notion, one needs to encode infinite objects, such as real numbers, as infinite sequences of finite objects, called representations. In the case of real numbers, this can be done with sequences of rationals converging rapidly to a given real number. This notion can be extended to more general objects like open/closed sets in \mathbb{R}^n , etc., by using representations (see [24] for a complete development).

In general, a represented space is a pair $(X; \delta)$ where X is a set, $\text{dom}(\delta) \subseteq \Sigma^{\mathbb{N}}$, and $\delta : \subseteq \Sigma^{\mathbb{N}} \rightarrow X$ is an onto map (" $\subseteq \Sigma^{\mathbb{N}}$ " is used to indicate that the domain of δ may be a subset of $\Sigma^{\mathbb{N}}$). Every $q \in \text{dom}(\delta)$ such that $\delta(q) = x$ is called a δ -name of x (or a name of x when δ is clear from context). For example, a possible representation of a real number is the encoding of a sequence of rationals converging rapidly to it (see more details below). An element $x \in X$ is said to be computable if it has a computable name in $\Sigma^{\mathbb{N}}$ (the notion of computability on $\Sigma^{\mathbb{N}}$ is well established). A map $\Phi : (X; \delta_X) \rightarrow (Y; \delta_Y)$ between two represented spaces is computable if there is a computable map $\phi : \subseteq \Sigma^{\mathbb{N}} \rightarrow \Sigma^{\mathbb{N}}$ such that $\Phi \circ \delta_X = \delta_Y \circ \phi$. Informally speaking, this means that there is a computer program that outputs a name of $\Phi(x)$ when given a name of x as input [5].

Since we are interested in computing the operator Φ which maps (f, t_0, y_0) to a solution y of (2), we need to have representations of real numbers (for the inputs t_0, y_0) and of functions (for the input f and the output y). Here we use the following representations for points in \mathbb{R}^n and for continuous functions defined on $I_1 \times I_2 \times \dots \times I_n \subset \mathbb{R}^n$, where the I_j 's are intervals:

- (1) For a point $x \in \mathbb{R}^n$, a name of x is a sequence $\{r_k\}$ of points with rational coordinates satisfying $|x - r_k| < 2^{-k}$. Thus x is computable if there is a Turing machine (or a computer program or an algorithm) that outputs a rational n -tuple r_k on input k such that $|r_k - x| < 2^{-k}$; for a sequence $\{x_j\}$, $x_j \in \mathbb{R}^n$, a name of $\{x_j\}$ is a double sequence $\{r_{j,k}\}$ of points with rational coordinates satisfying $|x_j - r_{j,k}| < 2^{-k}$.
- (2) For every continuous function f defined on $I_1 \times I_2 \times \dots \times I_n \subseteq \mathbb{R}^n$, where I_j is an interval with endpoints a_j and b_j , a name of f is a double sequence $\{P_{k,l}\}$ of polynomials with rational coefficients satisfying $d_k(P_{k,l}, f) < 2^{-l}$, where $d_k(g, f) = \max\{|g(x) - f(x)| : a_j + 2^{-k} \leq x_j \leq b_j - 2^{-k}, 1 \leq j \leq n\}$ ($d_k(g, f) = 0$ if $[a_j + 2^{-k}, b_j - 2^{-k}] = \emptyset$). Thus, f is computable if there is an (oracle) Turing machine that outputs $P_{k,l}$ (more precisely coefficients of $P_{k,l}$) on input k, l satisfying $d_k(P_{k,l}, f) < 2^{-l}$.
- (3) For every C^m function f defined on $E = I_1 \times I_2 \times \dots \times I_n \subseteq \mathbb{R}^n$, where I_j is an interval with endpoints a_j and b_j , a (C^m) name of f is a double sequence $\{P_{k,l}\}$ of polynomials with rational coefficients satisfying

$$d_k^m(P_{k,l}, f) < 2^{-l},$$

where

$$d_k^m(g, f) = \max_{0 \leq i \leq m} \max\{|D^i g(x) - D^i f(x)| : a_j + 2^{-k} \leq x_j \leq b_j - 2^{-k}\}$$

($d_k^m(g, f) = 0$ if $[a_j + 2^{-k}, b_j - 2^{-k}] = \emptyset$). We observe that a C^m name of f contains information on both f and $Df, D^2f, \dots, D^m f$, in the sense that (P_1, P_2, \dots) is a name of f while $(D^i P_1, D^i P_2, \dots)$ is a name of $D^i f$. See [25] for further details.

Because we also want to characterize the computability of the maximal interval of existence of (2), we need to have computability notions involving sets. Informally, a planar computable open set can be visualized on a computer screen with an arbitrarily high magnification.

- (4) For an open subset A of \mathbb{R}^n , a name of A consists of a pair of an inner-name and of an outer-name; an inner-name is a sequence of balls $B(a_n, r_n) = \{x \in \mathbb{R}^n : d(a_n, x) < r_n\}$, $a_n \in \mathbb{Q}^n$ and $r_n \in \mathbb{Q}$, exhausting A , i.e., $A = \bigcup_{n=1}^\infty B(a_n, r_n)$; an outer-name is a sequence dense in $\mathbb{R}^n - A$. A is said to be r.e. if the sequences $\{a_n\}$ and $\{r_n\}$ are computable; co-r.e. if the sequence (dense in A) is computable; and computable if it is r.e. and co-r.e..

We close this section by noting that computable functions are always continuous (see e.g. [5]).

4 Computability of the Solutions of Ordinary Differential Equations

In this section, we analyze the computability of ODEs in several settings. First we note that if f in (2) is C^1 on a compact set $E = I_1 \times I_2 \times \dots \times I_n \subset \mathbb{R}^n$, where $I_j \subseteq \mathbb{R}$ is a closed interval, then it is Lipschitz continuous there. This fact combined with the constructive nature of the Picard–Lindelöf theorem shows that there is a computable operator which locally computes the solution of an IVP (2) defined by a C^1 function f . The computability of this operator is uniform on f and on the initial data (t_0, y_0) . In particular, if f is computable, then (2) admits a local computable solution. A theorem by Osgood [1] shows that, in the case of an IVP defined over \mathbb{R}^2 , if f is computable, then so is its (local) solution (no C^1 assumption is needed). The idea behind this results is that it is possible to construct two sequences of functions, one converges from above to a solution of (2), while the other converges from below to a solution of (2). Since the solution is assumed to be unique, it follows that both sequences must converge to the solution of (2), thus ensuring its computability.

We notice that if the uniqueness requirement is dropped from (2), then it may happen that (2) has no computable solutions. This was shown in [22] and further improved in [17]. In particular, there is a polynomial-time computable function $f : [0, 1] \times [-1, 1] \rightarrow \mathbb{R}$ such that the IVP $y' = f(t, y), y(0) = 0$ does not have a computable solution on $[0, \delta]$ for any $\delta > 0$. Note that the latter IVP must have several solutions, since computability of f implies continuity of f and therefore at least a solution of the IVP must exist by Peano’s theorem. This solution cannot be unique because uniqueness would imply computability of the solution.

Next we consider the computability of IVPs (2) when f is defined on an open subset of \mathbb{R}^n . Are we able to compute the solution of (2) over its maximal interval of existence? The situation here is more complicated compared to the compact case, because the Picard–Lindelöf theorem can no longer be applied directly to the problem. Although, as described at the end of Sect. 2, it is possible to construct a global solution of (2) from infinitely many local solutions. But one may encounter a problem here - there may not have a master program to compute all local solutions despite the fact that each local solution is computable; in other words, the computations of local solutions are non-uniform and different algorithms are needed for different local solutions. Nevertheless, since the input function f is the same and the initial point for each new local solution is computable from the previous local solutions, we have almost all the ingredients needed to uniformly compute all local solutions and thus to compute the global solution of (2). The remaining problem is that, to ensure the proper convergence of Picard’s method to a local solution, we need a (local) Lipschitz constant; but there is no guarantee that such Lipschitz constants exist if f is merely continuous. However, if f is C^1 , then it is locally Lipschitz and its local Lipschitz constants are computable from its derivative. Hence, we can compute the solution of (2) globally from a C^1 -name of f . This is shown in [10].

Theorem 1. *Assume E is a r.e. open subset of \mathbb{R}^n . Consider the initial-value problem (2) where f is C^1 on E . Let (α, β) be the maximal interval of existence of the solution $y(\cdot)$ of (2) on E . Then:*

1. *The operator $(f, t_0, y_0) \mapsto y(\cdot)$ is computable;*
2. *The operator $(f, t_0, y_0) \mapsto (\alpha, \beta)$ is semi-computable (i.e. an inner name of (α, β) can be computed from f, t_0, y_0).*

In [10] it was also proved that the maximal interval can be non-computable.

Theorem 2. *There is an analytic and computable function $f : \mathbb{R} \rightarrow \mathbb{R}$ such that the unique solution of the problem*

$$\begin{cases} y' = f(y) \\ y(0) = 0 \end{cases} \tag{4}$$

is defined on a non-computable maximal interval of existence.

Since the function f in the above theorem is analytic, we conclude that the computability of the maximal interval (α, β) of existence is not tied up to the smoothness of f . The function f is constructed by creating a computable odd and bijective function $\varphi : (-\alpha, \alpha) \rightarrow \mathbb{R}$, where α is a non-computable real number, such that φ satisfies the following conditions: (i) φ is the solution of (4) for an analytic and computable function f ; and (ii) $\varphi(x) \rightarrow \pm\infty$ as $x \rightarrow \pm\alpha^\mp$.

The previous result can be generalized in the following manner. Let $E = \mathbb{R}^n$ and suppose that f is defined on \mathbb{R}^n . If a solution $y(t)$ of (2) is defined for all $t \geq t_0$, it is called a (positively) global solution that does not blow up in finite time. In general, it is difficult to predict whether or not a solution will blow

up in finite time from a given initial condition (t_0, y_0) , since it can be proved that this problem is undecidable, even if $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ has only polynomials as components [8]. Nevertheless, certain computational insights on the blow up problem are obtainable from f , as the following theorem in [23] suggests.

Theorem 3. *Consider the IVP (2), where f is locally Lipschitz. Let \mathcal{Z} be the set of all initial values (t_0, y_0) for which the corresponding unique solution of (2) is (positively) global. Then \mathcal{Z} is a G_δ -set (i.e. a countable intersection of open sets) and there is a computable operator determining \mathcal{Z} from f .*

We note that in Theorem 2 we have assumed f is C^1 over its open domain $E \subseteq \mathbb{R}^n$. However, the requirement for f to be C^1 is nevertheless not necessary for ensuring the existence of a unique solution of (2). It is then natural to ask whether it is possible to compute the unique solution of the IVP (2) under more general conditions. A possibility is to consider continuous functions f and to assume uniqueness of solutions of (2). This problem is studied in [7] and the following result, which strengthens Theorem 1, is presented there.

Theorem 4. *Consider the initial value problem (2) where f is continuous on the open set E . Suppose that (2) has a unique solution $y(\cdot)$ on E , defined on the maximal interval of existence (α, β) . Then*

1. *The operator $(f, t_0, y_0) \mapsto y(\cdot)$ is computable;*
2. *The operator $(f, t_0, y_0) \mapsto (\alpha, \beta)$ is semi-computable.*

In particular, if f is a computable function and t_0, y_0 are computable points, then (α, β) is a recursively enumerable open set and the solution $y(\cdot)$ is a computable function.

The proof of this theorem uses a quite different approach; the idea underlying the approach is to try to cover the solution with rational boxes and to test the following conditions, in an algorithmic way: (i) whether a given set of rational boxes is an actual covering of the solution of (2), and (ii) whether the “diameter of the covering” is sufficiently small so that the rational boxes provide an approximation of the solution with the desired accuracy, whenever (i) is satisfied. We can enumerate all possible families of rational boxes and apply the tests (i) and (ii) to each family generated, in a computable manner, therefore obtaining better and better approximations of the unique solution of (2), and thus proving that the solution is computable. This procedure can also be applied to study the computability of differential inclusions (see [7] for more details).

Due to the limited length of this short survey, we will only briefly discuss several results concerning computational complexity for solving ODEs. Let us begin by considering the case where f is defined on a compact set. In this case, it has been shown that the solution of (2), while computable, can have arbitrarily high complexity. The following theorem can be found in [18, p. 219].

Theorem 5. *Let a be an arbitrary computable real number in $[0, 1]$. Then there is a polynomial-time computable function f defined on $[0, 1] \times [-1, 1]$ such that $y(t) = at^2$ is the unique solution of the initial-value problem defined by*

$$y' = f(t, y) \text{ and } y(0) = 0. \tag{5}$$

One solution to this problem is to ensure that f is Lipschitz continuous. In this case, there are various techniques available for computing the solution of (2); for example, Euler’s method. The theorem below was proved in [18, p. 221] by a careful analysis of the computational resources needed when Euler’s method is applied.

Theorem 6. *Let $f : [0, 1] \times B(0, 1) \rightarrow \mathbb{R}^n$, with $B(0, 1) = \{x \in \mathbb{R}^n : \|x\| \leq 1\} \subseteq \mathbb{R}^n$, be a polynomial-time computable Lipschitz function and assume that $y : [0, 1] \rightarrow B(0, 1)$ is the solution of (5). Then y is polynomial-space computable.*

It was shown in [14, Corollary 3.3] that this bound is sharp in the sense that there is a polynomial-time computable Lipschitz function f such that the solution of (5) is PSPACE-hard. Note that the above results are non-uniform (they are only valid when f is polynomial-time computable). A uniform version of this result was proved in [15, Theorem 4.10].

Another related question is whether the smoothness of f helps to reduce the computational complexity of solving an IVP (5). This problem is analyzed in [16] and PSPACE-hardness results are also shown for C^1 functions. There it is also shown that if f is more than once differentiable, then the unique solution can be CH-hard, where $CH \subseteq PSPACE$ is the counting hierarchy.

When f is analytic, the solution of (5) is polynomial-time computable on a compact set. This follows from results of Müller [20] and Ko and Friedman [19]. As remarked in [16, last paragraph of Sect. 5.2], these results also show uniform polynomial computability of the IVP solving operator for analytic functions.

When we consider (5) and (2) for functions f defined over an open domain $E = I_1 \times I_2 \times \dots \times I_n \subset \mathbb{R}^n$, where $I_j \subseteq \mathbb{R}$ is an open interval, the situation is more complicated and very few results exist for that case. The problem is that the above complexity results rely on the existence of a Lipschitz constant and depend on its value. In a compact set, the value of the Lipschitz constant is fixed, and thus its value can be ignored. However, in the non-compact case we have to use local Lipschitz constants. Since the value of local Lipschitz constants can vary in non-trivial ways (it depends on the solution we are trying to compute), affecting hugely the local complexity, it is very hard to obtain a global complexity result. A possible way to solve this problem is to use a bound on the growth of the solution y of (5) as a parameter on the function used to measure the complexity, since the problem of knowing how quickly y can grow is not generally well-understood, even when f is constituted by polynomials. This approach is taken, for instance, in [4, 21] for the case of polynomial IVPs and in [3] for the case of analytic IVPs. In particular, in [21] it is shown that the complexity of solving a polynomial IVP over its maximal interval of definition is polynomial in the length of the solution curve y . The idea of using the length of the solution curve solves several

problems and provides a robust notion of complexity for the non-compact case. See [21] for more details.

Finally, we note that there are several qualitative results about ODEs. In general, the problem of determining the long-term behavior of a system defined with an ODE is a very complicated one. For that reason, qualitative results have been obtained in dynamical systems theory and computable version of several of these results exist. For example, in [11] a computable version of the stable manifold is given, while [12] provides a computable version of the Hartman-Grobman theorem. It is also shown in [9] that the domain of attraction of an hyperbolic equilibrium point x_0 (i.e. a zero of f) is in general non computable, i.e. one cannot decide whether the trajectory starting from some point will ultimately converge to x_0 .

Acknowledgments. Daniel Graça was partially supported by *Fundação para a Ciência e a Tecnologia* and EU FEDER POCTI/POCI via SQIG - Instituto de Telecomunicações through the FCT project UID/EEA/50008/2013. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 731143.

References

1. Birkhoff, G. (ed.): A Source Book in Classical Analysis, pp. 251–258. Harvard University Press, Cambridge (1973). Osgood's Existence Theorem
2. Birkhoff, G., Rota, G.C.: Ordinary Differential Equations, 4th edn. Wiley, Hoboken (1989)
3. Bournez, O., Graça, D.S., Pouly, A.: Solving analytic differential equations in polynomial time over unbounded domains. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 170–181. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22993-0_18
4. Bournez, O., Graça, D.S., Pouly, A.: On the complexity of solving initial value problems. In: Proceedings of 37th International Symposium on Symbolic and Algebraic Computation (ISSAC 2012), vol. abs/1202.4407 (2012)
5. Brattka, V., Hertling, P., Weihrauch, K.: A tutorial on computable analysis. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) New Computational Paradigms: Changing Conceptions of What is Computable, pp. 425–491. Springer, New York (2008). https://doi.org/10.1007/978-0-387-68546-5_18
6. Braverman, M., Cook, S.: Computing over the reals: foundations for scientific computing. *Not. Am. Math. Soc.* **53**(3), 318–329 (2006)
7. Collins, P., Graça, D.S.: Effective computability of solutions of differential inclusions – the ten thousand monkeys approach. *J. Univers. Comput. Sci.* **15**(6), 1162–1185 (2009)
8. Graça, D.S., Buescu, J., Campagnolo, M.L.: Boundedness of the domain of definition is undecidable for polynomial ODEs. In: Dillhage, R., Grubba, T., Sorbi, A., Weihrauch, K., Zhong, N. (eds.) Proceedings of 4th International Conference on Computability and Complexity in Analysis (CCA 2007). Electronic Notes in Theoretical Computer Science, vol. 202, pp. 49–57. Elsevier (2007)
9. Graça, D.S., Zhong, N.: An analytic system with a computable hyperbolic sink whose basin of attraction is non-computable. *Theory Comput. Syst.* **57**, 478–520 (2015)

10. Graça, D.S., Zhong, N., Buescu, J.: Computability, noncomputability and undecidability of maximal intervals of IVPs. *Trans. Am. Math. Soc.* **361**(6), 2913–2927 (2009)
11. Graça, D.S., Zhong, N., Buescu, J.: Computability, noncomputability, and hyperbolic systems. *Appl. Math. Comput.* **219**(6), 3039–3054 (2012)
12. Graça, D.S., Zhong, N., Dumas, H.S.: The connection between computability of a nonlinear problem and its linearization: the Hartman-Grobman theorem revisited. *Theor. Comput. Sci.* **457**(26), 101–110 (2012)
13. Hartman, P.: *Ordinary Differential Equations*, 2nd edn. Birkhäuser, Basel (1982)
14. Kawamura, A.: Lipschitz continuous ordinary differential equations are polynomial-space complete. *Comput. Complex.* **19**(2), 305–332 (2010)
15. Kawamura, A., Cook, S.: Complexity theory for operators in analysis. *ACM Trans. Comput. Theory* **42**(2), Article No. 5 (2012)
16. Kawamura, A., Ota, H., Rösnick, C., Ziegler, M.: Computational complexity of smooth differential equations. *Log. Methods Comput. Sci.* **10**(1:6), 1–15 (2014)
17. Ko, K.I.: On the computational complexity of ordinary differential equations. *Inf. Control* **58**, 157–194 (1983)
18. Ko, K.I.: *Complexity Theory of Real Functions*. Birkhäuser, Basel (1991)
19. Ko, K.I., Friedman, H.: Computing power series in polynomial time. *Adv. Appl. Math.* **9**(1), 40–50 (1988)
20. Müller, N.T.: Uniform computational complexity of Taylor series. In: Ottmann, T. (ed.) *ICALP 1987. LNCS*, vol. 267, pp. 435–444. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-18088-5_37
21. Pouly, A., Graça, D.S.: Computational complexity of solving polynomial differential equations over unbounded domains. *Theor. Comput. Sci.* **626**(2), 67–82 (2016)
22. Pour-El, M.B., Richards, J.I.: A computable ordinary differential equation which possesses no computable solution. *Ann. Math. Log.* **17**, 61–90 (1979)
23. Rettinger, R., Weihrauch, K., Zhong, N.: Topological complexity of blowup problems. *J. Univers. Comput. Sci.* **15**(6), 1301–1316 (2009)
24. Weihrauch, K.: *Computable Analysis: An Introduction*. Springer, Heidelberg (2000). <https://doi.org/10.1007/978-3-642-56999-9>
25. Zhong, N., Weihrauch, K.: Computability theory of generalized functions. *J. ACM* **50**(4), 469–505 (2003)



Topological Analysis of Representations

Mathieu Hoyrup^(✉)

Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
mathieu.hoyrup@inria.fr

Abstract. Computable analysis is the theoretical study of the abilities of algorithms to process infinite objects. The algorithms abilities depend on the way these objects are presented to them. We survey recent results on the problem of identifying the properties of objects that are decidable or semidecidable, for several concrete classes of objects and representations of them. Topology is at the core of this study, as the decidable and semidecidable properties are closely related to the open sets induced by the representation.

In order to process mathematical objects with algorithms one has to encode or represent these objects, by symbolic sequences or sequences of natural numbers. The choice of the representation has a direct impact on the algorithmic tasks that can be performed on these objects, for instance on the class of properties that can be decided or semidecided. A property is *decidable* if there is a program or Turing machine that given a representation of an input, halts and answers whether the input satisfies the property. A property is *semidecidable* if the program halts exactly when the input satisfies the property.

The problem of understanding the classes of decidable and semidecidable properties with respect to a given representation has been addressed in many ways in computability theory and computable analysis. Usually, a representation induces a topology (its final topology), and the semidecidable properties are the effective open sets and the computable functions are the effectively continuous ones. Therefore the problem often amounts to understanding what are the open sets in that topology.

The abstract correspondence between computability and topology has been thoroughly studied, on countably-based topological spaces in [1], on more general spaces in [2, 3] among others.

Our general problem is to understand, for a given class of objects with a particular representation, the information contained in the representation of objects. We investigate this problem by identifying what can be known about the objects from their representations, more precisely:

Problem 1. *Given a class of objects and a representation, identify the properties of objects that are semidecidable w.r.t. this representation.*

The decidable properties are then the ones that are semidecidable and have a semidecidable complement.

We will see several cases where a solution to this problem is known. When the class of semidecidable properties is not well-understood, one can try to identify the difficulty of describing these properties:

Problem 2. *Given a class of objects and a representation, identify the minimal complexity of a set $A \subseteq \mathbb{N}$ such that there is a computable indexing $(P_i)_{i \in A}$ of the semidecidable properties.*

The complexity of A is usually measured in terms of the arithmetical or hyperarithmetical hierarchies.

We present some recent results on this problem for various classes of objects and their representations:

- When considering computable objects only, they can be represented by finite programs, which is at the basis of Markov computability.
- Sometimes a task cannot be performed w.r.t. a particular representation, unless some finite advice is provided, which induces another representation.
- While computable analysis behaves very well on countably-based topological spaces, it is less understood on other spaces. A typical example is the space of higher-order partial or total continuous functionals, introduced by Kleene [4] and Kreisel [5].

Problems 1 and 2 are formulated for semidecidable properties but also make sense for other classes of properties. In $\mathbb{N}^{\mathbb{N}}$, the semidecidable properties are the effective open sets, or the Σ_1^0 -sets from the effective Borel hierarchy. This hierarchy provides different levels of computability of properties and can be transferred from the Baire space to any set X with a representation $\delta_X : \text{dom}(\delta_X) \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow X$ as follows: say that $A \subseteq X$ is a Σ_n^0 -subset of X if there exists a Σ_n^0 -subset P of $\mathbb{N}^{\mathbb{N}}$ such that $\delta_X^{-1}(A) = P \cap \text{dom}(\delta_X)$. This definition is at the basis the development of descriptive set theory on represented spaces [6].

With this definition, the semidecidable properties of points of X are exactly the Σ_1^0 -subsets of X . Then Problems 1 and 2 can also be formulated for Σ_n^0 -properties.

In this paper we use the approach to computable analysis using representations. We mention another important branch using domain theory [7–9].

1 Countably-Based Spaces

The standard way of representing a real number x is by giving a sequence of rational numbers converging at a certain fixed rate to x . Any such sequence can be encoded as an element of $\mathbb{N}^{\mathbb{N}}$ and is called a name of x . A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is then computable if there is a Turing machine converting any name of any $x \in \mathbb{R}$ to a name of $f(x)$. One of the earliest results in computable analysis is that every computable function is continuous. It implies for instance that no non-trivial subset of \mathbb{R} is decidable because it should be clopen. Similarly, the semidecidable

subsets of \mathbb{R} are exactly the effective open sets, i.e. the open sets that can be expressed as computable unions of open intervals with rational endpoints.

The relationship between computability and continuity has been taken for granted and has suggested a standard way of representing points in an arbitrary topological space with a countable basis: a point x is represented by any list of (indices of) the basic neighborhoods of x . With this representation, every computable function is continuous and moreover a function is continuous if and only if it is computable relative to some oracle. Similarly, the semidecidable sets are the effective open sets (i.e., the unions of computable sequences of basic open sets) and a set is open if and only if it is semidecidable relative to some oracle.

Thus for countably-based spaces with the standard representation, the situation is pretty clear and the solution to Problem 1 is:

Solution to Problem 1. *The semidecidable properties are the computable unions of basic open sets.*

Therefore, the answer to Problem 2 is as simple as possible: there is computable enumeration of the semidecidable properties, derived from a computable enumeration of the c.e. subsets of \mathbb{N} , so one can take $A = \mathbb{N}$.

Solution to Problem 2. *The minimal complexity is Δ_1^0 .*

Examples of countably-based spaces are:

- The real numbers with the Euclidean topology, generated by the rational open intervals: the standard representation is equivalent to the Cauchy representation.
- The Baire space $\mathbb{N}^{\mathbb{N}}$, or space of total functions from \mathbb{N} to \mathbb{N} with the product topology induced by the cylinders: the standard representation is equivalent to the trivial representation, where each $f : \mathbb{N} \rightarrow \mathbb{N}$ is a name of itself,
- The partial functions from \mathbb{N} to \mathbb{N} , with the Scott topology induced by the cylinders.

For a complete development of computable analysis on countably-based spaces, we refer the reader to [1].

2 Markov Computability

In Markov's school of recursive constructive mathematics, a real number is a program computing a Cauchy sequence of rationals converging at a certain fixed rate. A function on real numbers is Markov computable if there is a procedure that transforms a program for the input into a program for the output. The comparison of this notion with the more standard notion of computable function (defined in Sect. 1) has been thoroughly studied in the 50's.

When the inputs are the partial computable functions, having a program (or an index, or Gödel number) for the input or a standard name makes no difference: the decidable and semidecidable properties are the same (Rice and Rice-Shapiro

theorems), the computable functionals are the same (Myhill-Shepherdson theorem).

When the inputs are the total computable functions or the computable real numbers, having a program for the input or a name makes no difference when computing a total functional or deciding some property (Kreisel-Lacombe-Schoenfield/Ceitin Theorem), however it does make a difference when computing a partial functional or semideciding some property (Friedberg).

Let us give an example of a property of total computable functions f that is Markov semidecidable (i.e., semidecidable from any index of f) but not semidecidable from f itself. Let $(\varphi_e)_{e \in \mathbb{N}}$ be some canonical effective numbering of the partial computable functions. The property of function f is: for all n , there exists $e \leq n$ such that φ_e coincides with f on inputs $0, \dots, n$.

So Problem 1 arises here: what do the Markov semidecidable properties of total computable functions look like?

More generally, what exactly can be computed given an index, that cannot be computed given a name? What additional information does the index contain? We first show that the only additional information is an upper bound on the index. If X is a countably-based space and δ its standard representation, then let X_c be the set of computable points of X (the points that have a computable name),

- Let δ_M be the Markov representation, representing a point $x \in X_c$ by any index of a computable name of x ,
- And let δ_K be the representation of X_c that represents a point x by a standard name of x and any upper bound on an index of x . One can think of this upper bound as an upper bound on the Kolmogorov complexity of the point, or as giving a finite list of programs such that one of them computes the point.

Theorem 1 ([10]). *Let X be an effective topological space. A subset of X_c is Markov-semidecidable iff it is semidecidable given a standard name and an upper bound on any index.*

More generally, the representations δ and δ_K induce the same properties that are decidable with at most n mind changes (for any fixed $n \in \mathbb{N}$) and the same Σ_2^0 properties.

For $k \geq 3$, the representations δ , δ_K and δ_M induce the same Σ_k^0 properties, simply because X_c is a Σ_3^0 -set.

Is it possible to have a concrete description of the Markov semidecidable properties? To date Problem 1 remains open, however we now give a solution to Problem 2.

A property is Markov semidecidable if there exists a c.e. set $W \subseteq \mathbb{N}$ that is extensional, i.e. if $\varphi_i = \varphi_j$ is total then $i \in W \iff j \in W$. The set $\{e \in \mathbb{N} : W_e \text{ is extensional}\}$ is a Π_3^0 -set and immediately induces an indexing of the Markov semidecidable properties, so the complexity of describing the Markov semidecidable properties is at most Π_3^0 . We recently proved that this bound is optimal.

Theorem 2 ([11]). *There is no Σ_3^0 -indexing of the Markov semidecidable properties of total computable functions.*

The proof is based on a diagonalization, but requires several technical obstacles to be overcome. It follows the same structure as the proof of Theorem 5 below.

Thus,

Solution to Problem 2. *The minimal complexity of an indexing of the Markov semidecidable properties of total computable functions is Π_3^0 .*

3 Finite Advice

Let X be a set with a representation.

In many situations, an algorithm taking points of X as inputs needs an additional finite information about the input to perform a given task. Computations with finite advice have been studied in details by Ziegler [12]. Without loss of generality, that finite information is a natural number. If we denote by $X_n \subseteq X$ the set of points for which $n \in \mathbb{N}$ is a correct advice, then one has $X = \bigcup_{n \in \mathbb{N}} X_n$, and we define a new representation by describing a point $x \in X$ by a name of x together with any n such that $x \in X_n$.

If we understand the former representation, can we understand the new one?

Example 1 A polynomial is represented as an infinite sequence of real numbers (its coefficients) and an upper bound on its degree. Giving the coefficients only, without an upper bound on the degree, would make evaluation uncomputable (a precise account on the difficulty of bounding the degree from the coefficients can be found in [13]). The representation only gives an upper bound on the degree, because giving the exact degree would make simple operations such as addition uncomputable. We denote this representation by δ_{poly} .

Example 2 We saw that representing a computable object by an index is equivalent to representing it by a name plus an upper bound on an index of the object, in the sense that the two representations induce the same semidecidable properties.

Example 3 Given a random sequence $s \in \{-1, 1\}^{\mathbb{N}}$, the random harmonic series $\sum_{n \in \mathbb{N}} \frac{s_n}{n}$ converges and the sum can be uniformly computed from s and an upper bound on its randomness deficiency (it is layerwise computable, see [14]).

We give an answer to Problem 1 in a particular case.

Theorem 3 *Let (X, d) be a computable metric space such that $X = \bigcup_n X_n$ where X_n are uniformly effective compact sets. When representing points x by pairs (p, n) where p is a Cauchy name of x and n is such that $x \in X_n$, the semidecidable properties are:*

- The basic open metric balls,
- The set $\{x : \forall n, d(x, X_n) < \epsilon_n\}$, where $(\epsilon_n)_{n \in \mathbb{N}}$ is any computable sequence of positive rationals,
- Effective unions of finite intersections of these properties.

This result is unpublished, but its proof in a particular case appears in [15].

So we obtain a concrete description of the semidecidable properties, solving Problem 1. It implies a solution to Problem 2: there is a Π_2^0 -indexing of the semidecidable properties (the computable sequences of positive rationals can be enumerated from a Π_2^0 -set). Whether this complexity is optimal depends on the decomposition $(X_n)_{n \in \mathbb{N}}$.

Solution to Problem 1. A polynomial is a sequence of coefficients $c_n \in \mathbb{R}$ that is eventually null. The properties of polynomials that are semidecidable w.r.t. δ_{poly} are:

- Given a rational interval (a, b) and $n \in \mathbb{N}$, whether $c_n \in (a, b)$,
- Given a positive sequence $(\epsilon_n)_{n \in \mathbb{N}}$, whether $c_n < \epsilon_n$ for all n ,
- Effective unions of finite intersections of these properties.

Unfortunately, Theorem 3 does not apply to Example 2, i.e. to the Markov semidecidable properties of total computable functions. Indeed, the set X_n of total computable functions having an index smaller than n , although effectively compact, is not so *uniformly in n* . However when considering subrecursive classes rather than arbitrary total computable functions, one ends up with a uniformly effective compact decomposition to which Theorem 3 can be applied, as we now show.

Subrecursive classes of functions. Let $\text{PR} \subseteq \mathbb{N}^{\mathbb{N}}$ be the set of primitive recursive functions. For the purpose of semideciding a property, it can be easily shown that having a primitive recursive definition of $f \in \text{PR}$ is equivalent to having a direct access to f and an upper bound on an arbitrary primitive recursive definition of f , thus we are in the case of Theorem 3 where PR_n is the set of functions having a primitive recursive definition of length at most n (for any reasonable measure of length).

Solution to Problem 2. ([15]). The semidecidable properties of primitive recursive functions are:

- Given $a, b \in \mathbb{N}$, whether $f(a) = b$,
- Given a computable non-decreasing unbounded function h , whether for all n there exists a primitive recursive definition of length $\leq h(n)$ compatible with f on inputs $0, \dots, n$,
- Effective unions of finite intersections of these properties.

This result actually holds for any subrecursive class, i.e. any class C of total computable functions that admits a sound and complete programming language, i.e. a decidable language $L \subseteq \Sigma^*$ (where Σ is some finite alphabet) with computable semantics, i.e. with a computable surjective map from L to C . Again, for

the purpose of semideciding a property of functions, having a program $w \in L$ of f is equivalent to having a direct access to f and an upper bound on the length of any program $w \in L$ for f .

Examples of subrecursive classes are: the polynomial-time computable functions, the elementary functions, the computable functions that are provably total in Peano Arithmetic, etc.

4 Other Spaces

All the details and proofs of the results in this section can be found in [11].

So far, we have given results about semidecidable properties only, but it is possible to consider other classes of properties. In $\mathbb{N}^{\mathbb{N}}$, the semidecidable properties are the effective open sets, or the Σ_1^0 -sets from the effective Borel hierarchy. This hierarchy provides different levels of computability of properties.

Any space X with a representation $\delta_X : \mathbb{N}^{\mathbb{N}} \rightarrow X$ automatically inherits the effective Borel hierarchy: say that $A \subseteq X$ is a Σ_n^0 -subset of X if there exists a Σ_n^0 -subset P of $\mathbb{N}^{\mathbb{N}}$ such that $\delta_X^{-1}(A) = P \cap \text{dom}(\delta_X)$.

With this definition, the semidecidable properties are exactly the Σ_1^0 -subsets of X . Then Problems 1 and 2 can also be formulated for Σ_n^0 -properties.

For usual countably-based spaces such as \mathbb{R} , or any effective Polish space [16] or quasi-Polish space [17], the effective Borel hierarchy over X behaves nicely: the Σ_n^0 -subsets of X are the computable unions of (differences of) Π_{n-1}^0 -sets, so they can be inductively described (solving Problem 1) and effectively enumerated from \mathbb{N} (i.e., solution to Problem 2 is trivial).

We will see that some other spaces are not so well-behaved.

4.1 Open Subsets of the Baire Space

Here the objects are the open subsets of $\mathbb{N}^{\mathbb{N}}$. The space of these open subsets is denoted by $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$. An open subset of $\mathbb{N}^{\mathbb{N}}$ is naturally represented by any list of cylinders whose union is the open set. For this space, Problems 1 and 2 have a simple solution. The semidecidable properties of open sets $U \subseteq \mathbb{N}^{\mathbb{N}}$ are:

- Given an effective compact set $K \subseteq \mathbb{N}^{\mathbb{N}}$, whether $K \subseteq U$,
- Effective unions of these properties.

An effective compact set can be obtained as $\{f : f \leq g\} \setminus V$, where g is computable and V is an effective open set. In particular there is a Π_2^0 -indexing of the semidecidable properties, and this is optimal.

Therefore the semidecidable or Σ_1^0 properties are well-understood. However understanding the Σ_2^0 properties is an open problem. Contrary to what happens on Polish or quasi-Polish spaces, they cannot all be obtained as countable unions of differences of Π_1^0 -sets, and more generally as countable boolean combinations of open sets. We formalize it by using the notion of Borel set, for which some explanation is needed first.

The representation on the space $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ induces a topology. If $K \subseteq \mathbb{N}^{\mathbb{N}}$ is compact then the set $\{U \in \mathcal{O}(\mathbb{N}^{\mathbb{N}}) : K \subseteq U\}$ is an open subset of $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$, and the topology on $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ is generated by all these sets where K ranges over the compact subsets of $\mathbb{N}^{\mathbb{N}}$.

As a topological space, $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ has a notion of Borel subset: the class of Borel subsets of $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ is the smallest class of sets containing the open sets and closed under taking complements and countable unions. In this sense, the Borel sets can be seen as countable boolean combinations of open sets.

Observe that this standard notion of Borel sets should not be confused with the one derived from the representation, consisting of the sets having a Borel pre-image. The following result shows in particular that the notion derived from the representation is strictly more restrictive than the standard notion.

Theorem 4. *There exists a Σ_2^0 -subset of $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ that is not Borel.*

An ingredient of the proof is to show that the evaluation map $\text{Eval} : \mathbb{N}^{\mathbb{N}} \times \mathcal{O}(\mathbb{N}^{\mathbb{N}}) \rightarrow \mathbb{S}$, which is known to be discontinuous for the product topology on $\mathbb{N}^{\mathbb{N}} \times \mathcal{O}(\mathbb{N}^{\mathbb{N}})$ as $\mathbb{N}^{\mathbb{N}}$ is not locally compact (see [18] for instance), is not even Borel. Said differently, the set $\{(f, U) : f \in U\}$ is not a Borel subset of the space $\mathbb{N}^{\mathbb{N}} \times \mathcal{O}(\mathbb{N}^{\mathbb{N}})$ with the product topology. However, that set is open in the topology induced by the representation. In general, the topology induced by the representation on a product space is not in general the product topology but a stronger topology, except for countably-based spaces. Here, $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ is not countably-based and the set $\{(f, U) : f \in U\}$ is an example of a set that discriminates between the two topologies.

Problems 1 and 2 for the Σ_2^0 -subsets of $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ are still to be studied.

4.2 Kleene-Kreisel Functionals

The previous space is similar to the space of continuous partial functionals from $\mathbb{N}^{\mathbb{N}}$ to \mathbb{N} . We now consider the continuous *total* functionals from $\mathbb{N}^{\mathbb{N}}$ to \mathbb{N} . Such a functional $F : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ is represented by a list of pairs $([u], n)$ such that the value of F on $[u]$ is n (we do not require to list all such pairs, but a list that covers the whole space $\mathbb{N}^{\mathbb{N}}$).

What are the semidecidable properties of functionals? What are the open subsets of $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$? These questions are difficult and we do not have an answer so far. This topological space is not well-understood, although some of its properties are known [19]. We give a solution to Problem 2, which also classifies this space in terms of its *base-complexity* as defined in [20].

Theorem 5. *There is no continuous surjection from $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ to $\mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}})$.*

There is no Σ_2^1 -indexing of the semidecidable properties of functionals from $\mathbb{N}^{\mathbb{N}}$ to \mathbb{N} .

However there is a straightforward Π_2^1 -indexing of the semidecidable properties, which is optimal by the previous result, solving Problem 2.

Solution to Problem 2. *The minimal complexity of an indexing of the semidecidable subsets of $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ is Π_2^1 .*

The proof of Theorem 5 (and Theorem 2) is based on a diagonalization, but several technical problems have to be overcome. Let us briefly explain how it works.

The diagonal argument is very simple: if Y admits a fixed-point free function then there is no surjection from X to Y^X , because given $\phi : X \rightarrow Y^X$ one can build $f(x) = h(\phi(x)(x))$ which is not in the range of ϕ . In this general form, all set-theoretic functions are considered, but it applies as is to subclasses of functions: there is no *continuous* surjection from X to the space $\mathcal{C}(X, Y)$ of *continuous* functions from X to Y (for the suitable topologies). In other words, the argument applies in any cartesian-closed category (this general formulation was done by Lawvere in [21]). In order to apply it to $X = \mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ and $Y = \mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}})$, two problems have to be overcome:

- For any Z , every continuous function $h : \mathcal{O}(Z) \rightarrow \mathcal{O}(Z)$ is Scott continuous hence has a fixed-point by the Kleene fixed-point theorem. However we show that there is a continuous *multi-valued* function $h : \mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}) \rightrightarrows \mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}})$ that has no fixed-point, i.e. such that $U \notin h(U)$ for all $U \in \mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}})$.
- But then the diagonal argument only produces a multi-valued function, and not a function, *unless X is the Baire space or a subspace of it* (a continuous multi-valued function defined on a subset of the Baire space always admits a continuous selection function). It happens that $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ cannot be embedded into $\mathbb{N}^{\mathbb{N}}$. However we show a way of extending the argument to spaces X that contain a sufficiently rich closed set that can be embedded into the Baire space. We show that $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ satisfies this property, hence there is no continuous surjection from $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ to $\mathcal{C}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}, \mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}})) \cong \mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}} \times \mathbb{N}^{\mathbb{N}^{\mathbb{N}}}) \cong \mathcal{O}(\mathbb{N}^{\mathbb{N}^{\mathbb{N}}})$.

Higher order functionals can be generalized to any finite type:

- $\mathbb{N}\langle 0 \rangle = \mathbb{N}$,
- $\mathbb{N}\langle k \rangle + 1 = \mathbb{N}^{\mathbb{N}\langle k \rangle}$,

In the same way, we show that there is no continuous surjection from $\mathbb{N}\langle k \rangle$ to $\mathcal{O}(\mathbb{N}\langle k \rangle)$, and that there is no indexing of the semidecidable subsets of $\mathbb{N}\langle k \rangle$ from any Σ_k^1 -subset of \mathbb{N} .

The hierarchy of continuous functionals can be extended to the countable ordinals by adding:

- $\mathbb{N}\langle \lambda \rangle = \prod_{\alpha < \lambda} \mathbb{N}\langle \alpha \rangle$ for a limit countable ordinal λ (where some enumeration of the ordinals below λ is fixed).

We can then prove that for each countable ordinal α , there is no continuous surjection from $\mathbb{N}\langle \alpha \rangle$ to $\mathcal{O}(\mathbb{N}\langle \alpha \rangle)$. An effective version for the constructive ordinals probably holds, but we did not investigate it.

References

1. Weihrauch, K.: Computable Analysis. Springer, Berlin (2000). <https://doi.org/10.1007/978-3-642-56999-9>
2. Schröder, M.: Extended admissibility. *Theor. Comput. Sci.* **284**(2), 519–538 (2002)
3. Pauly, A.: On the topological aspects of the theory of represented spaces. *Computability* **5**(2), 159–180 (2015)
4. Kleene, S.: Countable functionals. In: *Constructivity in Mathematics*, pp. 81–100 (1959)
5. Kreisel, G.: Interpretation of analysis by means of functionals of finite type. In: *Constructivity in Mathematics*, pp. 101–128 (1959)
6. Pauly, A., de Brecht, M.: Descriptive set theory in the category of represented spaces. In: *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, 6–10 July 2015, Kyoto, Japan*, pp. 438–449. IEEE Computer Society (2015)
7. Escard, M.H.: PCF extended with real numbers. *Theor. Comput. Sci.* **162**(1), 79–115 (1996)
8. Edalat, A.: Domains for computation in mathematics, physics and exact real arithmetic. *Bull. Symb. Log.* **3**(4), 401–452 (1997)
9. Blanck, J.: Domain representations of topological spaces. *Theor. Comput. Sci.* **247**(1), 229–255 (2000)
10. Hoyrup, M., Rojas, C.: On the information carried by programs about the objects they compute. In: *STACS 2015, 4–7 March 2015, Garching, Germany. LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik*, vol. 30, pp. 447–459 (2015)
11. Hoyrup, M.: Results in descriptive set theory on some represented spaces. <http://arxiv.org/abs/1712.03680> (2017)
12. Ziegler, M.: Real computation with least discrete advice: a complexity theory of nonuniform computability with applications to effective linear algebra. *Ann. Pure Appl. Log.* **163**(8), 1108–1139 (2012). *Continuity, Computability, Constructivity: From Logic to Algorithms*
13. Pauly, A., Steinberg, F.: Comparing representations for function spaces in computable analysis. *Theory Comput. Syst.* **62**(3), 557–582 (2018)
14. Hoyrup, M., Rojas, C.: An application of Martin-Löf randomness to effective probability theory. In: *Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) CiE 2009. LNCS*, vol. 5635, pp. 260–269. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03073-4_27
15. Hoyrup, M.: The decidable properties of subrecursive functions. In: *ICALP 2016, 11–15 July 2016, Rome, Italy. LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik*, vol. 55, pp. 108:1–108:13 (2016)
16. Brattka, V.: Effective borel measurability and reducibility of functions. *Math. Log. Q.* **51**(1), 19–44 (2004)
17. de Brecht, M.: Quasi-polish spaces. *Ann. Pure Appl. Log.* **164**(3), 356–381 (2013)
18. Escardó, M., Heckmann, R.: Topologies on spaces of continuous functions. *Topol. Proc.* **26**(2), 545–564 (2001–2002)
19. Schröder, M.: The sequential topology on $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$ is not regular. *Math. Struct. Comput. Sci.* **19**(5), 943–957 (2009)
20. de Brecht, M., Schröder, M., Selivanov, V.: Base-complexity classifications of qcb_0 -spaces. *Computability* **5**(1), 75–102 (2016)
21. Lawvere, F.W.: Diagonal arguments and cartesian closed categories. *Category Theory, Homology Theory and their Applications II. LNM*, vol. 92, pp. 134–145. Springer, Heidelberg (1969). <https://doi.org/10.1007/BFb0080769>



A Unified Framework for Designing EPTAS's for Load Balancing on Parallel Machines

Ishai Kones and Asaf Levin^(✉)

Faculty of Industrial Engineering and Management,
The Technion, 32000 Haifa, Israel
ishai.kones@gmail.com, levinas@ie.technion.ac.il

Abstract. We consider a general load balancing problem on parallel machines. Our machine environment in particular generalizes the standard models of identical machines, and the model of uniformly related machines, as well as machines with a constant number of types, and machines with activation costs. The objective functions that we consider contain in particular the makespan objective and the minimization of the ℓ_p -norm of the vector of loads of the machines, and each case allow the possibility of job rejection.

We consider this general model and design an efficient polynomial time approximation scheme (EPTAS) that applies for all its previously-studied special cases. This EPTAS improves the current best approximation scheme for some of these cases where only a polynomial time approximation scheme (PTAS) was known into an EPTAS.

1 Introduction

We consider a model that generalizes many previously-studied optimization problems in the framework of scheduling and (minimization) load balancing problems on parallel machines. We use this generalization in order to exhibit that there is a standard way to design efficient polynomial time approximation schemes for all these special cases and for new special cases as well. In the earlier works, approximation schemes for many of special cases of our model were developed using ad-hoc tricks, we show that such ad-hoc methods are not necessary.

Before going into the details of the definition of our model, we define the types of approximation schemes. A ρ -approximation algorithm for a minimization problem is a polynomial time algorithm that always finds a feasible solution of cost at most ρ times the cost of an optimal solution. The infimum value of ρ for which an algorithm is a ρ -approximation is called the approximation ratio or the performance guarantee of the algorithm. A polynomial time approximation scheme (PTAS) for a given problem is a family of approximation algorithms such

This research was supported by a grant from the GIF, the German-Israeli Foundation for Scientific Research and Development (grant number I-1366-407.6/2016).

that the family has a $(1 + \varepsilon)$ -approximation algorithm for any $\varepsilon > 0$. An efficient polynomial time approximation scheme (EPTAS) is a PTAS whose time complexity is upper bounded by the form $f(\frac{1}{\varepsilon}) \cdot poly(n)$ where f is some computable (not necessarily polynomial) function and $poly(n)$ is a polynomial of the length of the (binary) encoding of the input. A fully polynomial time approximation scheme (FPTAS) is a stricter concept. It is defined like an EPTAS, with the added restriction that f must be a polynomial in $\frac{1}{\varepsilon}$. Note that whereas a PTAS may have time complexity of the form $n^{g(\frac{1}{\varepsilon})}$, where g is for example linear or even exponential, this cannot be the case for an EPTAS. The notion of an EPTAS is modern and finds its roots in the FPT (fixed parameter tractable) literature (see e.g. [4, 5, 9, 21]).

Since these problems are proven to be strongly NP-Hard [10] (as for example our model is an extension of the minimum makespan problem on identical machines), it is unlikely (impossible assuming $P \neq NP$) that an optimal polynomial time algorithm or an FPTAS will be found for them. In our research, we will focus on finding an EPTAS for this general model and as a bi-product, obtain improved results to many of its special cases. As usual, in order to present an EPTAS we can show that for a sufficiently small value of ε there exists an algorithm of time complexity of the form $f(\frac{1}{\varepsilon}) \cdot poly(n)$ with an approximation ratio of $1 + \kappa\varepsilon$ for an arbitrary constant κ (independent of ε).

Our Model. Being a scheduling problem, the definition of the problem can be partitioned into the characteristics of the machines, the properties of the jobs, and the objective function.

Machine Characteristics. We are given m machines denoted as $\{1, 2, \dots, m\}$ each of which can be activated to work in one of τ types denoted as $1, 2, \dots, \tau$. The type of the machine will influence the processing time of a job assigned to that machine. The input defines for every machine i a (positive rational) speed s_i and an activation cost function $\alpha_i(t)$ that is a non-negative rational number denoting the cost of activating machine i in type t . We are also given a budget \hat{A} on the total activation cost of all machines. The meaning of this budget is that a feasible solution needs to specify for every machine i its type t_i such that the total activation cost is at most the budget, that is, the following constraint holds $\sum_{i=1}^m \alpha_i(t_i) \leq \hat{A}$. In our work we assume that τ is a constant while m is a part of the input. Furthermore, without loss of generality we assume that $1 = s_1 \geq s_2 \geq \dots s_m > 0$.

Job Characteristics. There are n jobs denoted as $J = \{1, 2, \dots, n\}$. Job j is associated with a $(\tau$ -dimensional) vector p_j that specifies the size $p_j(t)$ of job j if it is assigned to a machine of type t . That is, if job j is assigned to machine i , and we activate machine i in type t , then the processing time of job j (on this machine) is $\frac{p_j(t)}{s_i}$. Furthermore, for every job j we are given a rejection penalty π_j that is a positive rational number denoting the cost of not assigning job j to any machine. A definition of a feasible solution specifies for every job j if j is rejected (and thus incurs a rejection penalty of π_j) or not and if it is not rejected (i.e., j is accepted), then the machine i that j is assigned to. Formally, we need

to specify a *job assignment* function $\sigma : J \rightarrow \{0, 1, 2, \dots, m\}$, where $\sigma(j) = 0$ means that j is rejected, and $\sigma(j) = i$ for $i \geq 1$ means that j is assigned to machine i .

Definition of the Objective Function. As stated above a feasible solution defines a type t_i for every machine i , and a job assignment function σ . The load of machine i in this solution is $A_i = \sum_{j \in J: \sigma(j)=i} p_j(t_i)/s_i$. Our objective function is specified using a function F defined over the vector of the loads of the machines $F(A_1, A_2, \dots, A_m)$ that is the assignment cost of the jobs to the machines. F is defined by two scalar parameters $\phi > 1$ and $1 \geq \psi \geq 0$ as follows: $F(A_1, A_2, \dots, A_m) = \psi \cdot \max_{i=1}^m A_i + (1 - \psi) \cdot \sum_{i=1}^m (A_i)^\phi$.

The value of ψ has the following meaning. For $\psi = 1$, the value of F is the makespan of the schedule, i.e., the maximum load of any machine, while for $\psi = 0$, the value of F is the sum of the ϕ powers of the loads of the machines, an objective that is equivalent to the ℓ_ϕ norm of the vector of loads. For ψ that is strictly between 0 and 1, the value of F is a convex combination of these classical objectives in the load balancing literature. The common values of ϕ that were motivated by various applications that were considered in the literature are $\phi = 2$ and $\phi = 3$.

Our objective is to find a type t_i for every machine i such that $\sum_{i=1}^m \alpha_i(t_i) \leq \hat{A}$, and a job assignment σ so that the following objective function (denoted as *obj*) will be minimized: $obj = F(A_1, A_2, \dots, A_m) + \sum_{j \in J: \sigma(j)=0} \pi_j$.

Our result is an EPTAS for this load balancing problem. For ease of notation we denote this problem by P and let $\varepsilon > 0$ be such that $1/\varepsilon \geq 100$ is an integer. We will use the fact that ϕ is a constant and the following simple property throughout the analysis.

Lemma 1. *Given a value of $\rho > 1$ and two vectors (A_1, \dots, A_m) and (A'_1, \dots, A'_m) such that for every i we have $A_i \leq A'_i \leq (1 + \varepsilon)^\rho A_i$, then*

$$F(A_1, A_2, \dots, A_m) \leq F(A'_1, A'_2, \dots, A'_m) \leq (1 + \varepsilon)^{\rho \cdot \phi} F(A_1, A_2, \dots, A_m) .$$

Special Cases of Our Model and Related Literature on These Cases.

The objective function we consider here generalizes the makespan minimization objective (the special case with all $\pi_j = \infty$ and $\psi = 1$), the sum of the ϕ powers of the machines loads (the special case with all $\pi_j = \infty$ and $\psi = 0$), as well as these two objectives with job rejections (i.e., finite π_j for some $j \in J$).

As for the machine model that we consider, next we state some of the earlier studied special cases of this model. We say that *machines have pre-specified type* if $\hat{A} = 0$ and for every i we have a value t_i such that $\alpha_i(t_i) = 0$ and $\alpha_i(t) = 1$ if $t \neq t_i$. This special case of the machine environment is the case of unrelated machines with a constant number of types, whose special case where machines have a common speed was studied in [16] who presented an EPTAS for the makespan objective (the extension of this scheme to machines of different speeds was explored in [17]). This EPTAS of [16] improves earlier PTAS's for that special case [3, 11, 23]. The ℓ_p -norm minimization objective for the case where machines have pre-specified types and all speeds are 1 admits a PTAS [3].

The case where machines have pre-specified types generalizes its special case of *uniformly related machines* that is the case where $\tau = 1$. For this machine model, Jansen [15] presented an EPTAS for the makespan objective improving the earlier PTAS established in the seminal work of Hochbaum and Shmoys [14], while Epstein and Levin [6] presented an EPTAS for the minimization of the ℓ_p -norm of the vector of machines loads improving an earlier PTAS by Epstein and Sgall [8]. Later on, Epstein and Levin [7] presented an EPTAS for another scheduling objective, namely total weighted completion times, and their scheme for the case where all jobs are released at time 0, implies a different EPTAS for the minimization of the sum of squares of the loads of the machines on uniformly related machines. As far as we know the two schemes of [6, 7] are the only examples for EPTAS's for load balancing objectives on uniformly related machines where one cannot use the dual approximation method of [13, 14]. Our approach here is based on [7].

The case of identical machines is the special case of uniformly related machines where all speeds are equal. See [1, 12, 13] for earlier approximation schemes for this case.

The next special objective we consider here is *scheduling with rejection*. This is the special case of our objective function where π_j is finite (at least for some jobs). In [2, 8] there is a PTAS for this variant (for $\psi \in \{0, 1\}$) on identical machines and on uniformly related machines.

The last special case we consider here is the *machines with activation costs* model that was considered by [19]. They considered the special case of our model with makespan objective and $\tau = 2$, with $\alpha_i(1) = 0$ for all i , and $p_j(1) = \infty$ for all $j \in J$. In this case activating a machine as type 1 means that the machine is not operating and cannot process any job. For this case [19] presents a PTAS.

Outline of the Scheme. We apply geometric rounding of the parameters of the input, followed by a guessing step in which we guess for each type the minimum index of the machine that is activated to this type together with its approximated load. This guessing is motivated by a standard characterization of near-optimal solutions that is described earlier. Based on these rounding and guessing steps, we formulate a mixed integer linear program (MILP) that is solved to optimality in polynomial time using [18, 20] and the property that the number of integer variables is a constant (see Sect. 3), and we prove that the optimal cost to our scheduling problem P is approximated by the solution obtained to the MILP. Last, we use the solution of the MILP to round it into a feasible solution to problem P whose cost is approximately the cost of the solution of the MILP (see Sect. 4).

2 Initial Steps

Rounding of the Input. In what follows we would like to assume that the speed of each machine is an integer power of $1 + \varepsilon$, and that for every job j and type t , we have that $p_j(t)$ is an integer power of $1 + \varepsilon$. Given an instance I of problem P that does not satisfy these conditions, we round down the speed of

each machine i to an integer power of $1 + \varepsilon$, and for each job j and type t , we round up the value of $p_j(t)$ to an integer power of $1 + \varepsilon$. That is, we create a new rounded instance I' in which the speed of machine i is s'_i , and for each job j and type t , we let $p'_j(t)$ be its size if it is assigned to a machine of type t , where we define $s'_i = (1 + \varepsilon)^{\lfloor \log_{1+\varepsilon} s_i \rfloor} \quad \forall i$, $p'_j(t) = (1 + \varepsilon)^{\lfloor \log_{1+\varepsilon} p_j(t) \rfloor} \quad \forall j, t$. The other parameters of the input are left in I' as they were in I . The analysis of this step is proved in the following lemma that follows using standard arguments. Recall that a feasible solution to P means selecting a type for each machine satisfying the total activation cost constraint and specifying a job assignment function. Then, given a feasible solution to I of cost C_I , then the same solution is a feasible solution to I' of cost (evaluated as a solution to I') at most $(1 + \varepsilon)^{2\phi} \cdot C_I$. Given a feasible solution to I' of cost $C_{I'}$, then the same solution is a feasible solution to I of cost (evaluated as a solution to I) at most $C_{I'}$. Noting that applying the rounding step takes linear time, we conclude that without loss of generality with a slight abuse of notation, we assume that the input instance satisfies the properties that s_i and $p_j(t)$ are integer powers of $1 + \varepsilon$ (for all i, j, t).

Characterization of Near-Optimal Solutions. We say that a feasible solution to P is *nice* if the following property holds. Let $i < i'$, be a pair of machines that are activated to a common type t such that i is the minimum index of a machine that is activated to type t , then the load of i is at least the load of i' times ε^2 . The definition of nice solutions together with the guessing step described next serve as an alternative to the dual approximation method of [13,14] and suit cases in which the dual approximation method does not work (i.e., non-bottleneck load balancing problems). We are able to show that given an instance of P and a feasible solution SOL of cost SOL, there exists a feasible solution SOL' that is a nice solution whose cost SOL' satisfies $SOL' \leq (1 + \varepsilon)^\phi \cdot SOL$.

Guessing Step. We apply a guessing step of (partial) information on an optimal solution among nice solutions. See e.g. [22] for an overview of this technique of guessing (or partitioning the solutions space) in the design of approximation schemes. In what follows, we consider one nice solution of minimal cost among all nice solutions to the (rounded) instance and denote both this solution and its cost by OPT together with its job assignment function σ^o and the type t_i^o assigned by OPT to machine i (for all i).

We guess the approximated value of the makespan in OPT, and denote it by O . That is, if OPT rejects all jobs then $O = 0$, and otherwise the makespan of OPT is in the interval $(O/(1 + \varepsilon), O]$. Furthermore, for every type t , we guess a minimum index $\mu(t)$ of a machine of type t (namely, $\mu(t) = \min_{i:t_i^o=t} i$), and its approximated load L_t that is a value such that the load of machine $\mu(t)$ is in the interval $(L_t - \frac{\varepsilon O}{\tau}, L_t]$. Without loss of generality, we assume that $O \geq \max_t L_t$. The number of different possibilities for the guessed information on OPT is $O(nm \log_{1+\varepsilon} n \cdot (m\tau/\varepsilon)^\tau)$. We note that if we consider the model of machines with pre-specified type, then we do not need to guess the value of $\mu(t)$ (for all t) and the number of different possibilities for the guessed information on OPT is $O(nm \log_{1+\varepsilon} n \cdot (\tau/\varepsilon)^\tau)$.

3 The Mixed Integer Linear Program

Let $\gamma \geq 10$ be a constant that is chosen later (γ is a function of τ and ε). For a type t and a real number W , we say that job j is *large for* (t, W) if $p_j(t) \geq \varepsilon^\gamma \cdot W$, and otherwise it is *small for* (t, W) .

Preliminaries. Our MILP follows the configuration-MILP paradigm as one of its main ingredients. Thus, next we define our notion of configurations. A *configuration* C is a vector encoding partial information regarding the assignment of jobs to one machine where C consists of the following components: $t(C)$ is the type assigned to a machine with configuration C , $s(C)$ is the speed of a machine with configuration C , $w(C)$ is an approximated upper bound on the total size of jobs assigned to a machine with this configuration where we assume that $w(C)$ is an integer power of $1 + \varepsilon$ and the total size of jobs assigned to this machine is at most $(1 + \varepsilon)^3 \cdot w(C)$, $r(C)$ is an approximated upper bound on the total size of small jobs (small for $(t(C), w(C))$) assigned to a machine with this configuration where we assume that $r(C)$ is an integer multiple of $\varepsilon \cdot w(C)$ and the total size of small jobs assigned to this machine is at most $r(C)$, last, for every integer value of ν such that $(1 + \varepsilon)^\nu \geq \varepsilon^\gamma \cdot w(C)$ we have a component $\ell(C, \nu)$ counting the number of large jobs assigned to a machine of configuration C with size $(1 + \varepsilon)^\nu$. Furthermore we assume that $r(C) + \sum_\nu (1 + \varepsilon)^\nu \cdot \ell(C, \nu) \leq (1 + \varepsilon)^3 \cdot w(C)$. Let \mathcal{C} be the set of all configurations. For every pair (s, w) , we have $|\{C \in \mathcal{C} : s(C) = s, w(C) = w\}| \leq \beta := \tau \cdot \left(\frac{2}{\varepsilon}\right)^{(2\gamma+1)^2 \log_{1+\varepsilon}(1/\varepsilon)}$.

Our MILP formulation involves several blocks and different families of variables that are presented next (these blocks have limited interaction). We present the variables and the corresponding constraints before presenting the objective function.

First Block - Machine Assignment Constraints. For every machine i and every type t , we have a variable $z_{i,t}$ that encodes if machine i is assigned type t , where $z_{i,t} = 1$ means that machine i is assigned type t . Furthermore, for every type t and every speed s , we have a variable $m(s, t)$ denoting the number of machines of (rounded) speed s that are assigned type t . For every type t , we have $z_{i,t} = 0$ for all $i < \mu(t)$ while $z_{\mu(t),t} = 1$ enforcing our guessing. The (additional) machine assignment constraints are as follows:

For every machine i , we require $\sum_{t=1}^\tau z_{i,t} = 1$, encoding the requirement that for every machine i , exactly one type is assigned to i . For every type t and speed s , we have $\sum_{i:s_i=s} z_{i,t} = m(s, t)$. Let m_s be the number of machines in the rounded instance of speed s , then $\sum_{t=1}^\tau m(s, t) = m_s$. Last, we have the machine activation budget constraint $\sum_{t=1}^\tau \sum_{i=1}^m \alpha_i(t) z_{i,t} \leq \hat{A}$.

The variables $z_{i,t}$ are fractional variables and their number is $O(m\tau)$, for every type t and for every speed s such that $s_{\mu(t)} \geq s \geq s_{\mu(t)} \cdot \varepsilon^\gamma$ we require that $m(s, t)$ is an integer variable while all other variables of this family of variables are fractional. Observe that the number of variables that belong to this family and are required to be integral (for the MILP) formulation is $O(\tau\gamma \log_{1+\varepsilon} \frac{1}{\varepsilon})$ that is bounded by a polynomial in $\frac{\tau\gamma}{\varepsilon}$, and the number of fractional variables of the family $m(s, t)$ is $O(n\tau)$.

Second Block - Job Assignment to Machine Types and Rejection Constraints. For every job j and every $t \in \{0, 1, \dots, \tau\}$, we have a variable $y_{j,t}$ that encodes if job j is assigned to machine that is activated to type t (for $t \geq 1$) or rejected (for $t = 0$). That is, for $t \geq 1$, if $y_{j,t} = 1$, then job j is assigned to machine of type t , and if $y_{j,0} = 1$ it means that j is rejected (and we will pay the rejection penalty π_j). Furthermore for every type t and every possible integer value ζ we have two variables $n(\zeta, t)$ and $n'(\zeta, t)$ denoting the number of jobs assigned to machine of type t whose (rounded) size (if they are assigned to machine of type t) is $(1 + \varepsilon)^\zeta$ that are assigned as large jobs and that are assigned as small jobs, respectively. Here, possible values of ζ for a given t are all integers for which $(1 + \varepsilon)^\zeta \leq s_{\mu(t)} \cdot \min\{L_t/(\varepsilon^3), O\}$ such that the rounded input contains at least one job whose size (when assigned to a machine of type t) is $(1 + \varepsilon)^\zeta$ (where recall that L_t is the guessed load of machine $\mu(t)$ and O is the guessed value of the makespan). We denote by $\zeta(t)$ the set of possible values of ζ for the given t . We implicitly use the variables $n(\zeta, t)$ and $n'(\zeta, t)$ for $\zeta \notin \zeta(t)$ (i.e., impossible values of ζ) by setting those variables to zero.

The constraints that we introduce for this block are as follows: For every job j , we should either assign it to a machine (of one of the types) or reject it, and thus we require that $\sum_{t=0}^\tau y_{j,t} = 1$. Furthermore, for every type t and possible value of ζ (i.e., $\zeta \in \zeta(t)$) we require, $\sum_{j:p_j(t)=(1+\varepsilon)^\zeta} y_{j,t} \leq n(\zeta, t) + n'(\zeta, t)$.

For the MILP formulation, the variables $y_{j,t}$ are fractional, while the variables $n(\zeta, t)$ and $n'(\zeta, t)$ are integer variables only if $\zeta \in \zeta(t)$ and $(1 + \varepsilon)^\zeta \geq s_{\mu(t)} L_t \varepsilon^\gamma$ (and otherwise they are fractional). Observe that we introduce for this block $O(n\tau)$ fractional variables (excluding variables that are set to 0 corresponding to impossible values of ζ) and $O(\tau\gamma \log_{1+\varepsilon} \frac{1}{\varepsilon})$ integer variables.

Third Block - Configuration Constraints. For every $C \in \mathcal{C}$ we have a variable x_C denoting the number of machines of speed $s(C)$ activated to type $t(C)$ whose job assignment is according to configuration C . Furthermore, for every configuration $C \in \mathcal{C}$ and every integer value of ν such that $(1 + \varepsilon)^\nu < \varepsilon^\gamma w(C)$ we have a variable $\chi(C, \nu)$ denoting the number of jobs whose size (when assigned to machine of type $t(C)$) is $(1 + \varepsilon)^\nu$ that are assigned to machines of configuration C . Such a variable $\chi(C, \nu)$ exists only if there exists at least one job j whose size (when assigned to a machine of type t) is $(1 + \varepsilon)^\nu$. For $C \in \mathcal{C}$, we let $\nu(C)$ denote the set of values of ν for which the variable $\chi(C, \nu)$ exist. For every t , we require that machine $\mu(t)$ has a configuration where $s_{\mu(t)} \cdot L_t$ is approximately $w(C)$. Thus, for every t , we will have the constraint

$$\sum_{C \in \mathcal{C}: s(C)=s_{\mu(t)}, t(C)=t, s_{\mu(t)} \cdot L_t \leq w(C) \leq (1+\varepsilon)^3 \cdot s_{\mu(t)} \cdot L_t} x_C \geq 1.$$

For the MILP formulation, x_C is required to be integer only if C is a heavy configuration, where C is heavy if $w(C) \geq \varepsilon^\gamma L_{t(C)} \cdot s_{\mu(t(C))}$. The variables $\chi(C, \nu)$ are fractional for all $C \in \mathcal{C}$ and $\nu \in \nu(C)$. Observe that the number of integer variables depends linearly in β where the coefficient is upper bounded by a polynomial function of $\frac{\gamma}{\varepsilon}$.

It remains to consider the constraints bounding these variables together with the $n(\zeta, t)$, $n'(\zeta, t)$ and $m(s, t)$ introduced for the earlier blocks. Here, the constraints have one sub-block for each type t . The *sub-block of type t* (for $1 \leq t \leq \tau$) consists of the following constraints:

For every type t and every (rounded) speed s we cannot have more than $m(s, t)$ machines with configurations satisfying $t(C) = t$ and $s(C) = s$, and therefore we have the constraint $\sum_{C \in \mathcal{C}: t(C)=t, s(C)=s} x_C \leq m(s, t)$. For every $\zeta \in \zeta(t)$, we have that all the $n(\zeta, t)$ jobs of size $(1 + \varepsilon)^\zeta$ that we guarantee to schedule on machine of type t are indeed assigned to such machine as large jobs. Thus, we have the constraints $\sum_{C \in \mathcal{C}: t(C)=t} \ell(C, \zeta) \cdot x_C = n(\zeta, t)$. The last constraints ensures that for every $\zeta \in \zeta(t)$, the total size of all jobs of size at least $(1 + \varepsilon)^\zeta$ that are scheduled as small jobs fits the total area of small jobs in configurations for which $(1 + \varepsilon)^\zeta$ is small with respect to $(t, w(C))$. Here, we need to allow some additional slack, and thus for configuration C we will allow to use $r(C) + 2\varepsilon w(C)$ space for small jobs. Thus, for every integer value of ζ we have the constraint $\sum_{\zeta' \geq \zeta} n'(\zeta', t) \cdot (1 + \varepsilon)^{\zeta'} \leq \sum_{C \in \mathcal{C}: t(C)=t, \varepsilon^\gamma \cdot w(C) > (1 + \varepsilon)^\zeta} (r(C) + 2\varepsilon w(C)) x_C$. Observe that while we define the last family of constraints to have an infinite number of constraints, we have that if when we increase ζ , the summation on the left hand side is the same, then the constraint for the larger value of ζ dominates the constraint for the smaller value of ζ . Thus, it suffices to have the constraints only for $\zeta \in \cup_{i=1}^\tau \zeta(t)$. In addition to the last constraints we have the non-negativity constraints (of all variables).

The Objective Function. Using these variables and (subject to these) constraints we define the minimization (linear) objective function of the MILP as

$$\psi \cdot O + (1 - \psi) \cdot \sum_{C \in \mathcal{C}} \left(\frac{w(C)}{s(C)} \right)^\phi \cdot x_C + \sum_{j=1}^n \pi_j \cdot y_{j,0} .$$

Our algorithm solves optimally the MILP and as described in the next section uses the solution for the MILP to obtain a feasible solution to problem P without increasing the cost too much. Thus, the analysis of the scheme is crucially based on proving the following. The optimal objective function value of the MILP is at most $(1 + \varepsilon)^\phi$ times the cost of OPT as a solution to P .

4 Transforming the Solution to the MILP into a Schedule

Consider the optimal solution $(z^*, m^*, y^*, n^*, n'^*, x^*, \chi^*)$ for the MILP, our first step is to round up each component of n^* and n'^* . That is, we let $\hat{n}(\zeta, t) = \lceil n^*(\zeta, t) \rceil$ and $\hat{n}'(\zeta, t) = \lceil n'^*(\zeta, t) \rceil$ for every ζ and every t . Furthermore, we solve the following linear program (denoted as $(LP - y)$) that has totally unimodular constraint matrix and integer right hand side, and let \hat{y} be an optimal integer solution for this linear program: $\min \sum_{j=1}^n \pi_j \cdot y_{j,0}$ s.t. $\sum_{t=0}^\tau y_{j,t} = 1, \forall j \in J, \sum_{j \in J: p_j(t) = (1 + \varepsilon)^\zeta} y_{j,t} \leq \hat{n}(\zeta, t) + \hat{n}'(\zeta, t) \forall t \in \{1, 2, \dots, \tau\}, \forall \zeta \in \zeta(t), y_{j,t} \geq 0 \forall j \in J, \forall t \in \{0, 1, \dots, \tau\}$. We will assign jobs to types (and reject some of the jobs) based on the values of \hat{y} , that is if $\hat{y}_{j,t} = 1$ we will assign j to a machine of type t (if $t \geq 1$) or reject it (if $t = 0$). Since y^* is a feasible solution to $(LP - y)$ of cost that equal the total rejection penalty of

the solution to the MILP, we conclude that the total rejection penalty of this (integral) assignment of jobs to types is at most the total rejection penalty of the solution to the MILP. In what follows we will assign $\hat{n}(\zeta, t) + \hat{n}'(\zeta, t)$ jobs of size $(1 + \varepsilon)^\zeta$ to machines of type t (for all t).

The next step is to round up each component of x^* , that is, let $\hat{x}_C = \lceil x_C^* \rceil$, and allocate \hat{x}_C machines of speed $s(C)$ that are activated as type $t(C)$ and whose schedule follows configuration C . These \hat{x}_C machines are partitioned into $x'_C = \lfloor x_C^* \rfloor$ actual machines and $\hat{x}_C - x'_C$ virtual machines. Both actual and virtual machines are not machines of the instance but temporary machines that we will use for the next step.

Then, it is possible to construct (in polynomial time) an allocation of $\hat{n}(\zeta, t)$ jobs of size $(1 + \varepsilon)^\zeta$ for all t, ζ to (actual or virtual) machines that follow configurations in $\{C \in \mathcal{C} : t(C) = t, (1 + \varepsilon)^\zeta \geq \varepsilon^\gamma \cdot w(C)\}$, and of $\hat{n}'(\zeta, t)$ jobs of size $(1 + \varepsilon)^\zeta$ for all t, ζ to (actual or virtual) machines that follow configurations in $\{C \in \mathcal{C} : t(C) = t, (1 + \varepsilon)^\zeta < \varepsilon^\gamma \cdot w(C)\}$, such that for every machine that follows configuration $C \in \mathcal{C}$, the total size of jobs assigned to that machine is at most $(1 + \varepsilon)^7 w(C)$.

The assignment of jobs for which $y_{j,0} \neq 0$ to machines is specified by assigning every job that was assigned to a virtual machine that follows configuration C to machine $\mu(t(C))$ instead, and assigning the jobs allocated to actual machines by allocating every actual machine to an index in $\{1, 2, \dots, m\}$ following the procedure described in the next step. Before describing the assignment of actual machines to indices in $\{1, 2, \dots, m\}$ of machines in the instance (of problem P), we analyze the increase of the load of machine $\mu(t)$ due to the assignment of jobs that were assigned to virtual machines that follow configuration with type t .

The performance guarantee that we prove for this step is that there is a value of γ for which the resulting total size of jobs assigned to machine $\mu(t)$ is at most $(1 + \varepsilon)^8 \cdot w(C(\mu(t)))$ where machine $\mu(t)$ follows the configuration $C(\mu(t))$.

Next, we describe the assignment of actual machine to indices in $\{1, 2, \dots, m\}$. More precisely, the last step is to assign a type \hat{t}_i for every machine i satisfying the total activation cost bound, and to allocate for every $C \in \mathcal{C}$ and for every actual machine that follows configuration C , an index i such that $\hat{t}_i = t(C)$. This assignment of types will enforce our guessing of $\mu(t)$ for all t . This step is possible using the integrality of the assignment polytope. Specifically we show that there is a polynomial time algorithm that finds a type t_i for every machine i , such that the total activation cost of all machines is at most \hat{A} , for all s, t the number of machines of speed s that are activated to type t is at least the number of actual machines that follow configurations with type t and speed s , and for all t $\mu(t)$ is the minimum index of a machine that is assigned type t .

References

1. Alon, N., Azar, Y., Woeginger, G.J., Yadid, T.: Approximation schemes for scheduling on parallel machines. *J. Sched.* **1**(1), 55–66 (1998)
2. Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multi-processor scheduling with rejection. *SIAM J. Discrete Math.* **13**(1), 64–78 (2000)

3. Bonifaci, V., Wiese, A.: Scheduling unrelated machines of few different types. <http://arxiv.org/abs/1205.0974> (2012)
4. Cesati, M., Trevisan, L.: On the efficiency of polynomial time approximation schemes. *Inf. Process. Lett.* **64**(4), 165–171 (1997)
5. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer-Verlag, Berlin (1999). <https://doi.org/10.1007/978-1-4612-0515-9>
6. Epstein, L., Levin, A.: An efficient polynomial time approximation scheme for load balancing on uniformly related machines. *Math. Progr.* **147**, 1–23 (2013)
7. Epstein, L., Levin, A.: Minimum total weighted completion time: Faster approximation schemes. <http://arxiv.org/abs/1404.1059> (2014)
8. Epstein, L., Sgall, J.: Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica* **39**(1), 43–57 (2004)
9. Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer-Verlag, Berlin (2006). <https://doi.org/10.1007/3-540-29953-X>
10. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York (1979)
11. Gehrke, J.C., Jansen, K., Kraft, S.E.J., Schikowski, J.: A PTAS for scheduling unrelated machines of few different types. In: Freivalds, R.M., Engels, G., Catania, B. (eds.) *SOFSEM 2016*. LNCS, vol. 9587, pp. 290–301. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49192-8_24
12. Hochbaum, D.S.: Various notions of approximations: good, better, best and more. In: Hochbaum, D.S. (ed.) *Approximation Algorithms*. PWS Publishing Company (1997)
13. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. *J. ACM* **34**(1), 144–162 (1987)
14. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM J. Comput.* **17**(3), 539–551 (1988)
15. Jansen, K.: An EPTAS for scheduling jobs on uniform processors: using a MILP relaxation with a constant number of integral variables. *SIAM J. Discrete Math.* **24**(2), 457–485 (2010)
16. Jansen, K., Maack, M.: An EPTAS for scheduling on unrelated machines of few different types. *Algorithms and Data Structures*. LNCS, vol. 10389, pp. 497–508. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62127-2_42
17. Jansen, K., Maack, M.: An EPTAS for scheduling on unrelated machines of few different types. *CoRR*, abs/1701.03263 (v2) (2017)
18. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: *Proceedings of STOC 1983*, pp. 193–206 (1983)
19. Khuller, S., Li, J., Saha, B.: Energy efficient scheduling via partial shutdown. In: *Proceedings of SODA 2010*, pp. 1360–1372 (2010)
20. Lenstra Jr., H.W.: Integer programming with a fixed number of variables. *Math. Oper. Res.* **8**(4), 538–548 (1983)
21. Marx, D.: Parameterized complexity and approximation algorithms. *Comput. J.* **51**(1), 60–78 (2008)
22. Schuurman, P., Woeginger, G.J.: *Approximation schemes - a tutorial* (2001). <http://www.win.tue.nl/~gwoegi/papers/ptas.pdf>
23. Wiese, A., Bonifaci, V., Baruah, S.K.: Partitioned EDF scheduling on a few types of unrelated multiprocessors. *Real-Time Syst.* **49**(2), 219–238 (2013)



Weak Reduction Principle and Computable Metric Spaces

Margarita Korovina¹(✉) and Oleg Kudinov²

¹ A.P. Ershov Institute of Informatics Systems, SbrAS, Novosibirsk, Russia
rita.korovina@gmail.com

² Sobolev Institute of Mathematics, SbrAS, Novosibirsk, Russia
kud@math.nsc.ru

Abstract. This paper is a part of the ongoing research on developing a foundation for studying arithmetical and descriptive complexity of partial computable functions in the framework of computable topology. We propose new principles for computable metric spaces. We start with a weak version of Reduction Principle (WRP) and prove that the lattice of the effectively open subsets of any computable metric space meets WRP. We illustrate the role of WRP within partial computability. Then we investigate the existence of a principal computable numbering for the class of partial computable functions from an effectively enumerable space to a computable metric space. We show that while in general such numbering does not exist in the important case of a perfect computable Polish space it does. The existence of a principal computable numbering gives an opportunity to make reasoning about complexity of well-known problems in computable analysis in terms of arithmetical and analytic complexity of their index sets.

1 Introduction

As it is well-known the class of distributive lattices is too rich and far from Boolean algebra \mathbb{B} that has simple descriptions. Hence an idea has been revealed to bound the class of distributive lattices by elementary conditions making them closer to \mathbb{B} . Thus Heyting algebras and the distributive lattices with Reduction principle have been proposed. In classical recursion theory Reduction principle is well-known for the lattice $\Sigma_1^0[\omega]$ [13], in admissible set theory for Σ -definable sets [1] and in descriptive set theory for effectively open subsets of some computable Polish spaces (e.g. Baire space) [11]. However, in general, Reduction principle does not hold for the lattice of effectively open subsets of computable Polish spaces. In particular it does not hold for the lattice of effectively open subsets of the real numbers due connectivity. In this paper we propose a weak version of Reduction principle called WRP.

The research has been partially supported by the DFG grants CAVER BE 1267/14-1 and WERA MU 1801/5-1, RFBR grant A-17-01-00247.

From one hand WRP holds for the lattice of effectively open subsets of any computable metric space (and obviously for the lattice of open subsets of any separable metric space). From the other hand this principle is powerful enough to provide pleasant properties of whole space. Thus we show that if WRP holds for $\mathcal{X} \times \mathbb{R}$, where \mathcal{X} is an effectively enumerable topological space, then $\mathcal{MC}_{\mathcal{X} \rightarrow \mathbb{R}} = \mathcal{PCF}_{\mathcal{X} \rightarrow \mathbb{R}}$, i.e., majorant-computability and partial computability coincide.

Since the notion of a partial computable function on a computable metric space is relatively new [6] the existence of a principal computable numbering for different classes of partial computable functions is not well studied yet. We show that while in general such numbering does not exist in the important case of a perfect computable Polish space it does. The existence of a principal computable numbering gives an opportunity to make reasoning about arithmetical and analytic complexity of well-known problems in computable analysis. We are interested in principal computable numberings since they adequately reflect arithmetical and analytic complexity of subclasses of the partial computable functions, i.e., the index sets of a subclass with respect to any other computable numberings are m -reducible to the index set of the subclass with respect to the principal computable numbering. For example if the index set of a subclass is Σ_n^0 -hard with respect to some computable numbering then it is Σ_n^0 -hard with respect to the principal computable numbering.

2 Basic Background

2.1 Preliminaries

We refer the reader to [2, 13, 14] for basic definitions and fundamental concepts of recursion theory. In this paper we work with the following notion of a computable metric space. A computable metric space is a separable metric space \mathcal{X} with a metric d and a countable dense subset $\mathcal{B} = \{b_1, b_2, \dots\}$ called a *basis* of \mathcal{X} such that the following two relations $\{(n, m, i) \mid d(b_n, b_m) < q_i, q_i \in \mathbb{Q}\}$ $\{(n, m, i) \mid d(b_n, b_m) > q_i, q_i \in \mathbb{Q}\}$ are computably enumerable (c.f. [12, 16]). In this paper we consider only perfect Polish spaces, i.e., Polish spaces without isolated points [12]. Hence, further on we omit the word “perfect” and when we write a computable Polish space we mean a perfect complete computable metric space. We consider these concepts in the framework of effectively enumerable spaces (see Sect. 2.2). The standard notations $B(x, y)$ and $\overline{B}(x, y)$ are used for open and closed balls with the center x and the radius y .

2.2 Effectively Enumerable Topological Spaces

Now we recall the notion of an effectively enumerable topological space. Let (X, τ, α) be a topological space, where X is a non-empty set, $B_\tau \subseteq 2^X$ is a base of the topology τ and $\alpha : \omega \rightarrow B_\tau$ is a numbering.

Definition 1 [8]. *A topological space (X, τ, α) is effectively enumerable if the following conditions hold.*

1. There exists a computable function $g : \omega \times \omega \times \omega \rightarrow \omega$ such that

$$\alpha(i) \cap \alpha(j) = \bigcup_{n \in \omega} \alpha(g(i, j, n)).$$

2. The set $\{i | \alpha(i) \neq \emptyset\}$ is computably enumerable.

For a computable metric space (X, \mathcal{B}, d) in a natural way we define the numbering of the base of the standard topology as follows. First we fix a computable numbering $\alpha^* : \omega \setminus \{0\} \rightarrow (\omega \setminus \{0\}) \times \mathbb{Q}^+$. Then,

$$\alpha(0) = \emptyset, \alpha(i) = B(b_n, r) \text{ if } i > 0 \text{ and } \alpha^*(i) = (n, r).$$

For $\alpha^*(i) = (n, r)$ later we use notation $n = u(i)$ and $r = r_i$.

It is easy to see that (X, τ, α) is an effectively enumerable topological space. Therefore we consider the computable metric spaces as a proper subclass of the effectively enumerable topological spaces. For details we refer to [8].

We recall the notion of an effectively open set.

Definition 2. Let (X, τ, α) be an effectively enumerable topological space. A set $A \subseteq X$ is effectively open if there exists a computably enumerable set $V \subseteq \omega$ such that $A = \bigcup_{n \in V} \alpha(n)$.

The set of all effectively open subsets of X is closed under intersection and union since the class of effectively enumerable subsets of ω is a lattice. By analogy to effective descriptive set theory (EDST), for a effectively enumerable T_0 -space \mathcal{X} , we say that a set $A \in EI[\mathcal{X}]$ if $A = \bigcap_{n \in \omega} O_n$, where $\{O_n\}_{n \in \omega}$ is a computable sequence of effectively open subsets of X . It is worth noting that, for a computable Polish space \mathcal{P} , $EI[\mathcal{P}] = \Pi_2^0[\mathcal{P}]$ however, in general, $EI[\mathcal{X}] \subset \Pi_2^0[\mathcal{X}]$ (see [5]).

2.3 PCF over Effectively Enumerable Topological Spaces

In this subsection we recall the notion of a partial computable function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X} = (X, \tau_X, \alpha)$ is an effectively enumerable topological space and $\mathcal{Y} = (Y, \tau_Y, \beta)$ is an effectively enumerable T_0 -space.

Definition 3 [7]. Let $\mathcal{X} = (X, \tau_X, \alpha)$ be an effectively enumerable topological space and $\mathcal{Y} = (Y, \tau_Y, \beta)$ be an effectively enumerable T_0 -space. A function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is called partial computable (pcf) if there exist a computable sequence of effectively open sets $\{O_n\}_{n \in \omega}$ and a computable function $H : \omega^2 \rightarrow \omega$ such that

1. $\text{dom}(f) = \bigcap_{n \in \omega} O_n$ and
2. $f^{-1}(\beta(m)) = \bigcup_{i \in \omega} \alpha(H(m, i)) \cap \text{dom}(f)$.

In the following if a partial computable function f is everywhere defined we say f is a total computable function. For an effectively enumerable topological space \mathcal{X} and an effectively enumerable T_0 -space \mathcal{Y} we denote the set of partial computable functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ as $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}}$.

Remark 1. It is worth noting that, for computable Polish spaces, Definition 3 corresponds to the results in [4], where partial *TTE*-computable functions have been described via effective continuity and Π_2^0 -domains. Moreover, for partial real functions, the classes of $\mathcal{PCF}_{\mathbb{R} \rightarrow \mathbb{R}}$, the *TTE*-computable [15], the domain-computable [3] and the majorant-computable functions [9] coincide.

The following propositions summarise properties of \mathcal{PCF} that we are going to use in this papers. For more details we refer to [6, 7].

Proposition 1. *Let $\mathcal{X} = (X, \tau, \alpha)$ be an effectively enumerable topological space and $\mathcal{Y} = (Y, \lambda, \beta)$ be an effectively enumerable T_0 -space.*

1. *If $f : \mathcal{X} \rightarrow \mathcal{Y}$ is a pcf then f is continuous at every points of $\text{dom}(f)$.*
2. *A total function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is computable if and only if it is effectively continuous.*

Proposition 2. *Over effectively enumerable T_0 -spaces, \mathcal{PCF} is closed under composition.*

Proposition 3. *A function $f : \mathcal{X} \rightarrow \mathbb{R}$ is a pcf if and only if there exist two effectively open sets $U, V \subseteq X \times \mathbb{R}$ such that*

1. *for all $x \in X$, $U(x)$ is closed downward and $V(x)$ is closed upward;*
2. *if $x \in \text{dom}(f)$ then $f(x) = y \leftrightarrow \{y\} = \mathbb{R} \setminus (U(x) \cup V(x))$;*
3. *if $x \notin \text{dom}(f)$ then the set $\mathbb{R} \setminus (U(x) \cup V(x))$ is not a singleton.*

2.4 \mathcal{MC} over Effectively Enumerable Topological Spaces

In this subsection we recall the notion of a majorant-computable function on an effectively enumerable topological space which agrees with the similar notion on the predicate structures [9, 10]. Let $U(x) < V(x)$ denote that, for all $z_1 \in \mathbb{R}$ and $z_2 \in \mathbb{R}$, if $z_1 \in U(x)$ and $z_2 \in V(x)$ then $z_1 < z_2$.

Definition 4. *Let \mathcal{X} be an effectively enumerable topological space. A partial function $f : \mathcal{X} \rightarrow \mathbb{R}$ is called majorant-computable if there exist two effectively open sets $U, V \subseteq X \times \mathbb{R}$ satisfying the following requirements:*

1. $(\forall x \in X) (U(x) < V(x))$;
2. *for all $x \in X$, $U(x)$ is closed downward and $V(x)$ is closed upward;*
3. $f(x) = y \leftrightarrow \{y\} = \mathbb{R} \setminus (U(x) \cup V(x))$.

If we fix a canonical numbering $\tilde{\alpha}$ of the base of the topology on $X \times \mathbb{R}$ then $U = \bigcup_{i \in V_1} \tilde{\alpha}(i)$ and $V = \bigcup_{i \in V_2} \tilde{\alpha}(i)$ for some computable enumerable sets V_1 and V_2 . We say that the pair (U, V) defines a partial majorant-computable function $f : \mathcal{X} \rightarrow \mathbb{R}$ if U and V satisfy the conditions of Definition 4. For an effectively enumerable space \mathcal{X} we denote the set of partial majorant-computable functions $f : \mathcal{X} \rightarrow \mathbb{R}$ as $\mathcal{MC}_{\mathcal{X} \rightarrow \mathbb{R}}$.

3 Weak Reduction Principle

Definition 5. Let L be a distributive lattice. We say that *Weak Reduction Principle holds on L* if for any $A, B \in L$ there exist $C, D \in L$ such that the following conditions hold

1. $A \subseteq B \cup C$ and $C \subseteq A$;
2. $B \subseteq A \cup D$ and $D \subseteq B$;
3. $C \cap D = \emptyset$.

Remark 2. If L admits relative complements then the first condition is equivalent to $A \setminus B \subseteq C \subseteq A$ and the second condition is equivalent to $B \setminus A \subseteq D \subseteq B$.

For an effectively enumerable topological space \mathcal{X} we write $\text{WRP}_{\mathcal{X}}$ if Weak Reduction Principle holds on the lattice of all effectively open subsets of X .

Theorem 1. For every computable metric space \mathcal{M} , $\text{WRP}_{\mathcal{M}}$ holds.

Proof. Assume $A = \bigcup_{i \in \omega} B(a_i, r_i)$ and $B = \bigcup_{j \in \omega} B(b_j, R_j)$, where $r_i, R_j \in \mathbb{Q}^+$ and the corresponding sequences are computable.

For $s \in \omega$ we define

$$x \in A^s \leftrightarrow (\exists i \leq s) \left(\frac{d(x, a_i)}{r_i} < 1 - \frac{1}{2^s} \wedge (\forall k \leq s) \frac{d(x, b_k)}{R_k} > 1 - \frac{1}{2^s} \right)$$

and

$$x \in B^s \leftrightarrow (\exists j \leq s) \left(\frac{d(x, b_j)}{R_j} < 1 - \frac{1}{2^s} \wedge (\forall k \leq s) \frac{d(x, a_k)}{r_k} > 1 - \frac{1}{2^s} \right).$$

Put $A' = \bigcup_{s \in \omega} A^s$ and $B' = \bigcup_{s \in \omega} B^s$. By construction, $A' \subseteq A$ and $B' \subseteq B$. To show that $A \setminus B \subseteq A'$ assume that $x \in A \setminus B$ and $x \in B(a_i, r_i)$ for some $i \in \omega$. Let us choose $s \gg i$ such that $\frac{d(x, a_i)}{r_i} < 1 - \frac{1}{2^s}$. From $\frac{d(x, a_i)}{r_i} < 1$ and $\frac{d(x, b_j)}{R_j} \geq 1$ for all $j \in \omega$ it follows that $x \in A^s$. By similar reasons, $B \setminus A \subseteq B'$.

We prove $A' \cap B' = \emptyset$ by contradiction. Assume without loss of generality that there exists $x \in X$ such that $x \in A^s$, $x \in B^l$ and $s \leq l$. Let us choose $i \leq s$ such that $\frac{d(x, a_i)}{r_i} < 1 - \frac{1}{2^s}$. From the definition of B^l it follows that $\frac{d(x, a_i)}{r_i} > 1 - \frac{1}{2^l}$. Hence $s > l$. This contradicts the choice of s and l . Therefore A' and B' are required subsets.

Proposition 4. There exists an effectively enumerable topological space \mathcal{X} such that $\text{WRP}_{\mathcal{X}}$ does not hold.

Proof. Let us define $X = \omega$, where the closed proper subsets are all finite subsets. Put $A = \omega \setminus \{0\}$ and $B = \omega \setminus \{1\}$. Since any nonempty open sets intersect there are no $A' \subseteq A$ and $B' \subseteq B$ such that $A' \cap B' = \emptyset$, $A' \neq \emptyset$ and $B' \neq \emptyset$. Hence $\text{WRP}_{\mathcal{X}}$ does not hold.

Proposition 5. For an effectively enumerable topological space \mathcal{X} , if we have the equality $\text{MC}_{\mathcal{X} \rightarrow \mathbb{R}} = \text{PCF}_{\mathcal{X} \rightarrow \mathbb{R}}$ then $\text{WRP}_{\mathcal{X}}$ holds.

Proof. Let A and B be effectively open sets. Put

$$U = A \times (-\infty, 0) \cup B \times (-\infty, 1) \text{ and } V = A \times (0, +\infty) \cup B \times (1, +\infty).$$

By Proposition 3 the sets U and V define the following pcf:

$$f(x) = \begin{cases} 0 & \text{if } x \in A \setminus B \\ 1 & \text{if } x \in B \setminus A \\ \uparrow & \text{otherwise.} \end{cases}$$

By assumption, f is a majorant-computable function. Therefore, there exist effectively open sets \tilde{U} and \tilde{V} that define f and $\tilde{U}(x) < \tilde{V}(x)$ for all $x \in X$.

Put

$$C = \{x \in A \mid \tilde{V}(x, \frac{1}{2})\} \text{ and } D = \{x \in B \mid \tilde{U}(x, \frac{1}{2})\}$$

It is clear that $A \setminus B \subseteq C$, $B \setminus A \subseteq D$ and $C \cap D = \emptyset$. Hence $\text{WRP}_{\mathcal{X}}$ holds.

Proposition 6. *For an effectively enumerable topological space \mathcal{X} , if $\text{WRP}_{\mathcal{X} \times \mathbb{R}}$ holds then $\text{MC}_{\mathcal{X} \rightarrow \mathbb{R}} = \text{PCF}_{\mathcal{X} \rightarrow \mathbb{R}}$.*

Proof. Let $f \in \text{PCF}_{\mathcal{X} \rightarrow \mathbb{R}}$. Without loss of generality we assume that $\text{im}(f) \subseteq (0, 1)$ and (U, V) defines f . Since $\text{WRP}_{\mathcal{X} \times \mathbb{R}}$ holds there exist effectively open sets C and D such that $U \setminus V \subseteq C \subseteq U$, $V \setminus U \subseteq D \subseteq V$ and $C \cap D = \emptyset$.

Put

$$U^*(x) = \{r \mid (\exists r' \in (0, 1))(r' > r \wedge [0, r'] \subseteq C(x))\};$$

$$V^*(x) = \{r \mid (\exists r' \in (0, 1))(r' < r \wedge [r', 1] \subseteq D(x))\}.$$

By construction, U^* and V^* are effectively open subset of $X \times \mathbb{R}$. Since $C \cap D = \emptyset$, $U^* \cap V^* = \emptyset$. From $C \supseteq U \setminus V$ and $D \supseteq V \setminus U$ it follows that $V^*(x) \subseteq V(x)$ and $U^*(x) \subseteq U(x)$. By construction, for $x \in \text{dom}(f)$, $V^*(x) = V(x)$ and $U^*(x) = U(x)$. Hence (U^*, V^*) defines a majorant computable function f^* that coincides with f on $\text{dom}(f)$. It is easy to see that $f = f^* + \chi_{\text{dom}(f)}$, where

$$\chi_{\text{dom}(f)}(x) = \begin{cases} 0 & \text{if } x \in \text{dom}(f) \\ \uparrow & \text{if } x \notin \text{dom}(f). \end{cases}$$

From [7] it follows that $\chi_{\text{dom}(f)}$ is a majorant-computable function. Hence f is majorant-computable.

Theorem 2. *For any computable metric space \mathcal{M} , $\text{MC}_{\mathcal{M} \rightarrow \mathbb{R}} = \text{PCF}_{\mathcal{M} \rightarrow \mathbb{R}}$.*

Proof. The claim follows from Propositions 5, 6 and Theorem 1.

4 On Principal Computable Numbering of \mathcal{PCF} over Computable Metric Spaces

A function $\gamma : \omega \rightarrow \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}}$ is called a *numbering* of $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}}$ if $\{\gamma(n) \mid n \in \omega\} = \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}}$. A numbering $\gamma : \omega \rightarrow \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}}$ is called *computable* if $\Gamma : \omega \times \mathcal{X} \rightarrow \mathcal{Y}$ such that $\Gamma(n, x) = \gamma(n)(x)$ is a partial computable function. The numbering γ is called *principal computable* if it is computable and every computable numbering ξ is computably reducible to γ .

Proposition 7 [7]. *For any effectively enumerable T_0 -space \mathcal{X} there exists a principal computable numbering of $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathbb{R}}$.*

Proposition 8 [6]. *For any computable Polish spaces \mathcal{P}_1 and \mathcal{P}_2 there exists a principal computable numbering of $\mathcal{PCF}_{\mathcal{P}_1 \rightarrow \mathcal{P}_2}$.*

Theorem 3. *There is a computable metric space \mathcal{M}_2 such that $\mathcal{PCF}_{\mathcal{M}_1 \rightarrow \mathcal{M}_2}$ does not have a computable numbering for any computable metric space \mathcal{M}_1 .*

Proof. We consider \mathbb{R} with the standard metric d and the basis $B = \mathbb{Q}$. Let \mathbb{R}_c be the computable elements of \mathbb{R} . We choose M_2 such that

- $\mathbb{Q} \subset M_2 \subset \mathbb{R}_c$;
- $M_2 \notin \Sigma_3^0[\mathbb{R}]$.

Put $\mathcal{M}_2 = (M_2, \mathbb{Q}, d)$. Now assume that $\mathcal{M}_1 = (M_1, \mathcal{B}, d')$ is any computable metric space and there exists a computable numbering γ of $\mathcal{PCF}_{\mathcal{M}_1 \rightarrow \mathcal{M}_2}$. It is worth noting that, for any fixed $b \in \mathcal{B}$, $\{\gamma(n)(b) \mid n \in \omega\} = M_2$. Hence

$$x \in M_2 \leftrightarrow (\exists n \in \omega) \left((n, b) \in \text{dom}(\gamma) \wedge (\forall m > 0) (\exists r \in \mathbb{Q}) (\exists q \in \mathbb{Q}) (r < q \wedge |r - q| < \frac{1}{m} \wedge y \in (r, q) \wedge b \in \gamma(n)^{-1}((r, q))) \right).$$

Therefore $M_2 \in \Sigma_3^0[\mathbb{R}]$. This contradicts the choice of M_2 .

These results motivate us to search for a wide class $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}}$ that from one hand enlarges the classes $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathbb{R}}$ and $\mathcal{PCF}_{\mathcal{P}_1 \rightarrow \mathcal{P}_2}$ and from the other hand admits a principal computable numbering.

Theorem 4. *For any effectively enumerable T_0 -space $\mathcal{X} = (X, \tau, \alpha)$ and any computable Polish space $\mathcal{P} = (P, \mathcal{B}, d)$ there exists a principal computable numbering of $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$.*

The proof of the theorem is based on the following Lemmas. Let us denote $\mathcal{B} = \{b_0, \dots, b_n, \dots\}$.

Lemma 1. *For any pcf $G : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}$ one can effectively construct a pcf $F : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}$ which is a restriction of G , i.e., $\text{dom}(F) \subseteq \text{dom}(G)$ and for $(x, y) \in \text{dom}(F)$ $F(x, y) = G(x, y)$, that has the following properties:*

1. If for some $b_n \in \mathcal{B}$ $F(x, b_n) \downarrow$ then $(\forall k \in \omega) F(x, b_k) \downarrow$.
2. If for some $b_n \in \mathcal{B}$ $F(x, b_n) = r$ then $(\exists b_k \in \mathcal{B}) F(x, b_k) < \frac{r}{2}$.
3. For all $b_k \in \mathcal{B}$ and $b_n \in \mathcal{B}$ if $F(x, b_k) \downarrow$ and $F(x, b_n) \downarrow$ then

$$d(b_k, b_n) \leq F(x, b_k) + F(x, b_n).$$

Proof. First we construct a restriction $G^* : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}$ that satisfies the property 1 as follows:

$$G^*(x, y) = \begin{cases} G(x, y) & \text{if } (\forall b_n \in \mathcal{B}) G(x, b_n) \downarrow \\ \uparrow & \text{otherwise.} \end{cases}$$

It is easy to see that $\text{dom}(G^*) \in EI[\mathcal{X} \times \mathcal{P}]$. Then, in order to satisfy the property 2 we construct a computable sequence $\{G_m \mid G_m : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}\}_{m \in \omega}$ of partial computable functions in the following way.

$$G_0 = G^*;$$

$$G_{k+1}(x, y) = \begin{cases} G_k(x, y) & \text{if } (\exists b_n \in \mathcal{B}) G_k(x, b_n) < \frac{G_k(x, y)}{2} \\ \uparrow & \text{otherwise.} \end{cases}$$

By construction,

$$\text{dom}(G_{k+1}) = \text{dom}(G_k) \cap \{(x, y) \mid (\exists n \in \omega)(\exists r \in Q^+)(x, y) \in \mathcal{V}_{2r}^k \wedge (x, b_n) \in \mathcal{O}_r^k\},$$

for some computable sequences $\{\mathcal{O}_r^k\}_{r \in Q^+}$ and $\{\mathcal{V}_q^k\}_{q \in Q^+}$ of effectively open sets such that

$$G_k(x, y) > q \leftrightarrow (x, y) \in \mathcal{V}_q^k \wedge (x, y) \in \text{dom}(G_k) \text{ and}$$

$$G_k(x, y) < r \leftrightarrow (x, y) \in \mathcal{O}_r^k \wedge (x, y) \in \text{dom}(G_k).$$

Hence $\text{dom}(G_k) \in EI[\mathcal{X} \times \mathcal{P}]$ uniformly in k . Put $\text{dom}(\tilde{G}) = \bigcap_{k \in \omega} G_k$. By construction, \tilde{G} is a restriction of G satisfying the properties 1–2. To satisfy the property 3 we define F as follows.

$$F(x, y) = \begin{cases} \tilde{G}(x, y) & \text{if } (\forall b_k \in \mathcal{B})(\forall b_l \in \mathcal{B}) d(b_k, b_l) \leq \tilde{G}(x, b_k) + \tilde{G}(x, b_l) \\ & \wedge \tilde{G}(x, b_k) \downarrow \wedge \tilde{G}(x, b_l) \downarrow \\ \uparrow & \text{otherwise.} \end{cases}$$

It is clear that $\text{dom}(F) \in EI[\mathcal{X} \times \mathcal{P}]$ hence F is a required restriction of G .

Lemma 2. *There exists a principal computable numbering γ of the class of pcfs $F : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}$ satisfying the properties 1–3.*

Proof. From Proposition 7 it follows that there exists the principal computable numbering $\alpha : \omega \rightarrow PCF_{\mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}}$. Using the notation $F = \text{res}(G)$ for the effective construction from Lemma 1 we define $\lambda x.\gamma(n, x) = \text{res}(\alpha(n, x))$.

Lemma 3. *Every pcf $F : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}$ satisfying the properties 1–3 uniquely defines a pcf $f : \mathcal{X} \rightarrow \mathcal{P}$ such that*

- (a) $f(x) \downarrow \leftrightarrow F(x, b_0) \downarrow$;
- (b) *if $x \in \text{dom}(f)$ then $f(x)$ is equal to the unique $y \in \mathcal{P}$ such that $(\forall r \in \mathbb{Q}^+)(\exists b_k \in B(y, r)) F(x, b_k) < r$.*

Proof. Since $\text{dom}(f) = \{x \mid \forall k F(x, b_k) \downarrow\} = \{x \mid F(x, b_0) \downarrow\}$ it is easy to see that $\text{dom}(f) \in EI(\mathcal{X})$. For $x \in \text{dom}(f)$ the property 2 ensures the existence of $y \in \mathcal{P}$ such that $(\forall r \in \mathbb{Q}^+)(\exists b_k \in B(y, r)) F(x, b_k) < r$. In order to show that y is uniquely defined we assume contrary that there exist y_1 and y_2 such that $y_1 \neq y_2$ and the properties (a) and (b) hold. We choose $r \in \mathbb{Q}^+$ such that $4r < d(y_1, y_2)$. By the property (b), there exist $b_k \in B(y_1, r)$ and $b_l \in B(y_2, r)$ such that $F(x, b_k) < r$ and $F(x, b_l) < r$. Hence $d(b_k, b_l) \geq 2r$. This contradicts the property 3. Hence y is uniquely defined and, for $x \in \text{dom}(f)$, $f(x) = y$. So f is a partial function with $\text{dom}(f) \in EI(\mathcal{X})$. From the equality $f^{-1}(B(a, q)) = \{x \in X \mid (\exists r \in \mathbb{Q}^+)(\exists k \in \omega)(F(x, b_k) < r \wedge d(a, b_k) + r < q)\}$ it follows that f is effectively continuous on its domain. Hence f is a pcf.

Remark 3. The construction in Lemma 3 is effective which means that the effective intersection of open sets in the definition of $\text{dom}(f)$ and the computable sequence $\{f^{-1}(B(a, r))\}_{a \in \mathcal{B}, r \in \mathbb{Q}^+}$ are effectively obtained from the definition of F . Further on for such functions we use the notation $f = \text{twist}(F)$.

Proof (Theorem 4). Using Lemma 2 we define $\lambda x. \delta(n, x) = \text{twist}(\lambda z. \gamma(n, z))$. Let us show that δ is a computable numbering of $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$. For given $f : \mathcal{X} \rightarrow \mathcal{P}$ we put $F(x, y) = d(y, f(x))$. It is clear that this function $F : \mathcal{X} \times \mathcal{P} \rightarrow \mathbb{R}^+$ satisfies the properties 1–3. Hence, for some $m \in \omega$, $F(z) = \gamma(m, z)$. By Lemma 3, $f = \text{twist}(F)$. So $\lambda x. f(x) = \delta(m, x)$ and δ is a computable numbering of $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$. In order to show that δ is a principal let us take any computable sequence $\{f_n\}_{n \in \omega}$ of function from $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$. Put $F_n(x, y) = d(y, f(x))$. Since $\{F_n\}_{n \in \omega}$ is a computable sequence and γ is principal, by Lemma 2, we have $F_n(x, y) = \gamma(g(n), (x, y))$, where $g : \omega \rightarrow \omega$ is a computable function. Since $f_n = \text{twist}(F_n)$ we obtain $f_n(x) = \delta(g(n), x)$. Hence δ is a principal computable numbering of $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$.

Definition 6 [13, 14]. *Let $\delta : \omega \rightarrow \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$ be a principal computable numbering and $L \subseteq \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$. The set $Ix(L) = \{n \mid \delta(n) \in L\}$ is called an index set for the subset L .*

In order to calculate complexity of index sets we use arithmetical and analytic hierarchies which classifies certain sets of the natural numbers based on the complexity of formulas defining them [13, 14].

Let $\perp_{\mathcal{X}\mathcal{P}}$ denote the nowhere defined function. In [6] we have shown that for computable Polish spaces \mathcal{X} and \mathcal{Y} the index set $Ix(\{f \in \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}} \mid f \neq \perp_{\mathcal{X}\mathcal{P}}\})$ is Σ_1^1 -complete. It is worth noting that in general this set is not even Σ_1^1 -hard. It is easy to see in the following trivial case when $X = \{0\}$ and $Y = \mathbb{R}$ since the index set $Ix(\{f \in \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{Y}} \mid f \neq \perp_{\mathcal{X}\mathcal{P}}\}) \in \Pi_2^0$ and actually it is Π_2^0 -complete. However the following claims hold in general.

Proposition 9. *Let $\mathcal{X} = (X, \tau, \alpha)$ be an effectively enumerable T_0 -space and $\mathcal{P} = (P, \mathcal{B}, d)$ be a computable Polish space. If K is any non-empty class of pcf $f : \mathcal{X} \rightarrow \mathcal{P}$ such that $\perp_{\mathcal{X}\mathcal{P}} \notin K$ then $Ix(K)$ is Π_2^0 -hard.*

Corollary 1. *Let $K \subseteq \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$. If $Ix(K) \in \Sigma_2^0$ then $Ix(K)$ is Σ_2^0 -complete.*

Corollary 2 (Generalised Rice Theorem). *Let $K \subseteq \mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$. The index set $Ix(K)$ is Δ_2^0 if and only if K is empty or coincides with $\mathcal{PCF}_{\mathcal{X} \rightarrow \mathcal{P}}$.*

References

1. Ershov, Y.L.: Definability and Computability. Springer, New York (1996)
2. Ershov, Y.L.: Theory of numberings. In: Griffor, E.R. (ed.) Handbook of Computability Theory, pp. 473–503. Elsevier Science B.V., Amsterdam (1999)
3. Edalat, A.: Domains for computation in mathematics, physics and exact real arithmetic. Bull. Symb. Logic **3**(4), 401–452 (1997)
4. Hemmerling, A.: Effective metric spaces and representations of the reals. Theor. Comput. Sci. **284**(2), 347–372 (2002)
5. Korovina, M., Kudinov, O.: On higher effective descriptive set theory. In: Kari, J., Manea, F., Petre, I. (eds.) CiE 2017. LNCS, vol. 10307, pp. 282–291. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58741-7_27
6. Korovina, M., Kudinov, O.: Outline of partial computability in computable topology. In: Kari, J., Manea, F., Petre, I. (eds.) CiE 2017. LNCS, vol. 10307, pp. 64–76. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58741-7_7
7. Korovina, M., Kudinov, O.: Complexity for partial computable functions over computable Polish spaces. Math. Struct. Comput. Sci. **28**(3), 429–447 (2016)
8. Korovina, M., Kudinov, O.: Towards computability over effectively enumerable topological spaces. Electr. Notes Theor. Comput. Sci. **221**, 115–125 (2008)
9. Korovina, M., Kudinov, O.: Towards computability of higher type continuous data. In: Cooper, S.B., Löwe, B., Torenvliet, L. (eds.) CiE 2005. LNCS, vol. 3526, pp. 235–241. Springer, Heidelberg (2005). https://doi.org/10.1007/11494645_30
10. Korovina, M.V., Kudinov, O.V.: Characteristic properties of majorant-computability over the reals. In: Gottlob, G., Grandjean, E., Seyr, K. (eds.) CSL 1998. LNCS, vol. 1584, pp. 188–203. Springer, Heidelberg (1999). https://doi.org/10.1007/10703163_14
11. Moschovakis, Y.N.: Descriptive Set Theory. North-Holland, Amsterdam (2009)
12. Moschovakis, Y.N.: Recursive metric spaces. Fund. Math. **55**, 215–238 (1964)
13. Rogers, H.: Theory of Recursive Functions and Effective Computability. McGraw-Hill, New York (1967)
14. Soare, R.I.: Recursively Enumerable Sets and Degrees: A Study of Computable Functions and Computably Generated Sets. Springer, Heidelberg (1987)
15. Weihrauch, K.: Computable Analysis. Springer, Heidelberg (2000). <https://doi.org/10.1007/978-3-642-56999-9>
16. Weihrauch, K.: Computability on computable metric spaces. Theor. Comput. Sci. **113**(1), 191–210 (1993)



Decidable and Undecidable Fragments of First-Order Concatenation Theory

Lars Kristiansen^{1,2(✉)} and Juvenal Murwanashyaka¹

¹ Department of Mathematics, University of Oslo, Oslo, Norway
larsk@math.uio.no

² Department of Informatics, University of Oslo, Oslo, Norway

Abstract. We identify a number of decidable and undecidable fragments of first-order concatenation theory. We also give a purely universal axiomatization which is complete for the fragments we identify. Furthermore, we prove some normal-form results.

1 Introduction

1.1 First-Order Concatenation Theory

First-order concatenation theory can be compared to first-order number theory, e.g., Peano Arithmetic or Robinson Arithmetic. The universe of a standard structure for first-order number theory is the set of natural numbers. The universe of a standard structure for first-order concatenation theory is a set of strings over some alphabet. A first-order language for number theory normally contains two binary functions symbols. In a standard structure these symbols will be interpreted as addition and multiplication. A first-order language for concatenation theory normally contains just one binary function symbol. In a standard structure this symbol will be interpreted as the operator that concatenates two strings. A classical first-order language for concatenation theory contains no other non-logical symbols apart from constant symbols.

In this paper we extend concatenation theory with a binary relation symbol and introduce bounded quantifiers analogous to the bounded quantifiers $(\forall x \leq t)\phi$ and $(\exists x \leq t)\phi$ we know from number theory. Before we go on and state our main results, we will explain some notation and state a few basic definitions.

1.2 Notation and Basic Definitions

We will use $\mathbf{0}$ and $\mathbf{1}$ to denote respectively the bits zero and one, and we use pretty standard notation when we work with bit strings: $\{\mathbf{0}, \mathbf{1}\}^*$ denotes the set of all finite bit strings; $(b)_i$ denotes the i^{th} bit of the bit string b ; and $\mathbf{01}^3\mathbf{0}^2\mathbf{1}$ denotes the bit string $\mathbf{0111001}$. The set $\{\mathbf{0}, \mathbf{1}\}^*$ contains the empty string which we will denote ε .

Let \mathcal{L}_{BT} denote the first-order language that consist of the constants symbols $e, 0, 1$, the binary function symbol \circ and the binary relation symbol \sqsubseteq . We will consider two \mathcal{L}_{BT} -structures named \mathfrak{B} and \mathfrak{D} .

The universe of \mathfrak{B} is the set $\{\mathbf{0}, \mathbf{1}\}^*$. The constant symbol 0 is interpreted as the string containing nothing but the bit $\mathbf{0}$, and the constant symbol 1 is interpreted as the string containing nothing but the bit $\mathbf{1}$, that is, $0^{\mathfrak{B}} = \mathbf{0}$ and $1^{\mathfrak{B}} = \mathbf{1}$. The constant symbol e is interpreted as the empty string, that is, $e^{\mathfrak{B}} = \varepsilon$. Moreover, $\circ^{\mathfrak{B}}$ is the function that concatenates two strings (e.g. $\mathbf{01} \circ^{\mathfrak{B}} \mathbf{000} = \mathbf{01000}$ and $\varepsilon \circ^{\mathfrak{B}} \varepsilon = \varepsilon$). Finally, $\sqsubseteq^{\mathfrak{B}}$ is the substring relation, that is, $u \sqsubseteq^{\mathfrak{B}} v$ iff there exists bit strings x, y such that $xuy = v$.

The structure \mathfrak{D} is the same structure as \mathfrak{B} with one exception: the relation $u \sqsubseteq^{\mathfrak{D}} v$ holds iff u is a prefix of v , that is, iff there exists a bit string x such that $ux = v$. To improve the readability we will use the symbol \preceq in place of the symbol \sqsubseteq when we are working in the structure \mathfrak{D} . Thus, $u \sqsubseteq v$ should be read as “ u is a substring of v ”, whereas $u \preceq v$ should be read as “ u is a prefix of v ”. When we do not have a particular structure in mind, e.g. when we deal with syntactical matters, we will stick to the symbol \sqsubseteq .

We introduce the *bounded quantifiers* $(\exists x \sqsubseteq t)\alpha$ and $(\forall x \sqsubseteq t)\alpha$ as syntactical abbreviations for respectively $(\exists x)[x \sqsubseteq t \wedge \alpha]$ and $(\forall x)[x \sqsubseteq t \rightarrow \alpha]$ (x is of course not allowed to occur in the term t), and we define the Σ -formulas inductively by

- α and $\neg\alpha$ are Σ -formulas if α is of the form $s \sqsubseteq t$ or of the form $s = t$ where s and t are terms
- $\alpha \vee \beta$ and $\alpha \wedge \beta$ are Σ -formulas if α and β are Σ -formulas
- $(\exists x \sqsubseteq t)\alpha$ and $(\forall x \sqsubseteq t)\alpha$ and $(\exists x)\alpha$ are Σ -formulas if α is a Σ -formula.

We assume that the reader notes the similarities with first-order number theory. The formulas that correspond to Σ -formulas in number theory are often called Σ_1 -formulas or Σ_1^0 -formulas. Next we introduce the biterals. The biterals correspond to the numerals of first-order number theory. Let b be a bit string. We define the *biteral* \bar{b} by $\bar{\varepsilon} = e$, $\bar{b\mathbf{0}} = \bar{b} \circ \mathbf{0}$ and $\bar{b\mathbf{1}} = \bar{b} \circ \mathbf{1}$.

A Σ -formula ϕ is called a $\Sigma_{n,m,k}$ -formula if it contains n unbounded existential quantifiers, m bounded existential quantifiers and k bounded universal quantifiers. A *sentence* is a formula with no free variables. The *fragment* $\Sigma_{n,m,k}^{\mathfrak{B}}$ ($\Sigma_{n,m,k}^{\mathfrak{D}}$) is the set of $\Sigma_{n,m,k}$ -sentences true in \mathfrak{B} (respectively, \mathfrak{D}).

To improve the readability we will skip the operator \circ in first-order formulas and simply write st in place of $s \circ t$. Furthermore, we will occasionally contract quantifiers and write, e.g., $\forall w_1, w_2 \sqsubseteq u[\phi]$ in place of $(\forall w_1 \sqsubseteq u)(\forall w_2 \sqsubseteq u)\phi$, and for $\sim \in \{\preceq, \sqsubseteq, =\}$, we write $s \not\sim t$ in place of $\neg s \sim t$.

1.3 Main Results and Related Work

We prove that the fragment $\Sigma_{0,m,k}^{\mathfrak{B}}$ is decidable (for any $m, k \in \mathbb{N}$), and we prove that $\Sigma_{1,2,1}^{\mathfrak{B}}$ and $\Sigma_{1,0,2}^{\mathfrak{B}}$ are undecidable. Furthermore, we prove that the fragments $\Sigma_{0,m,k}^{\mathfrak{D}}$ and $\Sigma_{n,m,0}^{\mathfrak{D}}$ are decidable (for any $n, m, k \in \mathbb{N}$), and we prove that $\Sigma_{3,0,2}^{\mathfrak{D}}$ and $\Sigma_{4,1,1}^{\mathfrak{D}}$ are undecidable. Our results on decidable fragments are corollaries of theorems that have an interest in their own right: We prove the existence of several normal forms, and we give a purely universal axiomatization of concatenation theory which is Σ -complete.

The Axioms of B

1. $\forall x[x = ex \wedge x = xe]$
2. $\forall xyz[(xy)z = x(yz)]$
3. $\forall xy[(x \neq y) \rightarrow ((x0 \neq y0) \wedge (x1 \neq y1))]$
4. $\forall xy[x0 \neq y1]$
5. $\forall x[x \sqsubseteq e \leftrightarrow x = e]$
6. $\forall x[x \sqsubseteq 0 \leftrightarrow (x = e \vee x = 0)]$
7. $\forall x[x \sqsubseteq 1 \leftrightarrow (x = e \vee x = 1)]$
8. $\forall xy[x \sqsubseteq 0y0 \leftrightarrow (x = 0y0 \vee x \sqsubseteq 0y \vee x \sqsubseteq y0)]$
9. $\forall xy[x \sqsubseteq 0y1 \leftrightarrow (x = 0y1 \vee x \sqsubseteq 0y \vee x \sqsubseteq y1)]$
10. $\forall xy[x \sqsubseteq 1y0 \leftrightarrow (x = 1y0 \vee x \sqsubseteq 1y \vee x \sqsubseteq y0)]$
11. $\forall xy[x \sqsubseteq 1y1 \leftrightarrow (x = 1y1 \vee x \sqsubseteq 1y \vee x \sqsubseteq y1)]$

Fig. 1. These are the axioms of the first-order theory B .

Recent related work can be found in Halfon et al. [6], Day et al. [2], Ganesh et al. [3], Karhumäki et al. [8] and several other places, see Sect. 6 of [3] for further references.

The material in Sect. 8 of the textbook Leary and Kristiansen [10] is also related to the research presented in this paper. So is a series of papers that starts with with Grzegorzczuk [4] and includes Grzegorzczuk and Zdanowski [5], Visser [13] and Horiata [7]. These papers deal with the essential undecidability of various first-order theories of concatenation. The relationship between the various axiomatizations of concatenation theory we find in these papers and the axiomatization we give below has not yet been investigated.

The theory of concatenation seems to go back to work of Tarski and Quine, see Visser [13] for a brief account of its history.

2 Σ -complete Axiomatizations

Theorem 1. *Let B and D be the set of axioms given in respectively Figs. 1 and 2. For any Σ -sentence ϕ , we have*

$$\mathfrak{B} \models \phi \Rightarrow B \vdash \phi \quad \text{and} \quad \mathfrak{D} \models \phi \Rightarrow D \vdash \phi$$

Proof. We give a brief sketch of the (very long) proof of $\mathfrak{B} \models \phi \Rightarrow B \vdash \phi$. The proof of $\mathfrak{D} \models \phi \Rightarrow D \vdash \phi$ is similar. Full proofs can found in Kristiansen and Murwanashyaka [9].

Prove (by induction on the structure of t) that there for any variable-free \mathcal{L}_{BT} -term t exists a biteral b such that

$$\mathfrak{B} \models t = b \Rightarrow B \vdash t = b. \tag{1}$$

Prove (by induction on the structure of b_2) that we for any biterals b_1 and b_2 have

$$\mathfrak{B} \models b_1 \neq b_2 \Rightarrow B \vdash b_1 \neq b_2. \tag{2}$$

The Axioms of D

- the first four axioms are the same as the first four axioms of B
- 5. $\forall x[x \preceq e \leftrightarrow x = e]$
- 6. $\forall xy[x \preceq y0 \leftrightarrow (x = y0 \vee x \preceq y)]$
- 7. $\forall xy[x \preceq y1 \leftrightarrow (x = y1 \vee x \preceq y)]$

Fig. 2. These are the axioms of the first-order theory D .

Use $B \vdash \forall x[x0 \neq e \wedge x1 \neq e]$ when proving (2). Furthermore, prove (by induction on the structure of b_2) that we for any biterals b_1 and b_2 have

$$\mathfrak{B} \models b_1 \sqsubseteq b_2 \Rightarrow B \vdash b_1 \sqsubseteq b_2 \quad \text{and} \quad \mathfrak{B} \models b_1 \not\sqsubseteq b_2 \Rightarrow B \vdash b_1 \not\sqsubseteq b_2. \quad (3)$$

It follows from (1), (2) and (3) that we have

$$\mathfrak{B} \models \phi \Rightarrow B \vdash \phi. \quad (4)$$

for any ϕ of one of the four forms $t_1 = t_2$, $t_1 \neq t_2$, $t_1 \sqsubseteq t_2$, and $t_1 \not\sqsubseteq t_2$ where t_1 and t_2 are variable-free terms.

Use induction on the structure of b to prove the following claim:

If $\phi(x)$ is an \mathcal{L}_{BT} -formula such that we have $\mathfrak{B} \models \phi(b) \Rightarrow B \vdash \phi(b)$ for any biteral b , then we also have

$$\mathfrak{B} \models (\forall x \sqsubseteq b)\phi(x) \Rightarrow B \vdash (\forall x \sqsubseteq b)\phi(x)$$

for any biteral b .

Finally, prove (by induction on the structure of ϕ) that we for any Σ -sentence ϕ have $\mathfrak{B} \models \phi \Rightarrow B \vdash \phi$. Use (4) in the base cases, that is, when ϕ is an atomic sentence or a negated atomic sentence. Use the claim and (1) in the case ϕ is of the form $(\forall x \sqsubseteq t)\psi$. The remaining cases are rather straightforward. \square

Corollary 2. *The fragments $\Sigma_{0,m,k}^{\mathfrak{B}}$ and $\Sigma_{0,m,k}^{\mathfrak{D}}$ are decidable (for any $m, k \in \mathbb{N}$).*

Proof. We prove that $\Sigma_{0,m,k}^{\mathfrak{B}}$ is decidable. Let ϕ be a $\Sigma_{0,m,k}$ -formula. The negation of a $\Sigma_{0,m,k}$ -formula is logically equivalent to a $\Sigma_{0,k,m}$ -formula (by De Morgan's laws). We can compute a $\Sigma_{0,k,m}$ -formula ϕ' which is logically equivalent to $\neg\phi$. By Theorem 1, we have $B \vdash \phi$ if $\mathfrak{B} \models \phi$, and we have $B \vdash \phi'$ if $\mathfrak{B} \models \neg\phi$. The set of formulas derivable from the axioms of B is computably enumerable. Hence it is decidable if ϕ is true in \mathfrak{B} . The proof that the fragment $\Sigma_{0,m,k}^{\mathfrak{D}}$ is decidable is similar. \square

3 Normal Forms

A proof of the next lemma can be found several places, see e.g. Büchi and Senger [1] or the proof of *Theorem 6* in Karhumäki et al. [8]. The lemma is also proved in [9].

Lemma 3. *Let $\mathfrak{A} \in \{\mathfrak{B}, \mathfrak{D}\}$, and let s_1, s_2, t_1, t_2 be \mathcal{L}_{BT} -terms. There exist \mathcal{L}_{BT} -terms s, t and variables v_0, \dots, v_k such that*

- (1) $\mathfrak{A} \models (s_1 = t_1 \wedge s_2 = t_2) \leftrightarrow s_1 0 s_2 s_1 1 s_2 = t_1 0 t_2 t_1 1 t_2$
- (2) $\mathfrak{A} \models (s_1 = t_1 \vee s_2 = t_2) \leftrightarrow \exists v_0 \dots v_k [s = t]$
- (3) $\mathfrak{A} \models (\neg s_1 = t_1) \leftrightarrow \exists v_0 \dots v_k [s = t]$.

Lemma 4. *Let s_1, t_1 be \mathcal{L}_{BT} -terms. There exist \mathcal{L}_{BT} -terms s, t and variables v_1, \dots, v_k such that*

- (1) $\mathfrak{D} \models s_1 \preceq t_1 \leftrightarrow \exists v_1 [s_1 v_1 = t_1]$ and (2) $\mathfrak{D} \models (s_1 \not\preceq t_1) \leftrightarrow \exists v_1 \dots v_k [s = t]$.

Proof. It is obvious that (1) holds. Furthermore, the formula $s_1 \not\preceq t_1$ is equivalent in \mathfrak{D} to the formula

$$(t_1 \preceq s_1 \wedge t_1 \neq s_1) \vee \exists xyz[(t_1 = x0y \wedge s_1 = x1z) \vee (t_1 = x1y \wedge s_1 = x0z)].$$

Thus, (2) follows by Lemma 3 and (1). \square

Comment: It is not known to us whether the bounded universal quantifier that appears in clause (2) of the next lemma can be eliminated.

Lemma 5. *Let s_1, t_1 be \mathcal{L}_{BT} -terms. There exist \mathcal{L}_{BT} -terms s, t and variables v_1, \dots, v_k such that (1) $\mathfrak{B} \models s_1 \sqsubseteq t_1 \leftrightarrow \exists v_1 v_2 [t_1 = v_1 s_1 v_2]$ and*

$$(2) \mathfrak{B} \models s_1 \not\sqsubseteq t_1 \leftrightarrow \forall v_1 \sqsubseteq t_1 \exists v_2 \dots v_k [s = t].$$

Proof. Cause (1) is trivial. Furthermore, observe that $s_1 \not\sqsubseteq t_1$ is equivalent in \mathfrak{B} to the formula $(\forall v \sqsubseteq t_1) \alpha$ where α is

$$\begin{aligned} \exists x [t_1 x = v s_1 \wedge x \neq e] \vee \exists xyz [(t_1 = x0y \wedge v s_1 = x1z) \vee \\ (t_1 = x1y \wedge v s_1 = x0z)]. \end{aligned}$$

If we let $v s_1 \preceq t_1$ abbreviate $\exists x [v s_1 x = t]$, then α can be written as $v s_1 \not\preceq t_1$. Thus, (2) follows by Lemma 3. \square

Theorem 6 (Normal Form Theorem I). *Any Σ -formula ϕ is equivalent in \mathfrak{D} to a \mathcal{L}_{BT} -formula ϕ' of the form*

$$\phi' \equiv (\mathbf{Q}_1^{t_1} v_1) \dots (\mathbf{Q}_m^{t_m} v_m) (s = t)$$

where t_1, \dots, t_m, s, t are \mathcal{L}_{BT} -terms and $\mathbf{Q}_j^{t_j} v_j \in \{\exists v_j, \exists v_j \preceq t_j, \forall v_j \preceq t_j\}$ for $j = 1, \dots, m$. Moreover, if ϕ does not contain bounded universal quantifiers, then ϕ' does not contain bounded universal quantifiers.

Proof. We proceed by induction on the structure of ϕ (throughout the proof we reason in the structure \mathfrak{D}). Suppose ϕ is an atomic formula or the negation of an atomic formula. If ϕ is of the form $s = t$, let ϕ' be $s = t$. Use Lemma 3(3) if ϕ is of the form $s \neq t$. Use Lemma 4 if ϕ is of one of the forms $s \preceq t$ and $s \not\preceq t$.

Suppose ϕ is of the form $\alpha \wedge \beta$. By our induction hypothesis, we have formulas

$$\alpha' \equiv (\mathbf{Q}_1^{t_1} x_1) \dots (\mathbf{Q}_k^{t_k} x_k)(s_1 = t_1) \quad \text{and} \quad \beta' \equiv (\mathbf{Q}_1^{s_1} y_1) \dots (\mathbf{Q}_m^{s_m} y_m)(s_2 = t_2)$$

which are equivalent to respectively α and β . Thus, ϕ is equivalent to a formula of the form $(\mathbf{Q}_1^{t_1} x_1) \dots (\mathbf{Q}_k^{t_k} x_k)(\mathbf{Q}_1^{s_1} y_1) \dots (\mathbf{Q}_m^{s_m} y_m)(s_1 = t_1 \wedge s_2 = t_2)$. By Lemma 3(1), we have a formula ϕ' of the desired form which is equivalent to ϕ . The case when ϕ is of the form $\alpha \vee \beta$ is similar. Use clause (2) of Lemma 3 in place of clause (1).

The theorem follows trivially from the induction hypothesis when ϕ is of one of the forms $(\exists v)\alpha$, $(\forall v \preceq t)\alpha$ and $(\exists v \preceq t)\alpha$. \square

Theorem 7 (Normal Form Theorem II). *Any Σ -formula ϕ is equivalent in \mathfrak{B} to a \mathcal{L}_{BT} -formula ϕ' of one of the forms*

$$\phi' \equiv (\mathbf{Q}_1^{t_1} v_1) \dots (\mathbf{Q}_m^{t_m} v_m)(s = t) \quad \text{or} \quad \phi' \equiv (\exists v)(\mathbf{Q}_1^{t_1} v_1) \dots (\mathbf{Q}_m^{t_m} v_m)(s = t)$$

where t_1, \dots, t_m, s, t are \mathcal{L}_{BT} -terms and $\mathbf{Q}_j^{t_j} v_j \in \{\exists v_j \sqsubseteq t_j, \forall v_j \sqsubseteq t_j\}$ for $j = 1, \dots, m$.

Proof. Proceed by induction on the structure of ϕ . This proof is similar to the proof of Theorem 6. A formula of the form $(\forall x \sqsubseteq t)(\exists y)\alpha$ is equivalent (in \mathfrak{B}) to one of the form $(\exists z)(\forall x \sqsubseteq t)(\exists y \sqsubseteq z)\alpha$, a formula of the form $(\exists x \sqsubseteq t)(\exists y)\alpha$ is equivalent to one of the form $(\exists y)(\exists x \sqsubseteq t)\alpha$, and a formula of the form $(\exists x)(\exists y)\alpha$ is equivalent to one of the form $(\exists z)(\exists x \sqsubseteq z)(\exists y \sqsubseteq z)\alpha$. Thus, the resulting normal form will contain maximum one unbounded existential quantifier. \square

Corollary 8. *The fragment $\Sigma_{n,m,0}^{\mathfrak{D}}$ is decidable (for any $n, m \in \mathbb{N}$).*

Proof. By Theorem 6, any $\Sigma_{n,m,0}$ -sentence is equivalent in \mathfrak{D} to a sentence of the normal form $(\exists v_1) \dots (\exists v_k)(s = t)$ (regard the bounded existential quantifiers as unbounded). The transformation of a $\Sigma_{n,m,0}$ -formula into an equivalent formula (in \mathfrak{D}) of normal form is constructive. Makanin [11] has proved that it is decidable whether an equation on the form

$$a_n x_n \dots a_1 x_1 a_0 = b_m y_m \dots b_1 y_1 b_0$$

where $a_1, \dots, a_n, b_1, \dots, b_m \in \{\mathbf{0}, \mathbf{1}\}^*$, has a solution in $\{\mathbf{0}, \mathbf{1}\}^*$. It follows that the fragment $\Sigma_{n,m,0}^{\mathfrak{D}}$ is decidable. \square

We have not been able to prove that any $\Sigma_{n,m,0}$ -sentence is equivalent in \mathfrak{B} to a sentence of the form $(\exists v_1) \dots (\exists v_k)(s = t)$. See the comment immediately before Lemma 5. Thus, we cannot use Makanin's [11] result to prove that the fragment $\Sigma_{n,m,0}^{\mathfrak{B}}$ is decidable.

Open Problem: Is the fragment $\Sigma_{n,m,0}^{\mathfrak{B}}$ decidable (for any $n, m \in \mathbb{N}$)?

4 Undecidable Fragments

Definition 9. Post's Correspondence Problem, *henceforth PCP*, is given by

- *Instance:* a list of pairs $\langle b_1, b'_1 \rangle, \dots, \langle b_n, b'_n \rangle$ where $b_i, b'_i \in \{\mathbf{0}, \mathbf{1}\}^*$
- *Solution:* a finite nonempty sequence i_1, \dots, i_m of indexes such that

$$b_{i_1} b_{i_2} \dots b_{i_m} = b'_{i_1} b'_{i_2} \dots b'_{i_m}.$$

We define the map $N : \{\mathbf{0}, \mathbf{1}\}^* \rightarrow \{\mathbf{0}, \mathbf{1}\}^*$ by $N(\varepsilon) = \varepsilon$, $N(\mathbf{0}) = \mathbf{010}$, $N(\mathbf{1}) = \mathbf{01}^2\mathbf{0}$, $N(b\mathbf{0}) = N(b)N(\mathbf{0})$ and $N(b\mathbf{1}) = N(b)N(\mathbf{1})$.

It is proved in Post [12] that PCP is undecidable. The proof of the next lemma is left to the reader.

Lemma 10. *The instance $\langle b_1, b'_1 \rangle, \dots, \langle b_n, b'_n \rangle$ of PCP has a solution iff the instance $\langle N(b_1), N(b'_1) \rangle, \dots, \langle N(b_n), N(b'_n) \rangle$ has a solution.*

We will now explain the ideas behind our proofs of the next few theorems. Given the lemma above, it is not very hard to see that an instance $\langle g_1, g'_1 \rangle, \dots, \langle g_n, g'_n \rangle$ of PCP has a solution iff there exists a bit string of the form

$$\mathbf{01}^5\mathbf{0}N(a_1)\mathbf{01}^4\mathbf{0}N(b_1)\mathbf{01}^5\mathbf{0} \dots N(a_m)\mathbf{01}^4\mathbf{0}N(b_m)\mathbf{01}^5\mathbf{0} \quad (*)$$

where

- (A) $N(a_m) = N(b_m)$
- (B) $N(a_1) = g_j$ and $N(b_1) = g'_j$ for some $1 \leq j \leq n$
- (C) $N(a_{k+1}) = N(a_k)N(g_j)$ and $N(b_{k+1}) = N(b_k)N(g'_j)$ for some $1 \leq j \leq n$.

We also see that an instance $\langle g_1, g'_1 \rangle, \dots, \langle g_n, g'_n \rangle$ of PCP has a solution iff there exists a bit string s of the form (*) that satisfies

- (a) there is $j \in \{1, \dots, n\}$ such that $\mathbf{01}^5\mathbf{0}N(g_j)\mathbf{01}^4\mathbf{0}N(g'_j)\mathbf{01}^5\mathbf{0}$ is an initial segment of s
- (b) if

$$\mathbf{01}^5\mathbf{0}N(a)\mathbf{01}^4\mathbf{0}N(b)\mathbf{01}^5\mathbf{0}$$

is a substring of s , then either $N(a) = N(b)$, or there is $j \in \{1, \dots, n\}$ such that

$$\mathbf{01}^5\mathbf{0}N(a)N(g_j)\mathbf{01}^4\mathbf{0}N(b)N(g'_j)\mathbf{01}^5\mathbf{0}$$

is a substring of s .

In the proof of Theorem 11 we give a formula which is true in \mathfrak{D} iff there exists a string of the form (*) that satisfies (A), (B) and (C). In the proof of Theorem 12 we give formulas which are true in \mathfrak{B} iff there exists a string of the form (*) that satisfies (a) and (b). In order to improve the readability of our formulas, we will write # in place of the biteral $\overline{01^50}$ and ! in place of the biteral $\overline{01^40}$.

Theorem 11. *The fragment $\Sigma_{3,0,2}^{\mathfrak{D}}$ is undecidable.*

Proof. Let $\psi(x) \equiv (\forall z \preceq x)(z\overline{1^4} \not\preceq x)$. Observe that ψ contains one bounded universal quantifier. Observe that $\psi(\overline{b})$ is true in \mathfrak{D} iff the bit string b does not contain 4 consecutive ones. Furthermore, let $\phi_n(x_1, \dots, x_n, y_1, \dots, y_n) \equiv$

$$\begin{aligned} & (\exists u) \left(\left(\bigvee_{j=1}^n \#x_j!y_j\# \preceq u \right) \wedge \right. \\ & \quad \left. (\forall v \preceq u) \left[v\# \not\preceq u \vee v\# = u \vee (\exists w_1, w_2) \left\{ v\#w_1!w_2\# \preceq u \wedge \right. \right. \right. \\ & \quad \left. \left. \left. \psi(w_1w_2) \wedge \left[w_1 = w_2 \vee \left(\bigvee_{j=1}^n v\#w_1!w_2\#w_1x_j!w_2y_j\# \preceq u \right) \right] \right\} \right] \right). \end{aligned}$$

Let $\langle g_1, g'_1 \rangle, \dots, \langle g_n, g'_n \rangle$ be an instance of PCP. We have

$$\mathfrak{D} \models \phi_n(\overline{N(g_1)}, \dots, \overline{N(g_n)}, \overline{N(g'_1)}, \dots, \overline{N(g'_n)})$$

iff there exists a bit sting of the form (*) that satisfies (A), (B) and (C) iff the instance $\langle g_1, g'_1 \rangle, \dots, \langle g_n, g'_n \rangle$ has a solution. Furthermore ϕ_n is a $\Sigma_{3,0,2}^{\mathfrak{D}}$ -formula. It follows that the fragment $\Sigma_{3,0,2}^{\mathfrak{D}}$ is undecidable. \square

Theorem 12. *The fragments $\Sigma_{1,2,1}^{\mathfrak{B}}$ and $\Sigma_{1,0,2}^{\mathfrak{B}}$ are undecidable.*

Proof. Let $\vec{x} = x_1, \dots, x_n$, let $\vec{y} = y_1, \dots, y_n$ and let

$$\alpha(\vec{x}, \vec{y}, z) \equiv \left(\bigvee_{j=1}^n \#x_j!y_j\# \sqsubseteq z \wedge 0\#x_j!y_j\# \not\sqsubseteq z \wedge 1\#x_j!y_j\# \not\sqsubseteq z \right).$$

Consider the $\Sigma_{1,2,1}$ -formula $\psi_n(\vec{x}, \vec{y}) \equiv$

$$\begin{aligned} & (\exists u) \left(\alpha(\vec{x}, \vec{y}, u) \wedge \right. \\ & \quad \left. (\forall v \sqsubseteq u) \left[\#v\# \not\sqsubseteq u \vee \overline{1^5} \sqsubseteq v \vee (\exists w_1, w_2 \sqsubseteq v) \left\{ v = w_1!w_2 \right. \right. \right. \\ & \quad \left. \left. \left. \wedge \overline{1^4} \not\sqsubseteq w_1 \wedge \overline{1^4} \not\sqsubseteq w_2 \wedge \left[w_1 = w_2 \vee \left(\bigvee_{j=1}^n \#w_1x_j!w_2y_j\# \sqsubseteq u \right) \right] \right\} \right] \right) \end{aligned}$$

and consider the $\Sigma_1^{1,0,2}$ -formula $\gamma_n(\vec{x}, \vec{y}) \equiv$

$$(\exists u) \left(\alpha(\vec{x}, \vec{y}, u) \wedge (\forall w_1, w_2 \sqsubseteq u) \left\{ \#w_1!w_2\# \not\sqsubseteq u \vee \overline{\mathbf{1}^4} \sqsubseteq w_1w_2 \right. \right. \\ \left. \left. \vee w_1 = w_2 \vee \left(\bigvee_{j=1}^n \#w_1x_j!w_2y_j\# \sqsubseteq u \right) \right\} \right).$$

Let $\langle g_1, g'_1 \rangle, \dots, \langle g_n, g'_n \rangle$ be an instance of PCP. We have

$$\mathfrak{B} \models \psi_n(\overline{N(g_1)}, \dots, \overline{N(g_n)}, \overline{N(g'_1)}, \dots, \overline{N(g'_n)})$$

iff

$$\mathfrak{B} \models \gamma_n(\overline{N(g_1)}, \dots, \overline{N(g_n)}, \overline{N(g'_1)}, \dots, \overline{N(g'_n)})$$

iff there exists a bit sting of the form (*) that satisfies (a) and (b) iff the instance $\langle g_1, g'_1 \rangle, \dots, \langle g_n, g'_n \rangle$ has a solution. It follows that the fragments $\Sigma_{1,2,1}^{\mathfrak{B}}$ and $\Sigma_{1,0,2}^{\mathfrak{B}}$ are undecidable. \square

The proof of the next theorem is based on the following idea: The instance $\langle g_1, g'_1 \rangle, \dots, \langle g_n, g'_n \rangle$ of PCP has a solution iff there exists a bit string of the form

$$\mathbf{01}^5\mathbf{0}N(a_1)\mathbf{01}^4\mathbf{0}N(b_1)\mathbf{01}^6\mathbf{0}N(a_2)\mathbf{01}^4\mathbf{0}N(b_2)\mathbf{01}^7\mathbf{0} \dots \\ \dots \mathbf{01}^{5+m-1}\mathbf{0}N(a_m)\mathbf{01}^4\mathbf{0}N(b_m)\mathbf{01}^{5+m}\mathbf{0}$$

with the properties (A), (B) and (C) given above.

Theorem 13. *The fragment $\Sigma_{4,1,1}^{\mathfrak{D}}$ is undecidable.*

Proof. Let $!^k \equiv \overline{\mathbf{01}^k\mathbf{0}}$. The $\Sigma_{4,1,1}$ -formula

$$(\exists u) \left(\left(\bigvee_{j=1}^n !^5x_j!^4y_j!^6 \preceq u \right) \wedge (\forall v \preceq u) \left[v\overline{\mathbf{1}^5\mathbf{0}} \not\preceq u \vee v = \mathbf{0} \vee \right. \right. \\ \left. \left. (\exists w_1, w_2, y)(\exists z \preceq v) \left\{ v = z\mathbf{0}y\overline{\mathbf{1}^5\mathbf{0}}w_1!^4w_2\mathbf{0}1y \wedge 1y = y\mathbf{1} \wedge \right. \right. \right. \\ \left. \left. \left[w_1 = w_2 \vee \left(\bigvee_{j=1}^n v\overline{\mathbf{1}^5\mathbf{0}}w_1x_j!^4w_2y_j\mathbf{0}11y\overline{\mathbf{1}^5\mathbf{0}} \preceq u \right) \right] \right\} \right] \right)$$


yields the desired statement. Note that y is a solution of the equation $1y = y\mathbf{1}$ iff $y \in \{\mathbf{1}\}^*$. \square

References

1. Büchi, J.R., Senger, S.: Coding in the existential theory of concatenation. *Arch. Math. Logik* **26**, 101–106 (1986)
2. Day, J., Ganesh, V., He, P., Manea, F., Nowotka, D.: The satisfiability of extended word equations: the boundary between decidability and undecidability. [arXiv:1802.00523](https://arxiv.org/abs/1802.00523) (2018)
3. Ganesh, V., Minnes, M., Solar-Lezama, A., Rinard, M.: Word equations with length constraints: what's decidable? In: Biere, A., Nahir, A., Vos, T. (eds.) *HVC 2012*. LNCS, vol. 7857, pp. 209–226. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39611-3_21
4. Grzegorzcyk, A.: Undecidability without arithmetization. *Stud. Logica*. **79**, 163–230 (2005)
5. Grzegorzcyk, A., Zdanowski, K.: Undecidability and concatenation. In: Ehrenfeucht, A., et al. (eds.) *Andrzej Mostowski and Foundational Studies*, pp. 72–91. IOS, Amsterdam (2008)
6. Halfon, S., Schnoebelen, P., Zetsche, G.: Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In: *Proceedings of LICS 2017*, pp. 1–12. IEEE Computer Society (2017)
7. Horihata, Y.: Weak theories of concatenation and arithmetic. *Notre Dame J. Formal Log.* **53**, 203–222 (2012)
8. Karhumäki, J., Mignosi, F., Plandowski, W.: The expressibility of languages and relations by word equations. *J. ACM* **47**, 483–505 (2000)
9. Kristiansen, L., Murwanashyaka, J.: Notes on fragments of first-order concatenation theory. [arXiv:1804.06367](https://arxiv.org/abs/1804.06367) (2018)
10. Leary, C., Kristiansen, L.: *A Friendly Introduction to Mathematical Logic*, 2nd edn. Milne Library, SUNY Geneseo, Geneseo (2015)
11. Makanin, G.S.: The problem of solvability of equations in a free semigroup. *Math. USSR-Sbornik* **32**, 129–198 (1977)
12. Post, E.L.: A variant of a recursively unsolvable problem. *Bull. Am. Math. Soc.* **52**, 264–268 (1946)
13. Visser, A.: Growing commas. A study of sequentiality and concatenation. *Notre Dame J. Formal Log.* **50**, 61–85 (2009)



Algorithm Analysis Through Proof Complexity

Massimo Lauria^(✉) 

Dipartimento di Scienze Statistiche, Università “La Sapienza” di Roma,
Rome, Italy

massimo.lauria@uniroma1.it

<http://www.massimolauria.net>

Abstract. Proof complexity can be a tool for studying the efficiency of algorithms. By proving a single lower bound on the length of certain proofs, we can get running time lower bounds for a wide category of algorithms. We survey the proof complexity literature that adopts this approach relative to two NP-problems: k -clique and 3-coloring.

1 Introduction

The problem of *satisfiability* (SAT) is a paradigmatic example of a theoretical problem which arises from concrete applications as diverse as circuit verification, electronic design automation, artificial intelligence, cryptography, operations research, railway signaling systems, etc. Given a formula over Boolean variables and operators \vee, \wedge, \neg , the problem SAT asks whether it is possible to assign Boolean values to the variables so that the formula evaluates to true. SAT is the original NP-complete problem [16, 36], meaning that many important problems can be solved in polynomial time if SAT itself can. Many experts in the field think it cannot, and one of the most difficult open problems in modern mathematics, namely P vs NP, is to determine whether that’s true. Despite their importance, to solve NP-complete problems we do not know anything much faster than just searching exhaustively through the set of “all potential solutions”, which is immensely large. Neither the increasing computational power nor massively parallel architectures can help here: even a 100x speedup is useless when a computation takes 100 billion years.

Still, we need to solve NP-complete problems and there are several concrete algorithms and heuristics designed for this purpose. These simple algorithms do not necessarily exhibit worst-case behavior, therefore it makes sense to study how they perform in practice and which inputs make them inefficient. We argue that proof complexity, i.e., the study of propositional proofs, can be used to do that. The central idea is that in practice most algorithms used to solve NP-complete problems are relatively simple, hence we do not need to tackle P vs NP to study their performance, their limits and new ways to improve them. Together with the answer, these algorithms efficiently compute *logical proofs* of their correctness. Technically we encode an instance of the NP-complete problem

that we want to study as a set of propositional or algebraic constraints so that when the algorithm outputs 0 (i.e. the instance does not belong to the NP set) it simultaneously builds a proof that these constraints are collectively unsatisfiable. The length of this proof is proportional to the running time. For most algorithms this proof has a *simple structure* and is amenable to analysis: we can prove running time lower bounds by proving proof length lower bounds. Since we deal with the proof itself we can drop many details of the algorithm that builds it, hence the same analysis applies simultaneously to many variants of the same algorithm. We showcase the connection between algorithms and proofs using two widely known NP-complete graph problems: 3-coloring and k -clique. First, though, we give basic information about proof complexity and SAT.

The seminal work of Cook and Reckhow [17] laid out the *proof complexity* framework around the question of how large a proof of unsatisfiability can be.¹ Deciding unsatisfiability of a CNF formula (i.e. a conjunction of disjunctions of Boolean variables and their negations) is a coNP-complete problem. If there are CNF formulas with no short proofs of unsatisfiability then we get that $\text{NP} \neq \text{coNP}$ and therefore $\text{P} \neq \text{NP}$, since P is closed under complementation. This leads to the concept of proof system: a language that describes such proofs.

Definition 1 ([17]). A proof system is an algorithm P that takes in input a CNF formula F and a candidate proof π , so that

- $P(F, \pi)$ runs in time polynomial in $|F| + |\pi|$;
- there is a proof π so that $P(F, \pi) = 1$ iff F is UNSAT.

In the literature a proof system is usually described by the shape of the proofs it accepts, e.g. the format of its proof lines, and its inference rules.

The most studied proof system is *resolution* [11].

Definition 2 (Resolution). A resolution proof of unsatisfiability (i.e. a resolution refutation) of a CNF formula $F = \bigwedge_{i=1}^m C_i$ is a sequence of clauses D_1, \dots, D_L so that

- D_L is the empty clause.
- each clause in the sequence is either some C_i for $i \in [m]$ or is obtained from previous clauses in the sequence by the means of the resolution rule

$$\frac{A \vee x \quad B \vee \neg x}{A \vee B} \tag{1}$$

The size (or length) of such resolution refutation is L .

For example a resolution refutation of $C_1 = \neg x$, $C_2 = \neg y$, $C_3 = x \vee y$, is obtained inferring clause x from C_2 and C_3 , and then inferring the empty clause by resolving x with C_1 . The resolution rule is sound, therefore the derivation of the empty clause witnesses the unsatisfiability of F .

¹ The possibility of short proofs for all tautologies was already mentioned by Gödel in a letter to von Neumann [27].

In 1985, Haken showed that there are unsatisfiable formulas with no resolution refutation of polynomial length [29]. The core of most SAT solvers² is essentially an algorithm that, when run on an UNSAT formula, produces a resolution proof of unsatisfiability. Hence no SAT solver based on resolution can run efficiently on Haken’s formula.

For a more general account of proof complexity research we suggest the following books and surveys [8, 32, 44, 45, 53].

2 Resolution Based Algorithms for k -Clique

A k -clique in a graph $G = (V, E)$ is a subset of k vertices $U \subseteq V$ such that all pairs $\{u, v\} \subseteq U$ are in E , i.e., are edges of G . The concept of clique models hidden substructures in larger structure, like a set of mutually compatible constraints, a group of friends in a huge social network, or even patterns in DNA sequences [47]. How hard is to test whether a graph has a k -clique?

When $k \ll n$, an algorithm that checks every possible set of k vertices needs roughly $O(n^k)$ steps. Is this strategy optimal? There are indeed good reasons to believe that any algorithm for k -clique must have running time $n^{\Omega(k)}$ [37], and indeed in some computational models $n^{\Omega(k)}$ operations are required [50–52].

We encode the k -clique problem for a graph $G = (V, E)$ as the CNF formula (2a)–(2c) defined on propositional variables $s_{i,v}$ for $i \in [k]$ and $v \in V$. The Boolean value of variable $s_{i,v}$ indicates whether v is the i th member of the k -clique.

$$\bigvee_{v \in V} s_{i,v} \quad \text{for } i \in [k], \tag{2a}$$

$$\neg s_{i,u} \vee \neg s_{j,v} \quad \text{for } i, j \in [k], u, v \in V, \text{ when } i \neq j \text{ and } \{u, v\} \notin E, \tag{2b}$$

$$\neg s_{i,u} \vee \neg s_{i,v} \quad \text{for } i \in [k], u, v \in V, \text{ when } u \neq v. \tag{2c}$$

The formula is satisfiable if and only if G has a k -clique. Chvátal [14] describes a specialized proof system that captures some algorithms for k -clique that were known at the time. The proofs in this system can be converted efficiently into resolution proofs of essentially the same length.

Many algorithms for k -clique explore the search space in a tree-like fashion, using branch and bound strategies. Bron-Kerbosch [12] is a classic example, but a more recent algorithm of this type is by Östergård [46] and it is the base of his Cliquer software. Both algorithms can be formalized in subsystems of resolution.

Some of these branch and bound algorithms drastically reduce the size of the search space using the fact that when a graph is ℓ -colorable any maximum clique picks at most one element from each color class [41, 49]. To do that they compute partial colorings of the graph. The proof complexity approach is very fruitful here: these colorings must be found quickly, and the heuristics for that

² A SAT solver is a software that decides satisfiability. While nowadays solvers go beyond resolution, their main component still builds resolution proofs.

vary widely in the literature. A lower bound for the resolution proof of CNF formula (2a)–(2c) shows the inefficiency of these coloring based techniques, regardless of the quality of their heuristics.

Beame et al. [7] proved super-polynomial lower bounds on the running time of these algorithms by proving lower bounds for the resolution refutations of the k -clique formula, for the case $k = \Theta(n)$. For the interesting case of $k \ll n$ the problem seems beyond the current proof complexity methods. So far the only results are for restricted forms of resolution.

The hard instance we study is the random graph model $\mathcal{G}(n, p)$ ³. The value $p^* = n^{-\frac{2}{k-1}}$ is a threshold: if $p \ll p^*$ then $G \sim \mathcal{G}(n, p)$ is very unlikely to contain a k -clique; if $p \gg p^*$ then $G \sim \mathcal{G}(n, p)$ is very likely to have one. These graphs are hard for tree-like resolution, a subsystem of resolution where intermediate clauses in the proof must be used at most once as premises.

Theorem 1 ([10]). *For any integer $k > 1$, fix $p = n^{-(1+\varepsilon)\frac{2}{k-1}}$ for some $\varepsilon > 0$. Any tree-like resolution proof of the k -clique formula described in (2a)–(2c) for $G \sim \mathcal{G}(n, p)$ must have length $n^{\Omega(k)}$ with high probability over the sampling of G .*

Very recently [2] made a huge progress extending the result to regular resolution, a subsystem of resolution that is able to capture most of the algorithms mentioned in this section. Hence the following lower bound applies to those algorithms too.

Theorem 2 ([2]). *For any $k \ll \sqrt{n}$, fix $p = n^{-(1+\varepsilon)\frac{2}{k-1}}$ for some $\varepsilon > 0$. Any regular resolution proof of the k -clique formula described in (2a)–(2c) for $G \sim \mathcal{G}(n, p)$ must have length $n^{\Omega(k)}$ with high probability over the sampling of G .*

Both theorems hold for $p = 1/2$ and $k = 2 \log(n)$, too. This is a natural choice of parameters: indeed $G \sim \mathcal{G}(n, 1/2)$ with high probability contains neither a $2 \log(n)$ -clique nor a $2 \log(n)$ -independent set. Such graphs are called *Ramsey graphs*. It is natural to ask whether the previous lower bounds hold for every Ramsey graph (i.e. with probability 1). This is still open, as it is also open whether we can improve Theorem 2 to general resolution. Both questions have been answered in [34, 35], for a very different encoding of (2a)–(2c). This alternative encoding allows one to prove stronger lower bounds but fails to capture the behavior of interesting algorithms.

An even more bold goal is to prove an $n^{\Omega(k)}$ lower bound for geometric proof systems. In this context the k -clique problem is expressed as an integer program, and it is solved by the means of linear or semidefinite programming relaxations. These algorithms are not very practical in general, but may perform better on very tricky instances.

³ The graph has n vertices and the edges are independent $\{0, 1\}$ -valued random variables with expected value p .

3 Algebraic Algorithms for 3-Coloring

Given an undirected graph $G = (V, E)$ and a positive integer k , the graph coloring problem asks whether it is possible to assign a color between 1 and k to each vertex so that no two vertices connected by an edge have the same color. This is a widely studied NP-complete problem, for $k \geq 3$. The fastest algorithm known for 3-coloring runs in time $O(1.3289^n)$ [9], and a survey about various approaches is in [31].

McDiarmid developed a method to prove non-3-colorability that captures many concrete algorithms [42]. This method can be viewed as a way to build resolution proofs that a graph is non-3-colorable. The exponential lower bound on the length of resolution proofs of non-3-colorability by [5] implies for example that every method described in the framework of [42] must take exponential time.

Viewing 3-colorability from the proof complexity perspective, it is interesting to observe that there are various algebraic attacks [1, 38–40]. In particular the thesis [4] introduces the approach of encoding the 3-coloring problem as polynomial equations and then using Hilbert’s Nullstellensatz. The polynomial equations, that are on $\{0, 1\}$ -variables $x_{v,i}$, for $v \in V(G)$ and $i \in \{1, 2, 3\}$, are

$$x_{v,i}^2 - x_{v,i} = 0 \quad \text{for } v \in V(G) \text{ and } i \in \{1, 2, 3\} \tag{3a}$$

$$x_{v,1} + x_{v,2} + x_{v,3} = 1 \quad \text{for } v \in V(G) \tag{3b}$$

$$x_{u,i}x_{v,i} = 0 \quad \text{for } \{u, v\} \in E(G) \text{ and } i \in \{1, 2, 3\}. \tag{3c}$$

Clearly this set of equations has a common solution if and only if G is 3-colorable. Hilbert’s Nullstellensatz claims that the Eqs. (3a)–(3c), enumerated as $f_1 = 0, \dots, f_m = 0$, have no common solution if and only if there are polynomials g_1, \dots, g_m so that

$$\sum_{i=1}^m g_i f_i = 1. \tag{4}$$

Polynomials g_1, \dots, g_m are in fact a so called Nullstellensatz [6] proof of non-3-colorability. The key measure of efficiency of a Nullstellensatz is the maximum degree among $\deg(f_i g_i)$ for $i \in [m]$. Low degree proofs are easy to find using linear algebra.

Fact 1. *Nullstellensatz proofs of degree d for a system of $n^{O(1)}$ polynomial equations over n Boolean variables can be found and output in time $n^{O(d)}$.*

The initial degree of (3a)–(3c) is very low, and yet (as we will see in Theorem 3) there are graphs for which non-3-colorability is unprovable unless the degree of the proof is very large, rendering Fact 1 useless in practice.

There is a more flexible (and efficient) way to show that Eqs. (3a)–(3c) logically imply $1 = 0$, i.e. to show that 1 is in the ideal generated by polynomials (3a)–(3c). This type of proof is based on the computation of Gröbner bases, a concept from commutative algebra that we do not discuss here. It is sufficient to know that a proof of non-3-colorability produced by algorithms based

on either Nullstellensatz or Gröbner bases computation is easily and efficiently represented as proofs in *polynomial calculus* [15].

Definition 3 (Polynomial calculus (PC)). *Given a sequence of polynomials f_1, \dots, f_m over a field \mathbb{F} , and over variables x_1, \dots, x_n , a polynomial calculus refutation of f_1, \dots, f_m is a sequence p_1, \dots, p_L where*

- p_L is the polynomial 1
- each p_j in the sequence is either some f_i for $i \in [m]$ or is obtained from previous polynomials in the sequence by the means of one of the inference rules

$$\frac{q}{xq} \quad \frac{p \ q}{\alpha p + \beta q} \tag{5}$$

where α, β are elements of \mathbb{F} and x is one of the variables.

The degree of a PC refutation is the largest degree among the polynomials p_1, \dots, p_L , and the size of the refutations is the number of monomials that occur.

Nullstellensatz proofs are special cases of PC proofs. Small degree proofs in PC can be found using Buchberger’s algorithm, that computes the Gröbner basis [19] of a system of polynomials.

Fact 2. *Polynomial calculus proofs of degree d for a system of $n^{O(1)}$ polynomials over n Boolean variables can be found and output in time $n^{O(d)}$.*

We stress that a set of polynomials may have a proof of much smaller degree in polynomial calculus than in Nullstellensatz. Indeed there are sets of polynomials over $O(n)$ variables that can be proved in degree $O(1)$ in polynomial calculus but require degree $\Omega(n/\log(n))$ in Nullstellensatz [13].

Another very important observation is that while degree gives an indication regarding how easy it is to find a polynomial calculus proof, there are systems of polynomials over n variables that require degree $\Omega(\sqrt{n})$ proofs and yet have proofs of size $n^{O(1)}$ [26]. Hence the proof search strategy in Fact 2 is not optimal, and it is unlikely that any proof strategy can be [25].

Nullstellensatz and polynomial calculus proofs of non-3-colorability have a special connection with a long series of algorithms [20–24, 30, 43]. They present ways to use algebraic techniques to decide whether a graph is k -colorable, techniques that can be represented in polynomial calculus, and sometimes even in the weaker proof system Nullstellensatz.

Introducing these ideas, De Loera et al. [22, 23] were able to solve some instances previously considered to be tricky, but observed in [24] that these benchmarks actually have Nullstellensatz certificates of very small constant degree. A dramatic improvement in the state of the art of degree lower bound came with [33].

Theorem 3 ([33]). *There is a family of graphs $\{G_n\}_{n \in \mathbb{N}}$ so that*

- each G_n has $O(n)$ vertices and $O(1)$ degree;
- each G_n is not 3-colorable;

- any PC proof of (3a)–(3c) for these graphs requires degree $\Omega(n)$;
- any PC proof of (3a)–(3c) for these graphs requires size $2^{\Omega(n)}$.

The proof of this theorem is essentially the proof that certain reductions between constraint satisfaction problems work in PC. A vast generalization of this result, that applies to general CSP in a wide variety of proof systems, was presented independently in [3].

The algorithms in [22, 23] look for the Nullstellensatz refutation by solving a system of linear equations of size roughly $n^{\Theta(d)}$. If a refutation of degree d exists then the algorithm succeed. The encoding for 3-coloring in these algorithm is different from the one in (3a)–(3c) and while the degree lower bound holds in this alternative encoding, the size lower bound may not. Hence these algorithms still require exponential time on the graphs in Theorem 3, even if it is consistent with the current state of knowledge that there may be small proofs.

What about proofs of non-3-colorability in other proof systems? The graph coloring instances in Theorem 3 are easy for *cutting planes* [18], a proof system that formalizes integer programming techniques. It is open whether there are coloring instances hard for cutting planes. Before concluding, it is worth mentioning Hajós calculus [28], a proof system specifically for non-3-colorability. It is a very powerful computational model and can probably capture most algorithms [48]. It is very unlikely that we can prove any lower bound for it.

4 Conclusion

We discussed some examples of how proof complexity can be used to lower bound the performance of algorithms for combinatorial/graph problems. We can show that specific families of algorithms are inefficient by proving lower bounds for the length of proofs of the appropriate encoding of the problem, in the appropriate proof systems. Naturally there are still problems left open in this line of research. Regarding k -clique the most immediate open problem is to extend the lower bound from regular to general resolution. Regarding 3-coloring, recall that [5] shows resolution lower bounds for a natural distribution of random graphs, while the result in [33] holds for a very specific and artificial graph construction. Extending the lower bound to random graphs is a natural next step. Another open problem related to our discussion is that the proof size lower bound in Theorem 3 does not hold for the alternative encoding in [22]. It seems unlikely that short high-degree proofs of k -colorability could exist just by the virtue of the encoding, but we do not know yet.

References

1. Alon, N., Tarsi, M.: Colorings and orientations of graphs. *Combinatorica* **12**(2), 125–134 (1992)
2. Atserias, A., Bonacina, I., de Rezende, S.F., Lauria, M., Nordström, J., Razborov, A.A.: Clique is hard on average for regular resolution. In: *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2008)* (2018, to appear)

3. Atserias, A., Ochremiak, J.: Proof complexity meets algebra. In: ICALP 2017. Leibniz International Proceedings in Informatics (LIPIcs), vol. 80, pp. 110:1–110:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2017)
4. Bayer, D.A.: The division algorithm and the Hilbert scheme. Ph.D. thesis, Harvard University, Cambridge, MA, USA, June 1982. <https://www.math.columbia.edu/~bayer/papers/Bayer-thesis.pdf>
5. Beame, P., Culberson, J.C., Mitchell, D.G., Moore, C.: The resolution complexity of random graph k -colorability. *Discrete Appl. Math.* **153**(1–3), 25–47 (2005)
6. Beame, P., Impagliazzo, R., Krajíček, J., Pitassi, T., Pudlák, P.: Lower bounds on Hilbert’s Nullstellensatz and propositional proofs. In: Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1994), pp. 794–806, November 1994
7. Beame, P., Impagliazzo, R., Sabharwal, A.: The resolution complexity of independent sets and vertex covers in random graphs. *Comput. Complex.* **16**(3), 245–297 (2007)
8. Beame, P., Pitassi, T.: Propositional proof complexity: past, present, and future. In: *Current Trends in Theoretical Computer Science*, pp. 42–70. World Scientific Publishing (2001)
9. Beigel, R., Eppstein, D.: 3-coloring in time $O(1.3289^n)$. *J. Algorithms* **54**(2), 168–204 (2005)
10. Beyersdorff, O., Galesi, N., Lauria, M.: Parameterized complexity of DPLL search procedures. *ACM Trans. Comput. Log.* **14**(3), 20:1–20:21 (2013). Preliminary version in SAT 2011
11. Blake, A.: Canonical expressions in Boolean algebra. Ph.D. thesis, University of Chicago (1938)
12. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* **16**(9), 575–577 (1973)
13. Buresh-Oppenheim, J., Clegg, M., Impagliazzo, R., Pitassi, T.: Homogenization and the polynomial calculus. *Comput. Complex.* **11**(3–4), 91–108 (2002). Preliminary version in ICALP 2000
14. Chvátal, V.: Determining the stability number of a graph. *SIAM J. Comput.* **6**(4), 643–662 (1977)
15. Clegg, M., Edmonds, J., Impagliazzo, R.: Using the Groebner basis algorithm to find proofs of unsatisfiability. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC 1996), pp. 174–183, May 1996
16. Cook, S.A.: The complexity of theorem proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971), pp. 151–158 (1971)
17. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. *J. Symb. Log.* **44**, 36–50 (1979)
18. Cook, W., Coullard, C.R., Turán, G.: On the complexity of cutting-plane proofs. *Discrete Appl. Math.* **18**(1), 25–38 (1987)
19. Cox, D., Little, J., O’Shea, D.: *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, 3rd edn. Springer, New York (2007). <https://doi.org/10.1007/978-0-387-35651-8>
20. De Loera, J.A.: Gröbner bases and graph colorings. *Beiträge zur Algebra und Geometrie* **36**(1), 89–96 (1995). <https://www.emis.de/journals/BAG/vol.36/no.1/>

21. De Loera, J.A., Margulies, S., Pernpeintner, M., Riedl, E., Rolnick, D., Spencer, G., Stasi, D., Swenson, J.: Graph-coloring ideals: Nullstellensatz certificates, Gröbner bases for chordal graphs, and hardness of Gröbner bases. In: Proceedings of the 40th International Symposium on Symbolic and Algebraic Computation (ISSAC 2015), pp. 133–140, July 2015
22. De Loera, J.A., Lee, J., Malkin, P.N., Margulies, S.: Hilbert’s Nullstellensatz and an algorithm for proving combinatorial infeasibility. In: Proceedings of the 21st International Symposium on Symbolic and Algebraic Computation (ISSAC 2008), pp. 197–206, July 2008
23. De Loera, J.A., Lee, J., Malkin, P.N., Margulies, S.: Computing infeasibility certificates for combinatorial problems through Hilbert’s Nullstellensatz. *J. Symb. Comput.* **46**(11), 1260–1283 (2011)
24. De Loera, J.A., Lee, J., Margulies, S., Onn, S.: Expressing combinatorial problems by systems of polynomial equations and Hilbert’s Nullstellensatz. *Comb. Probab. Comput.* **18**(04), 551–582 (2009)
25. Galesi, N., Lauria, M.: On the automatizability of polynomial calculus. *Theory Comput. Syst.* **47**(2), 491–506 (2010)
26. Galesi, N., Lauria, M.: Optimality of size-degree trade-offs for polynomial calculus. *ACM Trans. Comput. Log.* **12**(1), 4:1–4:22 (2010)
27. Gödel, K.: Ein brief an Johann von Neumann, 20. März, 1956. In: Clote, P., Krajčček, J. (eds.) *Arithmetic, Proof Theory, and Computational Complexity*, pp. 7–9. Oxford University Press, Oxford (1993)
28. Hajós, G.: Über eine konstruktion nicht n-farbbarer graphen. *Wissenschaftliche Zeitschrift der Martin-Luther-Universität Halle-Wittenberg, A* **10**, 116–117 (1961)
29. Haken, A.: The intractability of resolution. *Theoret. Comput. Sci.* **39**, 297–308 (1985)
30. Hillar, C.J., Windfeldt, T.: Algebraic characterization of uniquely vertex colorable graphs. *J. Comb. Theory Ser. B* **98**(2), 400–414 (2008)
31. Husfeldt, T.: Graph colouring algorithms. In: Beineke, L.W., Wilson, R.J. (eds.) *Topics in Chromatic Graph Theory, Encyclopedia of Mathematics and its Applications*, pp. 277–303. Cambridge University Press, May 2015. Chap. 13
32. Krajčček, J.: *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. *Encyclopedia of Mathematics and its Applications*, vol. 60. Cambridge University Press, Cambridge (1995)
33. Lauria, M., Nordström, J.: Graph colouring is hard for algorithms based on Hilbert’s Nullstellensatz and Gröbner bases. In: O’Donnell, R. (ed.) *32nd Computational Complexity Conference (CCC 2017)*. *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 79, pp. 2:1–2:20. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2017)
34. Lauria, M., Pudlák, P., Rödl, V., Thapen, N.: The complexity of proving that a graph is Ramsey. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) *ICALP 2013*. *LNCS*, vol. 7965, pp. 684–695. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39206-1_58
35. Lauria, M., Pudlák, P., Rödl, V., Thapen, N.: The complexity of proving that a graph is Ramsey. *Combinatorica* **37**(2), 253–268 (2017). Preliminary version in *ICALP 2013*
36. Levin, L.A.: Universal sequential search problems. *Probl. Peredachi Informatsii* **9**(3), 115–116 (1973)
37. Lokshitanov, D., Marx, D., Saurabh, S., et al.: Lower bounds based on the exponential time hypothesis. *Bull. EATCS* **3**(105), 41–72 (2013)

38. Lovász, L.: Stable sets and polynomials. *Discrete Math.* **124**(1–3), 137–153 (1994)
39. Matiyasevich, Y.V.: A criterion for vertex colorability of a graph stated in terms of edge orientations. *Diskretnyi Analiz* **26**, 65–71 (1974). <http://logic.pdmi.ras.ru/~yumat/papers/22paper/>. English translation of the Russian original
40. Matiyasevich, Y.V.: Some algebraic methods for calculating the number of colorings of a graph. *J. Math. Sci.* **121**(3), 2401–2408 (2004)
41. McCreesh, C.: Solving hard subgraph problems in parallel. Ph.D. thesis, University of Glasgow (2017)
42. McDiarmid, C.: Colouring random graphs. *Ann. Oper. Res.* **1**(3), 183–200 (1984)
43. Mnuk, M.: Representing graph properties by polynomial ideals. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) *CASC 2001*, pp. 431–444. Springer, Heidelberg (2001). https://doi.org/10.1007/978-3-642-56666-0_33
44. Nordström, J.: A (biased) proof complexity survey for SAT practitioners. In: Sinz, C., Egly, U. (eds.) *SAT 2014*. LNCS, vol. 8561, pp. 1–6. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09284-3_1
45. Nordström, J.: On the interplay between proof complexity and SAT solving. *ACM SIGLOG News* **2**(3), 19–44 (2015)
46. Östergård, P.R.J.: A fast algorithm for the maximum clique problem. *Discrete Appl. Math.* **120**(1), 197–207 (2002)
47. Pevzner, P.A., Sze, S.-H., et al.: Combinatorial approaches to finding subtle signals in DNA sequences. In: *ISMB*, vol. 8, pp. 269–278 (2000)
48. Pitassi, T., Urquhart, A.: The complexity of the Hajós calculus. *SIAM J. Discrete Math.* **8**(3), 464–483 (1995)
49. Prosser, P.: Exact algorithms for maximum clique: a computational study. *Algorithms* **5**(4), 545–587 (2012)
50. Razborov, A.A.: Lower bounds for the monotone complexity of some Boolean functions. *Soviet Math. Dokl.* **31**(2), 354–357 (1985). English translation of a paper in *Doklady Akademii Nauk SSSR*
51. Rossman, B.: On the constant-depth complexity of k -clique. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pp. 721–730. ACM (2008)
52. Rossman, B.: The monotone complexity of k -clique on random graphs. In: *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, 23–26 October 2010, Las Vegas, Nevada, USA, pp. 193–201. IEEE Computer Society (2010)
53. Segerlind, N.: The complexity of propositional proofs. *Bull. Symb. Log.* **13**(4), 417–481 (2007)



Computing with SAT Oracles: Past, Present and Future

Joao Marques-Silva^(✉)

LASIGE, Faculty of Science, University of Lisbon, Lisbon, Portugal
jpms@ciencias.ulisboa.pt

Abstract. Boolean Satisfiability (SAT) epitomizes NP-completeness, and so what is arguably the best known class of intractable problems. NP-complete decision problems are pervasive in all areas of computing, with literally thousands of well-known examples. Nevertheless, SAT solvers routinely challenge the problem's intractability by solving propositional formulas with millions of variables, many representing the translation from some other NP-complete or NP-hard problem. The practical effectiveness of SAT solvers has motivated their use as oracles for NP, enabling new algorithms that solve an ever-increasing range of hard computational problems. This paper provides a brief overview of this ongoing effort, summarizing some of the recent past and present main successes, and highlighting directions for future research.

1 Introduction

Boolean Satisfiability (SAT) represents the decision problem for propositional logic. Given a propositional logic formula, the goal is to decide whether an assignment satisfying the formula exists, or not. SAT is renowned as the first NP-complete decision problem [19], and as such it is the first (complete) member of one of the most studied classes of complexity, NP. Despite being extensively studied, there are still fundamental open questions about the class NP and about NP-complete problems, including whether $P=NP$, i.e. whether the problems that can be solved in deterministic polynomial time match the problems that can be solved in non-deterministic polynomial time, i.e. whose solutions can be certified in deterministic polynomial time. (It is widely believed that this is not the case, with far reaching consequences if it were.)

Many different algorithmic solutions have been envisioned for solving NP-complete and related NP-hard optimization problems. Some of these are remarkable in their depth, representing important inroads, both in the theory of algorithms and in computational complexity [20, 25, 87]. Unfortunately, the practical impact of such solutions has been modest, at best. For some classes of problems, a more pragmatic approach has been adopted, with SAT being one concrete example. Commonly referred to as conflict-driven clause learning (CDCL) [14,

This work was supported by FCT grant ABSOLV (028986/02/SAICT/2017), and LASIGE Research Unit, ref. UID/CEC/00408/2013.

Chap. 4], this class of SAT algorithms has made a visible impact, which has been significant from a theoretical perspective (e.g. in propositional proof complexity), but which has delivered absolutely remarkable performance gains in practice. These days, SAT solvers, i.e. the practical implementation of SAT algorithms, find widespread industrial use and also hundreds of different practical applications.

While there continue to exist improvements made to SAT algorithms [54, 56], some of the key ideas have remain unchanged since the 90s [14, 62, 68]. Nevertheless, this is a time of unprecedented progress in SAT solving. Recent times have witnessed a trend in exploiting SAT solvers as oracles for the class NP, that have enabled solving not only other NP-complete problems, but also NP-hard optimization problems (with NP-complete decision versions) and, perhaps more importantly, problems complete for other classes of complexity, including the first levels of the polynomial hierarchy, but also PSPACE, and even beyond PSPACE. In some cases, the success observed has been paradigm-changing [41, 44, 45].

This paper provides a brief overview of this ongoing effort, which we will refer to as *Computing with SAT Oracles*¹. We will summarize work on solving decision problems, function problems, and enumeration problems, for different classes of complexity. We will highlight the most visible practical results, but will also identify active areas of research. We will also suggest areas where the use of SAT solvers as oracles for NP are posed to provide additional paradigm-changing results, and propose a few challenges aiming at far reaching impact.

The paper is organized as follows. Section 2 briefly overviews the definitions and notation used in the remainder of the paper. The class NP of decision problems is analyzed in Sect. 3. This section also overviews key techniques used for solving large-scale NP-complete decision problems. Decision problems are one example of the computational problems of interest, but many other exist. Section 4 covers function, counting and enumeration problems, and the uses of SAT oracles in these settings. Section 5 addresses problem solving beyond NP, emphasizing the polynomial hierarchy, PSPACE, but also classes beyond PSPACE. A brief overview of ongoing efforts is presented in Sect. 6, highlighting a sample of paradigm-changing challenges. Finally, the paper concludes in Sect. 7.

2 Preliminaries

This section introduces the notation and definitions used throughout the paper.

2.1 Basic Definitions

Standard propositional logic definitions apply (e.g. [14]). CNF formulas are defined over a set of propositional variables. A CNF formula \mathcal{F} is a finite conjunction of clauses, also interpreted as a finite set of clauses. A clause is a disjunction

¹ The paper aims to highlight the many uses of SAT oracles, and so the list of references does not aim to be exhaustive. Additional references can be found in the references cited.

of literals, also interpreted as a set of literals. A literal is a variable or its complement. The standard definitions of assignment, model and satisfiable formula are assumed.

SAT Oracles. There has been extensive recent work on exploiting SAT solvers as oracles. A SAT oracle is modeled as a function call that, in this paper, returns an outcome, which is either true (or SAT), for a satisfiable formula, or false (or UNSAT), for an unsatisfiable formula. For the cases the outcome is true, the SAT oracle also returns a witness of satisfiability μ (i.e. a satisfying truth assignment, mapping variables to values in $\{0, 1\}$). Given a formula \mathcal{F} , a SAT solver call is thus represented as $(st, \mu) \leftarrow \text{SAT}(\mathcal{F})$, where st is either TRUE (or SAT) or FALSE (or UNSAT). Although this paper focuses exclusively on exploiting SAT oracles in practical reasoners, SAT oracles have also recently been used for characterizing classes of complexity in fixed-parameter tractability [22].

Computational Complexity. Standard computational complexity definitions are assumed throughout the paper [28, 75]. The notation used is adapted from [75]. The paper will refer to different classes of complexity, mainly of decision and function problems. For decision problems, well-known classes of complexity include NP, coNP and D^P , PSPACE, EXP, NEXP, etc. For function problems example classes include FP, FNP, etc. In addition, well-known hierarchies are assumed, including the polynomial hierarchy (PH) and its function counterpart (FPH).

2.2 Tools of the Trade

While manipulating SAT solvers as oracles for NP, a number of practicalities must be taken into account when devising practically efficient algorithms.

Exploiting SAT Oracles. In many cases, including solving other NP-complete problems, it suffices to encode the original problem into propositional logic, and invoke a SAT solver. A wealth of encodings have been devised, and these are briefly overview in Sect. 3. Alternatively, the SAT solver internals can be exposed or modified, which enables extending the basic functionality of the solver. For example, this is often done in Satisfiability Modulo Theory (SMT) solving [14, 49, 80], but also in Answer Set Programming (ASP) [29]. In this paper, we mostly focus on the use of the SAT solver as a black-box, where no changes to the oracle are required nor expected, and where changes to the oracle abstraction are kept to a minimum. These are briefly discussed in this section.

Producing Witnesses. Technically, a SAT solver models a more powerful abstraction than an NP oracle. Indeed, if the call to the oracle returns TRUE, the solver can also provide a (total) satisfying assignment, which represents a witness for a formula to be declared satisfiable, enabling certification in polynomial time.

As a result, from a computational complexity perspective, SAT solvers can be viewed as witness-producing oracles for NP [46,60].

Incremental SAT. In most settings, SAT oracles are used incrementally [23]. A working formula is given to the SAT solver, which can then be updated or configured in between oracle calls. Clauses can be temporarily removed using the so-called selection variables, which represent a form of clause half-reification. Selection variables can be turned off or on using assumptions, which are communicated to the solver at the outset of each call. Assumptions can be more general than selecting or de-selecting clauses, enabling for example the specification of concrete assignment to some of the variables.

Cores, Traces, Proofs and Interpolants. In some situations, the SAT solver oracle abstraction is further extended to enable the oracle to produce a *summary* when the call returns FALSE. The *summary* should not be confused with the certificate produced when the oracle returns TRUE. A certificate can be checked in polynomial time. A *summary* indicates that it contains enough information for the oracle to return once again FALSE. Such summaries are referred to as (*unsatisfiable*) *cores* and find a wide range of applications [2]. SAT oracles can also produce traces of their execution [2], from which proofs of unsatisfiability can be generated. From this, interpolants can be computed [78], which also find a wide range of applications [64].

Enabling Parallelization. A recent trend for large scale problem solving with SAT oracles has been the so-called Cube&Conquer approach [35]. At the outset, some procedure identifies a (possibly very large) number of sub-problems to solve, e.g. by heuristically picking some variables and considering all of their assignments, and then solving such sub-problems in parallel [35]. The practical effectiveness of this approach has been demonstrated in different settings, including proving open problems in Mathematics [34].

3 SAT Oracles Within NP

Propositional Encodings. Propositional logic is a fairly restricted knowledge representation language. As a result, many computational problems do not exhibit compact or straightforward representations in propositional logic.

Over the years, different encodings have been devised, to enable mapping more expressive problem domains into SAT. One concrete example is SMT. Although most modern SMT solvers implement the so-called *lazy* approach, where a SAT solver is used as a fast model enumerator [80], an alternative is the *eager* approach, where the SMT formula is encoded into propositional logic [14,49]. Another example is ASP. Similarly to SMT, modern ASP solvers exploit modified SAT solvers [29]. Nevertheless, another approach is to develop encodings from ASP into SAT [40]. The area of constraint programming has also

investigated mappings into SAT [85] and exploiting SAT oracles [74], although most existing CP solvers do not yet extensively borrow from SAT technology. Furthermore, encodings of relational logic into SAT have been investigated [83].

At a lower level of expressivity, encodings of different types of constraints have been investigated. Cardinality constraints, of the form $\sum_i x_i \leq k$, with x_i propositional, represent one concrete example. Different encodings have been devised over the years [5, 14, 24, 81, 86], most of which ensure generalized arc consistency (GAC) through unit propagation (UP).

Similarly, encodings for pseudo-Boolean constraints, of the form $\sum_i a_i x_i \leq b$, with x_i propositional, have been proposed over the years [1, 10, 24, 86], enabling polynomial-size encodings, while ensuring GAC through UP. Efficient handling of cardinality and pseudo-Boolean constraints is of crucial importance in the efficiency of SAT-based approaches for solving different problems.

SAT as the Oracles' Oracle. As suggested above, SAT is pervasive as the engine underlying solvers for more expressive domains, including SMT [49], ASP [29], and in some cases CP [74]. More importantly, SAT oracles now find widespread use in modern theorem provers, with AVATAR representing a paradigmatic example [84]. The significance and impact of modern SAT solvers also motivated proposals for languages aiming at directly encoding problems into SAT [16]. Indirectly, problem solving with SAT oracles enabled the recent discovery of the dual-rail MaxSAT proof system [15, 37].

The Promise of Abstraction. One drawback of exploiting SAT solvers is the often unwieldy problem representation size. In recent years, abstraction-based techniques, including the well-known abstraction refinement paradigm, have enabled solving an ever-increasing range of practical problems. Abstraction is extensively applied in solving computational problems beyond NP [41, 43, 82], and is expected to play a significant role when tackling large-scale problem representations.

4 Beyond Decision Problems

Most naturally-occurring computational problems are not decision problems. We often want some optimal solution, or some minimal set respecting some constraints, or to list the solutions for some set of constraints, or to count the number of solutions. SAT oracles have made important inroads at solving a wide range of computational problems, including function, counting and enumeration problems.

Function Problems. One class of problems are the so-called function (or search) problems, where some solution is to be computed. Concrete examples include minimal set problems, but also optimization problems.

Minimal set problems find important applications, including the analysis of over-constrained systems of constraints [79]. For unsatisfiable formulas, relevant minimal set problems include finding a subset-minimal set of clauses which is unsatisfiable. These subset-minimal sets of clauses are referred to as minimal unsatisfiable subformulas (MUSes). Tightly related with MUSes are the minimal correction subsets (MCSes), i.e. a minimal removal of clauses such that the remaining clauses are satisfiable. There is a well-known minimal hitting set duality relationship between MUSes and MCSes [79]. The computation of MUSes and MCSes has been the subject of remarkable performance improvements over the last decade [8, 9, 11, 30, 57, 65, 66, 69, 73], involving several important new ideas in terms of the algorithms and the optimizations used.

Minimal set problems find other important uses, including computing the lean kernel of an unsatisfiable formula [50] or, for satisfiable formulas, computing the backbone of the formula [42], i.e. the set of variables that exhibit the same value in all models, but also the computation of all prime implicants or implicants [77], with applications in knowledge compilation. Somewhat surprisingly, all these minimal set problems can be solved with essentially the same algorithms based on SAT oracles. Under the name *Minimal Sets of Monotone Predicates* (MSMP) a wealth of different minimal set problems have recently been shown to be tightly related [58, 60].

An arguably better known class of function problems is related with optimization. Concrete examples include maximum satisfiability (MaxSAT), and its different variants, but also minimal satisfiability (MinSAT). In recent years, algorithms based on SAT oracles, and the iterated identification of unsatisfiable cores, enabled remarkable progress in the state of the art of MaxSAT and MinSAT solving [21, 26, 32, 33, 61, 63, 67, 70].

Counting and Enumeration Problems. Counting is a fundamental problem in computer science, with a wide range of applications [14] and associated problems and classes of complexity, exemplified by #SAT and #P [75]. Efficient algorithms for approximate model counting, based on iterative SAT oracle calls, have been studied in recent years [17].

A related problem is the enumeration of solutions respecting some sort of constraint. Enumeration of models is one example. Enumeration of cost-optimum solutions represents another one. But many other examples exist. For example, enumeration of MCSes is of paramount importance when diagnosing systems and streaming through possible system repairs. In a similar vein, the enumeration of MUSes is apparently harder than the enumeration of MCSes, finding important applications in the analysis of over-constrained systems. The most efficient approaches for MUS enumeration are based on exploiting SAT oracles and implicit hitting set dualization [55].

5 Beyond NP

Recent years have witnessed a paradigm shift in solving problems that involve some form of quantification. The integration of SAT oracles with abstraction

refinement solutions enabled the development of efficient approaches for a growing range of problems.

PH and PSPACE. There has been important recent progress on practical algorithms for solving computational problems with decision versions complete for the first levels of the polynomial hierarchy. Concrete examples include Boolean function minimization [39], propositional abduction [36], smallest MUS extraction [55], and k QBF [41, 43]. A similar trend has been observed for PSPACE-complete decision problems, including QBF [41] and modal logics [51], among others.

Beyond PSPACE. There are a wealth of important problems complete for classes beyond PSPACE originating in different areas of application. There are preliminary approaches exploiting SAT oracles for some of these problems. Concrete examples include Effectively Propositional Logic (EPR) [76], Bit-Vector Logic (BVL) [72], theorem provers [27, 48, 84] and model finders [83], model checkers [18], but also SUCCINT-circuit SAT/VAL [75].

6 A Glimpse of the Future

The success of SAT solvers in theorem provers motivates the question: *to which other computational problems can SAT solvers be successfully applied?* This section lists recent successes and proposes a number of challenges for the near future. Solutions to any of these will have far reaching impact.

Recent work has obtained truly impressive performance gains in a number of areas. Concrete examples include axiom pinpointing for the tractable decision logic \mathcal{EL}^+ [4], approximate model counting [17], Boolean function minimization using SAT oracles [39], propositional abduction [36] and also model-based diagnosis [59]. These examples suggest that better understanding of problem domains, but also of the many different ways in which SAT oracles can be used, can enable further significant gains in different areas.

Challenges Within NP. Given the recent trends and successes within NP, but also within PSPACE, one would expect that the efforts of applying SAT and SAT oracles will continue.

Moreover, a different proof system [37] has been proposed, referred to as dual-rail MaxSAT, and which is stronger than resolution but expected to be weaker than the cutting plane proof system [15]. A new proof system that in practice yields efficient algorithms does not necessarily render CDCL obsolete, quite the contrary.

Challenge 1. Devise high performance NP and PSPACE reasoners using the DRMaxSAT proof system, by building on CDCL SAT oracles.

Given the complexity of formulas tackled by SAT solvers, and the vast number of ways in which SAT solvers can be organized, a recent line of research is the application of machine learning techniques to configure the building blocks of a SAT solver [54].

Challenge 2. Devise high performance automatically assembled SAT solvers.

With the explosive growth of machine learning (ML) applications, a general concern is that of explainability [3, 52, 53], i.e. to enable ML models with the capability of associating explanations with classifications which can be understood by human decision makers. Naturally, one would expect logic-based ML models to be more amenable to providing accurate explanations.

Challenge 3. Devise explainable ML models based on SAT oracles.

Examples of recent work on tackling this challenge include [38, 71].

Challenges for PH and PSPACE. Despite the significant progress made in QBF solving in recent years, the scale of formulas that can be solved with QBF solvers is dwarfed by the scale of formulas easily solved by SAT solvers.

Challenge 4. Devise high performance QBF reasoners, by exploiting SAT oracles, that can tackle formulas rivaling in size with those of SAT solvers.

Despite the successes of SAT solvers in the 90s [47], the most effective approaches for automated planning have since been based on heuristic search [31]. A recent alternative exploiting SAT oracles and abstraction [82], although promising, still lags in performance. An immediate question is whether this state of affairs can be changed.

Challenge 5. Devise high performance planners based on SAT oracles.

There has also been recent significant progress for proof systems for QBF [13, 44], but also for DQBF [12], well-known to be NEXP-complete.

Challenge 6. Develop a detailed map of proof complexity for QBF and DQBF.

Challenges Beyond PSPACE. Recent work on axiom pinpointing for tractable description logics suggests that SAT solvers could provide one way to unlock the often-observed performance bottleneck of reasoners for the semantic web [6, 7]. For example, for expressive description logics, most reasoners are still tableaux-based [7].

Challenge 7. Devise high-performance reasoners for expressive description logics using SAT oracles.

Similarly, and with the exception of (grounded) ASP, SAT oracles have not been exploited in logic programming. Concrete examples include reasoners for basic Datalog (and its extensions), but also Prolog.

Challenge 8. Devise a high performance Prolog engine based on SAT oracles.

Over the years, there have been important successes on exploiting SAT solvers in theorem provers. Nevertheless, and with the exception of Inst-Gen [27, 48], the use of SAT solvers is usually one of a number of reasoning techniques.

Challenge 9. Devise a high performance FOL theorem prover based on SAT oracles.

7 Conclusions

The importance of SAT as the de facto NP oracle in a growing range of applications cannot be overstated. This paper provides a brief overview of the many uses of SAT as a witness-producing NP oracle, for solving decision, function, counting and enumeration problems for different classes of complexity, up until semi-decidable problems. The paper also lists challenges that would further extend the reach of SAT solvers as oracles and which, if successful, would have a much wider impact.

References

1. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E., Mayer-Eichberger, V.: A new look at BDDs for pseudo-boolean constraints. *J. Artif. Intell. Res.* **45**, 443–480 (2012)
2. Achá, R.J.A., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Practical algorithms for unsatisfiability proof and core generation in SAT solvers. *AI Commun.* **23**(2–3), 145–157 (2010)
3. Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., Rudin, C.: Learning certifiably optimal rule lists. In: *KDD*, pp. 35–44 (2017)
4. Arif, M.F., Mencía, C., Marques-Silva, J.: Efficient MUS enumeration of horn formulae with applications to axiom pinpointing. In: Heule, M., Weaver, S. (eds.) *SAT 2015*. LNCS, vol. 9340, pp. 324–342. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24318-4_24
5. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality networks: a theoretical and empirical study. *Constraints* **16**(2), 195–221 (2011)
6. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge (2007)
7. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: *An Introduction to Description Logic*. Cambridge University Press, Cambridge (2017)
8. Bacchus, F., Davies, J., Tsimpoukelli, M., Katsirelos, G.: Relaxation search: a simple way of managing optional clauses. In: *AAAI*, pp. 835–841 (2014)
9. Bacchus, F., Katsirelos, G.: Using minimal correction sets to more efficiently compute minimal unsatisfiable sets. In: Kroening, D., Păsăreanu, C.S. (eds.) *CAV 2015*. LNCS, vol. 9207, pp. 70–86. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21668-3_5
10. Bailleux, O., Boufkhad, Y., Roussel, O.: New encodings of Pseudo-Boolean constraints into CNF. In: Kullmann, O. (ed.) *SAT 2009*. LNCS, vol. 5584, pp. 181–194. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02777-2_19

11. Belov, A., Lynce, I., Marques-Silva, J.: Towards efficient MUS extraction. *AI Commun.* **25**(2), 97–116 (2012)
12. Beyersdorff, O., Chew, L., Schmidt, R.A., Suda, M.: Lifting QBF resolution calculi to DQBF. In: Creignou, N., Le Berre, D. (eds.) *SAT 2016*. LNCS, vol. 9710, pp. 490–499. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_30
13. Beyersdorff, O., Pich, J.: Understanding Gentzen and Frege systems for QBF. In: *LICS*, pp. 146–155 (2016)
14. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): *Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications*, vol. 185. IOS Press, Amsterdam (2009)
15. Buss, S., Bonet, M.L., Ignatiev, A., Marques-Silva, J., Morgado, A.: MaxSAT resolution with the dual rail encoding. In: *AAAI*, February 2018
16. Cadoli, M., Schaerf, A.: Compiling problem specifications into SAT. *Artif. Intell.* **162**(1–2), 89–120 (2005)
17. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Algorithmic improvements in approximate counting for probabilistic inference: from linear to logarithmic SAT calls. In: *IJCAI*, pp. 3569–3576 (2016)
18. Cimatti, A., Griggio, A.: Software model checking via IC3. In: Madhusudan, P., Seshia, S.A. (eds.) *CAV 2012*. LNCS, vol. 7358, pp. 277–293. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_23
19. Cook, S.A.: The complexity of theorem-proving procedures. In: *STOC*, pp. 151–158 (1971)
20. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: *Parameterized Algorithms*. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-21275-3>
21. Davies, J., Bacchus, F.: Solving MAXSAT by solving a sequence of simpler SAT instances. In: Lee, J. (ed.) *CP 2011*. LNCS, vol. 6876, pp. 225–239. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23786-7_19
22. de Haan, R., Szeider, S.: The parameterized complexity of reasoning problems beyond NP. In: *KR* (2014)
23. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) *SAT 2003*. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24605-3_37. MiniSat 2.2. <https://github.com/niklasso/minisat.git>
24. Eén, N., Sörensson, N.: Translating pseudo-Boolean constraints into SAT. *JSAT* **2**(1–4), 1–26 (2006)
25. Fomin, F.V., Kaski, P.: Exact exponential algorithms. *Commun. ACM* **56**(3), 80–88 (2013)
26. Fu, Z., Malik, S.: On solving the partial MAX-SAT problem. In: Biere, A., Gomes, C.P. (eds.) *SAT 2006*. LNCS, vol. 4121, pp. 252–265. Springer, Heidelberg (2006). https://doi.org/10.1007/11814948_25
27. Ganzinger, H., Korovin, K.: New directions in instantiation-based theorem proving. In: *LICS*, pp. 55–64 (2003)
28. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)
29. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: *Answer Set Solving in Practice*. Morgan & Claypool Publishers, San Rafael (2012)
30. Grégoire, É., Lagniez, J., Mazure, B.: An experimentally efficient method for (MSS, CoMSS) partitioning. In: *AAAI*, pp. 2666–2673 (2014)
31. Helmert, M.: The fast downward planning system. *J. Artif. Intell. Res.* **26**, 191–246 (2006)

32. Heras, F., Morgado, A., Marques-Silva, J.: Core-guided binary search algorithms for maximum satisfiability. In: AAAI (2011)
33. Heras, F., Morgado, A., Planes, J., Silva, J.P.M.: Iterative SAT solving for minimum satisfiability. In: ICTAI, pp. 922–927 (2012)
34. Heule, M.J.H., Kullmann, O.: The science of brute force. *Commun. ACM* **60**(8), 70–79 (2017)
35. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving very hard problems: cube-and-conquer, a hybrid SAT solving method. In: IJCAI, pp. 4864–4868 (2017)
36. Ignatiev, A., Morgado, A., Marques-Silva, J.: Propositional abduction with implicit hitting sets. In: ECAI, pp. 1327–1335 (2016)
37. Ignatiev, A., Morgado, A., Marques-Silva, J.: On tackling the limits of resolution in SAT solving. In: Gaspers, S., Walsh, T. (eds.) SAT 2017. LNCS, vol. 10491, pp. 164–183. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66263-3_11
38. Ignatiev, A., Pereira, F., Narodytska, N., Marques-Silva, J.: A SAT-based approach to learn explainable decision sets. In: IJCAR (2018)
39. Ignatiev, A., Previti, A., Marques-Silva, J.: SAT-based formula simplification. In: Heule, M., Weaver, S. (eds.) SAT 2015. LNCS, vol. 9340, pp. 287–298. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24318-4_21
40. Janhunen, T., Niemelä, I.: Compact translations of non-disjunctive answer set programs to propositional clauses. In: Balduccini, M., Son, T.C. (eds.) Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning. LNCS (LNAI), vol. 6565, pp. 111–130. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20832-4_8
41. Janota, M., Klieber, W., Marques-Silva, J., Clarke, E.M.: Solving QBF with counterexample guided refinement. *Artif. Intell.* **234**, 1–25 (2016)
42. Janota, M., Lynce, I., Marques-Silva, J.: Algorithms for computing backbones of propositional formulae. *AI Commun.* **28**(2), 161–177 (2015)
43. Janota, M., Marques-Silva, J.: Abstraction-based algorithm for 2QBF. In: Sakallah, K.A., Simon, L. (eds.) SAT 2011. LNCS, vol. 6695, pp. 230–244. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21581-0_19
44. Janota, M., Marques-Silva, J.: Expansion-based QBF solving versus Q-resolution. *Theor. Comput. Sci.* **577**, 25–42 (2015)
45. Janota, M., Marques-Silva, J.: Solving QBF by clause selection. In: IJCAI, pp. 325–331 (2015)
46. Janota, M., Marques-Silva, J.: On the query complexity of selecting minimal sets for monotone predicates. *Artif. Intell.* **233**, 73–83 (2016)
47. Kautz, H.A., Selman, B.: Unifying SAT-based and graph-based planning. In: IJCAI, pp. 318–325 (1999)
48. Korovin, K.: Inst-Gen – a modular approach to instantiation-based automated reasoning. In: Voronkov, A., Weidenbach, C. (eds.) Programming Logics: Essays in Memory of Harald Ganzinger. LNCS, vol. 7797, pp. 239–270. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37651-1_10
49. Kroening, D., Strichman, O.: Decision Procedures - An Algorithmic Point of View, 2nd edn. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-662-50497-0>
50. Kullmann, O., Marques-Silva, J.: Computing maximal autarkies with few and simple oracle queries. In: Heule, M., Weaver, S. (eds.) SAT 2015. LNCS, vol. 9340, pp. 138–155. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24318-4_11
51. Lagniez, J., Berre, D.L., de Lima, T., Montmirail, V.: A recursive shortcut for CEGAR: application to the modal logic K satisfiability problem. In: IJCAI, pp. 674–680 (2017)

52. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: a joint framework for description and prediction. In: KDD, pp. 1675–1684 (2016)
53. Li, O., Liu, H., Chen, C., Rudin, C.: Deep learning for case-based reasoning through prototypes: a neural network that explains its predictions. In: AAAI, February 2018
54. Liang, J.H., Ganesh, V., Poupard, P., Czarnecki, K.: Exponential recency weighted average branching heuristic for SAT solvers. In: AAAI, pp. 3434–3440 (2016)
55. Liffiton, M.H., Previti, A., Malik, A., Marques-Silva, J.: Fast, flexible MUS enumeration. *Constraints* **21**(2), 223–250 (2016)
56. Luo, M., Li, C., Xiao, F., Manyà, F., Lü, Z.: An effective learnt clause minimization approach for CDCL SAT solvers. In: IJCAI, pp. 703–711 (2017)
57. Marques-Silva, J., Heras, F., Janota, M., Previti, A., Belov, A.: On computing minimal correction subsets. In: IJCAI, pp. 615–622 (2013)
58. Marques-Silva, J., Janota, M., Belov, A.: Minimal sets over monotone predicates in Boolean formulae. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 592–607. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_39
59. Marques-Silva, J., Janota, M., Ignatiev, A., Morgado, A.: Efficient model based diagnosis with maximum satisfiability. In: IJCAI, pp. 1966–1972 (2015)
60. Marques-Silva, J., Janota, M., Mencía, C.: Minimal sets on propositional formulae. problems and reductions. *Artif. Intell.* **252**, 22–50 (2017)
61. Marques-Silva, J., Planes, J.: On using unsatisfiability for solving maximum satisfiability. *CoRR*, abs/0712.1097 (2007)
62. Marques-Silva, J., Sakallah, K.A.: GRASP: a search algorithm for propositional satisfiability. *IEEE Trans. Comput.* **48**(5), 506–521 (1999)
63. Martins, R., Joshi, S., Manquinho, V., Lynce, I.: Incremental cardinality constraints for MaxSAT. In: O’Sullivan, B. (ed.) CP 2014. LNCS, vol. 8656, pp. 531–548. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10428-7_39
64. McMillan, K.L.: Interpolation and SAT-based model checking. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 1–13. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45069-6_1
65. Mencía, C., Ignatiev, A., Previti, A., Marques-Silva, J.: MCS extraction with sub-linear oracle queries. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 342–360. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_21
66. Mencía, C., Previti, A., Marques-Silva, J.: Literal-based MCS extraction. In: IJCAI, pp. 1973–1979 (2015)
67. Morgado, A., Dodaro, C., Marques-Silva, J.: Core-guided MaxSAT with soft cardinality constraints. In: O’Sullivan, B. (ed.) CP 2014. LNCS, vol. 8656, pp. 564–573. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10428-7_41
68. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: DAC, pp. 530–535 (2001)
69. Nadel, A., Ryvchin, V., Strichman, O.: Efficient MUS extraction with resolution. In: FMCAD, pp. 197–200 (2013)
70. Narodytska, N., Bacchus, F.: Maximum satisfiability using core-guided MaxSAT resolution. In: AAAI, pp. 2717–2723 (2014)
71. Narodytska, N., Ignatiev, A., Pereira, F., Marques-Silva, J.: Learning optimal decision trees with SAT. In: IJCAI (2018)
72. Niemetz, A., Preiner, M., Biere, A.: Propagation based local search for bit-precise reasoning. *Form. Methods Syst. Des.* **51**(3), 608–636 (2017)
73. Nöhler, A., Biere, A., Egyed, A.: Managing SAT inconsistencies with HUMUS. In: VaMoS, pp. 83–91 (2012)

74. Ohrimenko, O., Stuckey, P.J., Codish, M.: Propagation via lazy clause generation. *Constraints* **14**(3), 357–391 (2009)
75. Papadimitriou, C.H.: *Computational Complexity*. Addison-Wesley, Reading (1993)
76. Piskac, R., de Moura, L.M., Bjørner, N.: Deciding effectively propositional logic using DPLL and substitution sets. *J. Autom. Reason.* **44**(4), 401–424 (2010)
77. Previtì, A., Ignatiev, A., Morgado, A., Marques-Silva, J.: Prime compilation of non-clausal formulae. In: *IJCAI*, pp. 1980–1988 (2015)
78. Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.* **62**(3), 981–998 (1997)
79. Reiter, R.: A theory of diagnosis from first principles. *Artif. Intell.* **32**(1), 57–95 (1987)
80. Sebastiani, R.: Lazy satisfiability modulo theories. *JSAT* **3**(3–4), 141–224 (2007)
81. Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints. In: van Beek, P. (ed.) *CP 2005*. LNCS, vol. 3709, pp. 827–831. Springer, Heidelberg (2005). https://doi.org/10.1007/11564751_73
82. Suda, M.: Property directed reachability for automated planning. *J. Artif. Intell. Res.* **50**, 265–319 (2014)
83. Torlak, E., Jackson, D.: Kodkod: a relational model finder. In: *TACAS*, pp. 632–647 (2007)
84. Voronkov, A.: AVATAR: the architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) *CAV 2014*. LNCS, vol. 8559, pp. 696–710. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_46
85. Walsh, T.: SAT *v* CSP. In: Dechter, R. (ed.) *CP 2000*. LNCS, vol. 1894, pp. 441–456. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45349-0_32
86. Warners, J.P.: A linear-time transformation of linear inequalities into conjunctive normal form. *Inf. Process. Lett.* **68**(2), 63–69 (1998)
87. Woeginger, G.J.: Exact algorithms for NP-hard problems: a survey. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) *Combinatorial Optimization — Eureka, You Shrink!*. LNCS, vol. 2570, pp. 185–207. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36478-1_17



The Isometry Degree of a Computable Copy of ℓ^p

Timothy H. McNicholl¹(✉) and Don Stull^{2,3}

¹ Department of Mathematics, Iowa State University, Ames, IA, USA
mcnichol@iastate.edu

² Department of Computer Science, Iowa State University, Ames, IA, USA
dstull@iastate.edu

³ Laboratoire lorrain de recherche en informatique et ses applications,
Campus scientifique, BP 239, 54506 Vandoeuvre-lés Nancy Cedex, France
donald.stull@inria.fr

Abstract. Suppose p is a computable real so that $p \geq 1$. We define the isometry degree of a computable presentation of ℓ^p to be the least powerful Turing degree \mathbf{d} by which it is \mathbf{d} -computably isometrically isomorphic to the standard presentation of ℓ^p . We show that this degree always exists and that when $p \neq 2$ these degrees are precisely the c.e. degrees.

Keywords: Computable analysis · Computable structure theory
Functional analysis

1 Introduction

Complexity of isomorphisms is a fundamental theme of computable structure theory. For example, a computably presentable structure is *computably categorical* if there is a computable isomorphism between any two of its computable presentations; it is Δ_n^0 -*categorical* if there is a Δ_n^0 isomorphism between any two of its computable copies. The *degree of categoricity* of a computable structure is the least powerful oracle that computes an isomorphism between any two of its computable copies [5]. There is at this time no characterization of the degrees of categoricity. Partial results can be found in [1, 4, 5].

Throughout most of its development, computable structure theory has focused on countable structures. However, there has recently emerged a program to apply the concepts of computable structure theory to the uncountable structures commonly encountered in analysis such as metric spaces and Banach spaces. For example, Melnikov has shown that $C[0, 1]$ is not computably categorical as a metric space [10], and Melnikov and Ng have shown that $C[0, 1]$

D. Stull—Research of the first author supported in part by a Simons Foundation grant # 317870. Research of the second author supported in part by National Science Foundation Grants 1247051 and 1545028.

is not computably categorical as a Banach space [11]. In their seminal text, Pour-El and Richards proved that ℓ^1 is not computably categorical and that ℓ^2 is computably categorical (though the results were not framed in the language of computable structure theory). In 2013 Melnikov asked if ℓ^p is computably categorical for any values of p besides 2 [10]. In 2015 McNicholl answered this question in the negative and later showed that ℓ^p is Δ_2^0 -categorical whenever p is a computable real so that $p \geq 1$ [8,9].

Here we put forward the study of a new notion: the *degree of isomorphism* for a pair $(\mathcal{A}^\#, \mathcal{A}^+)$ of computable presentations of a structure \mathcal{A} ; this is defined to be the least powerful oracle that computes an isomorphism of $\mathcal{A}^\#$ onto \mathcal{A}^+ . This notion fits in with the general theme of studying complexity of isomorphisms and is a local version of the concept of degree of categoricity. If among all computable presentations of \mathcal{A} one is regarded as standard, then we define the isomorphism degree of a single computable presentation $\mathcal{A}^\#$ of \mathcal{A} to be the least power oracle that computes an isomorphism of the standard presentation with $\mathcal{A}^\#$.

We propose to study degrees of isomorphism in the context of the new intersection of computable structure theory and computable analysis, specifically with regard to computable copies of ℓ^p . So, whenever $(\ell^p)^\#$ is a computable presentation of ℓ^p , we define the isometry degree of $(\ell^p)^\#$ to be the least powerful Turing degree that computes a linear isometry of the standard presentation of ℓ^p onto $(\ell^p)^\#$.

It is not obvious that degrees of isomorphism always exist. For example, Miller has produced a computable structure with no degree of computable categoricity [12]. We are thus pleasantly surprised to find that computable presentations of ℓ^p always have an isometry degree and that we can say precisely what these degrees are. Specifically, we prove the following two theorems.

Theorem 1. *When p is a computable real so that $p \geq 1$, every computable presentation of ℓ^p has a degree of isometry, and this degree is c.e.*

Theorem 2. *When p is a computable real so that $p \geq 1$ and $p \neq 2$, the isometry degrees of the computable presentations of ℓ^p are precisely the c.e. degrees.*

One direction of Theorem 2 is already known; namely that every c.e. degree is an isometry degree [9]. However, we give a new proof which we believe is simpler and more intuitive.

The paper is organized as follows. Sections 2 and 3 cover background and preliminaries from functional analysis and computable analysis. Section 4 contains a required result on the complexity of uniformly right-c.e. sequences of reals which is perhaps interesting in its own right. Section 5 contains the new proof that, when $p \neq 2$, every c.e. degree is the isometry degree of a computable presentation of ℓ^p . In Sect. 6, we show that every computable presentation of ℓ^p has a degree of linear isometry and that this degree is c.e.

2 Background

2.1 Arboreal Matters

We use fairly standard terminology and notation for trees (as subsets of $\omega^{<\omega}$). We let ν^- denote the parent of a nonempty node ν . Finally, we say that a function $f : S \rightarrow \mathbb{R}$ is *decreasing* if $f(\nu) > f(\nu')$ whenever $\nu' \in S$ and $\nu \subset \nu'$.

2.2 Background from Functional Analysis

We assume that the field of scalars is the complex numbers although all results hold when the field of scalars is the real numbers. A scalar is *unimodular* if $|\lambda| = 1$.

Recall that a Banach space is a complete normed linear space. A subset of a Banach space \mathcal{B} is *linearly dense* if its linear span is dense in \mathcal{B} .

The simplest example of a Banach space is $\mathbb{C}^{\mathbb{N}}$ where the norm is given by

$$\|(z_1, \dots, z_n)\| = \sqrt{\sum_{j=1}^n |z_j|^2}.$$

Suppose $1 \leq p < \infty$. Recall that ℓ^p is the set of all functions $f : \mathbb{N} \rightarrow \mathbb{C}$ so that $\sum_{n=0}^{\infty} |f(n)|^p < \infty$. When, $f \in \ell^p$, the ℓ^p -norm of f is defined to be

$$\|f\|_p = \left(\sum_{n=0}^{\infty} |f(n)|^p \right)^{1/p}.$$

It is well-known that ℓ^p is a Banach space. For each $n \in \mathbb{N}$, let $e_n = \chi_{\{n\}}$. Then, $\{e_0, e_1, \dots\}$ is the standard basis for ℓ^p .

Suppose that \mathcal{B}_0 and \mathcal{B}_1 are Banach spaces and that $T : \mathcal{B}_0 \rightarrow \mathcal{B}_1$. If there is a constant $C > 0$ so that $\|T(v)\|_{\mathcal{B}_1} \leq C \|v\|_{\mathcal{B}_0}$ for all $v \in \mathcal{B}_0$, then T is *bounded*. If T is linear, then T is continuous if and only if T is bounded. T is an *isomorphism* if it is a linear homeomorphism. T is *isometric* if $\|T(u) - T(v)\|_{\mathcal{B}_1} = \|u - v\|_{\mathcal{B}_0}$ whenever $u, v \in \mathcal{B}_0$. An isometric isomorphism thus preserves the linear and metric structure of the Banach spaces. Finally, if $\mathcal{B}_1 = \mathbb{C}$, then T is a *functional*.

Suppose $1 \leq p < \infty$ and $\frac{1}{p} + \frac{1}{q} = 1$ (i.e. q is the *conjugate* of p). When $f \in \ell^p$ and $g \in \ell^q$, let

$$\langle f, g \rangle = \sum_{n=0}^{\infty} f(n) \overline{g(n)}.$$

When $f \in \ell^q$, let $f^*(g) = \langle g, f \rangle$ for all $g \in \ell^p$. It follows from Hölder's inequality that $|f^*(g)| \leq \|g\|_p \|f\|_q$ and so f^* is a bounded linear functional on ℓ^p .

When $f \in \ell^p$, the *support* of f , which we denote by $\text{supp}(f)$, is the set of all $n \in \mathbb{N}$ so that $f(n) \neq 0$. Vectors $f, g \in \ell^p$ are *disjointly supported* if their supports are disjoint. A subset of ℓ^p is disjointly supported if any two of its elements are disjointly supported. We will utilize the following whose proof is elementary.

Proposition 1. *Suppose $1 \leq p < \infty$ and $\{g_n\}_{n \in \mathbb{N}}$ is a sequence of disjointly supported nonzero vectors of ℓ^p . Then, there is a unique linear isometry $T : \ell^p \rightarrow \ell^p$ so that $T(e_n) = \|g_n\|^{-1} g_n$.*

When $f, g \in \ell^p$, let $\sigma_0(f, g) = |2(\|f\|_p^p + \|g\|_p^p) - \|f + g\|_p^p - \|f - g\|_p^p|$. The following was proven in 1956 by Hanner and independently by Lamperti in 1958 [6, 7].

Proposition 2. *Suppose $1 \leq p < \infty$ and $p \neq 2$. Then, $f, g \in \ell^p$ are disjointly supported if and only if $\sigma_0(f, g) = 0$.*

The following are more or less immediate consequences of Proposition 2. They were first observed by Banach and later rigorously proven by Lamperti [2, 7].

Theorem 3. *Suppose $1 \leq p < \infty$ and $p \neq 2$. If $T : \ell^p \rightarrow \ell^p$ is linear and isometric, then T preserves disjointness of support. That is, $T(f)$ and $T(g)$ are disjointly supported whenever $f, g \in \ell^p$ are disjointly supported.*

Theorem 4. *Suppose p is a real number so that $p \geq 1$ and $p \neq 2$. Let T be a linear map of ℓ^p into ℓ^p . Then, T is an isometric isomorphism if and only if there is a permutation ϕ of \mathbb{N} and a sequence $\{\lambda_n\}_{n \in \mathbb{N}}$ of unimodular scalars so that $T(e_n) = \lambda_n e_{\phi(n)}$ for all n . Furthermore, if ϕ is a permutation of \mathbb{N} , and if $\Lambda = \{\lambda_n\}_{n \in \mathbb{N}}$ is a sequence of unimodular scalars, then there is a unique isometric isomorphism $T_{\phi, \Lambda}$ of ℓ^p so that $T_{\phi, \Lambda}(e_n) = \lambda_n e_{\phi(n)}$ for each $n \in \mathbb{N}$.*

We now summarize some definitions and results from [8]. When $f, g \in \ell^p$, write $f \preceq g$ if and only if $f = g \cdot \chi_A$ for some $A \subseteq \mathbb{N}$. In this case we say f is a *subvector* of g . It follows that the subvector relation is a partial order on ℓ^p . Accordingly, if \mathcal{B} is a subspace of ℓ^p , then $f \in \mathcal{B}$ is an *atom* of \mathcal{B} if there is no $g \in \mathcal{B}$ so that $\mathbf{0} \prec g \prec f$. It follows that f is an atom of ℓ^p if and only if f is a unimodular scalar multiple of a standard basis vector.

Note that f is a subvector of g if and only if f and $g - f$ are disjointly supported. Thus, when $p \neq 2$, the subvector ordering of ℓ^p is preserved by linear isometries.

Suppose S is a tree and $\phi : S \rightarrow \ell^p$. We say ϕ is *separating* if $\phi(\nu)$ and $\phi(\nu')$ are disjointly supported whenever $\nu, \nu' \in S$ are incomparable. We say ϕ is *summative* if for every nonterminal node ν of S , $\phi(\nu) = \sum_{\nu'} \phi(\nu')$ where ν' ranges over the children of ν in S . Finally, we say ϕ is a *disintegration* if it is injective, separating, summative, never zero, and if its range is linearly dense in ℓ^p .

Suppose $\phi : S \rightarrow \ell^p$ is a disintegration. A chain $C \subseteq S$ is *almost norm-maximizing* if whenever $\nu \in C$ is a nonterminal node of S , C contains a child ν' of ν so that

$$\max_{\mu} \|\phi(\mu)\|_p^p \leq \|\nu'\|_p^p + 2^{-|\nu|}$$

where μ ranges over the children of ν in S . The existence of such a child follows from calculus.

The following is proven in [8].

Theorem 5. *Suppose $1 \leq p < \infty$ and $p \neq 2$, and let ϕ be a disintegration of ℓ^p*

1. *If C is an almost norm-maximizing chain of ϕ , then the \preceq -infimum of $\phi[C]$ exists and is either $\mathbf{0}$ or an atom of \preceq . Furthermore, $\inf \phi[C]$ is the limit in the ℓ^p norm of $\phi(\nu)$ as ν traverses the nodes in C in increasing order.*
2. *If $\{C_n\}_{n=0}^\infty$ is a partition of $\text{dom}(\phi)$ into almost norm-maximizing chains, then $\inf \phi[C_0], \inf \phi[C_1], \dots$ are disjointly supported. Furthermore, for each $j \in \mathbb{N}$, there exists a unique n so that $\{j\}$ is the support of $\inf \phi[C_n]$.*

2.3 Background from Computable Analysis

We assume the reader is familiar with the central concepts of computability theory, including computable and computably enumerable sets, Turing reducibility, and enumeration reducibility. These are explained in [3]. We begin with the application of computability concepts to Banach spaces. Our approach is essentially the same as in [13].

A real r is *left (right)-c.e.* if its left (right) Dedekind cut is c.e. A sequence $\{r_n\}_{n \in \mathbb{N}}$ of reals is *uniformly left (right)-c.e.* if the left (right) Dedekind cut of r_n is c.e. uniformly in n .

Let \mathcal{B} be a Banach space. A function $R : \mathbb{N} \rightarrow \mathcal{B}$ is a *structure* on \mathcal{B} if its range is linearly dense in \mathcal{B} . If R is a structure on \mathcal{B} , then (\mathcal{B}, R) is a *presentation* of \mathcal{B} .

A Banach space may have a presentation that is designated as *standard*; such a space is identified with its standard presentation. In particular, if we let $R(n) = e_n$, then (ℓ^p, R) is the standard presentation of ℓ^p . If $R(j)$ is the $(j+1)$ st vector in the standard basis for \mathbb{C}^n when $j < n$, and if $R(j) = \mathbf{0}$ when $j \geq n$, then (\mathbb{C}^n, R) is the standard presentation of \mathbb{C}^n .

Suppose $\mathcal{B}^\# = (\mathcal{B}, R)$ is a presentation of \mathcal{B} . Then, $\mathcal{B}^\#$ induces associated classes of rational vectors and rational open balls as follows. We say $v \in \mathcal{B}$ is a *rational vector* of $\mathcal{B}^\#$ if there exist $\alpha_0, \dots, \alpha_M \in \mathbb{Q}(i)$ so that $v = \sum_{j=0}^M \alpha_j R(j)$. A *rational open ball* of $\mathcal{B}^\#$ is an open ball whose center is a rational vector of $\mathcal{B}^\#$ and whose radius is a positive rational number.

The rational vectors of $\mathcal{B}^\#$ then give rise to associated classes of computable vectors and sequences. A vector $v \in \mathcal{B}$ is a *computable vector* of $\mathcal{B}^\#$ if there is an algorithm that given any $k \in \mathbb{N}$ as input produces a rational vector u of $\mathcal{B}^\#$ so that $\|u - v\|_{\mathcal{B}} < 2^{-k}$. A sequence $\{v_n\}_{n \in \mathbb{N}}$ of vectors of \mathcal{B} is a *computable sequence* of $\mathcal{B}^\#$ if v_n is a computable vector of $\mathcal{B}^\#$ uniformly in n .

When $X \subseteq \mathbb{N}$, the classes of X -computable vectors and X -computable sequences of $\mathcal{B}^\#$ are defined by means of the usual relativizations. If $S \subseteq \mathbb{N}^*$, then the definitions of the classes of computable and X -computable maps from S into $\mathcal{B}^\#$ are similar to the definitions of computable and X -computable sequences of $\mathcal{B}^\#$.

Presentations $\mathcal{B}_0^\#$ and $\mathcal{B}_1^\#$ of Banach spaces \mathcal{B}_0 and \mathcal{B}_1 respectively induce an associated class of computable maps from $\mathcal{B}_0^\#$ into $\mathcal{B}_1^\#$. Namely, a map $T : \mathcal{B}_0 \rightarrow \mathcal{B}_1$ is said to be a *computable map* of $\mathcal{B}_0^\#$ into $\mathcal{B}_1^\#$ if there is a computable function P that maps rational balls of $\mathcal{B}_0^\#$ to rational balls of $\mathcal{B}_1^\#$ so that $T[B_1] \subseteq P(B_1)$

whenever $P(B_1)$ is defined and so that whenever U is a neighborhood of $T(v)$, there is a rational ball B_1 of $\mathcal{B}_1^\#$ so that $v \in B_1$ and $P(B_1) \subseteq U$. In other words, it is possible to compute arbitrarily good approximations of $T(v)$ from sufficiently good approximations of v . This definition relativizes in the obvious way.

When the map T is linear, the following well-known characterization is useful.

Theorem 6. *Suppose $\mathcal{B}_1^\# = (\mathcal{B}_1, R_1)$ is a presentation of a Banach space and $\mathcal{B}_2^\#$ is a presentation of a Banach space. Suppose also that $T : \mathcal{B}_1 \rightarrow \mathcal{B}_2$ is linear. Then, T is an X -computable map of $\mathcal{B}_1^\#$ into $\mathcal{B}_2^\#$ if and only if $\{T(R_1(n))\}_{n \in \mathbb{N}}$ is an X -computable sequence of $\mathcal{B}_2^\#$.*

We say that a presentation $\mathcal{B}^\#$ of a Banach space \mathcal{B} is a *computable presentation* if the norm is a computable map from $\mathcal{B}^\#$ into \mathbb{C} . It follows that the standard presentations of ℓ^p and \mathbb{C} are computable.

For a proof of the following see [15] or Sect. 6.3 of [14].

Proposition 3. *Suppose r is a computable positive number. If f is a computable function on $\overline{D}(0; r)$, and if f has exactly one zero, then this zero is a computable point. Furthermore, this zero can be computed uniformly in f, r .*

The following is proven in [8].

Theorem 7. *Suppose p is a computable real so that $p \geq 1$ and $p \neq 2$. Then, every computable presentation of ℓ^p has a computable disintegration.*

3 Preliminaries

3.1 Preliminaries from Functional Analysis

Let $1 \leq p < \infty$, and suppose f is a unit atom of ℓ^p (i.e. an atom of norm 1). Then, f is also a unit vector of ℓ^q where q is the conjugate of p . So, $|f^*(g)| \leq \|g\|_p$. It also follows that $f^*(g)f \preceq g$ for all $g \in \ell^p$. Suppose g is an atom of ℓ^p . If $f^*(g) = 0$, then f and g are disjointly supported; otherwise $\text{supp}(f) = \text{supp}(g)$ and $f^*(g)f = g$.

The proof of Theorem 2 utilizes the following.

Lemma 1. *Suppose $1 \leq p < \infty$, and suppose $\phi : S \rightarrow \ell^p$ is a disintegration of ℓ^p . Let $C \subseteq S$ be a chain so that whenever $\nu \in C$ is a nonterminal node of S , C contains a child ν' of ν so that*

$$\max\{\|\phi(\mu)\|_p^p : \mu \in \nu_S^\perp\} - \|\phi(\nu')\|_p^p < \min\{\|\phi(\nu)\|_p^p, 2^{-|\nu|}\}.$$

Suppose f is a unit atom of ℓ^p .

1. *If $\inf \phi[C]$ is nonzero, then there is a $\nu \in C$ so that*

$$\|\phi(\nu) - f^*(\phi(\nu))f\|_p^p + \epsilon(\nu) < \|f^*(\phi(\nu))f\|_p^p. \tag{1}$$

2. *If $\nu \in C$ satisfies (1), then $\inf \phi[C] = f^*(\phi(\nu))f$.*

3.2 Preliminaries from Computable Analysis

We first extend some of the results in [8] on partitioning the domain of a disintegration into almost norm-maximizing chains.

Lemma 2. *Suppose $p \geq 1$ is computable and that $(\ell^p)^\#$ is a computable presentation of ℓ^p . Suppose also that ϕ is a computable disintegration of $(\ell^p)^\#$. Then, from a nonterminal node ν of $\text{dom}(\phi)$ and a positive rational number ϵ it is possible to compute a child ν' of ν in $\text{dom}(\phi)$ so that*

$$\max_{\mu} \|\phi(\mu)\|_p^p - \|\phi(\nu')\|_p^p < \epsilon$$

where μ ranges over all children of ν in $\text{dom}(\phi)$.

Theorem 8. *Suppose $p \geq 1$ is computable and let $(\ell^p)^\#$ be a computable presentation of ℓ^p . Suppose also that ϕ is a computable disintegration of $(\ell^p)^\#$ and that $\epsilon : \text{dom}(\phi) \rightarrow (0, \infty)$ is lower semicomputable. Then, there is a partition $\{C_n\}_{n \in \mathbb{N}}$ of $\text{dom}(\phi)$ into uniformly computable chains so that whenever $\nu \in C_n$ is a nonterminal node of $\text{dom}(\phi)$, C_n contains a child ν' of ν so that*

$$\max_{\mu} \|\phi(\mu)\|_p^p - \|\phi(\nu')\|_p^p < \epsilon(\nu)$$

where μ ranges over all children of ν in $\text{dom}(\phi)$.

Proposition 4. *Suppose p is a computable real so that $p \geq 1$, and let $(\ell^p)^\#$ be a computable presentation of ℓ^p . Suppose f is a unit atom of ℓ^p . If f is a computable vector of $(\ell^p)^\#$, then f^* is a computable functional of $(\ell^p)^\#$.*

4 A Compression Theorem

Our proof of Theorem 2 will utilize the following theorem which we believe is interesting in its own right. Roughly speaking, it gives conditions under which the information in a sequence of reals can be compressed into a single real.

Theorem 9. *Let $\{r_n\}_{n \in \mathbb{N}}$ be a sequence of real numbers.*

1. *If $\{r_n\}_{n \in \mathbb{N}}$ is uniformly right-c.e., then there is a right-c.e. real r so that the join of the left Dedekind cuts of the r_n 's is enumeration-equivalent to the left Dedekind cut of r .*
2. *If $\{r_n\}_{n \in \mathbb{N}}$ is uniformly left-c.e., then there is a left-c.e. real r so that the join of the right Dedekind cuts of the r_n 's is enumeration-equivalent to the right Dedekind cut of r .*

5 Every C.E. Degree Is a Degree of Linear Isometry

The following establishes half of Theorem 2. It essentially follows from the proof of the main theorem of [9].

Theorem 10. *If $p \geq 1$ is a computable real so that $p \neq 2$, then every c.e. degree is a degree of isometry of some computable presentation of ℓ^p .*

6 Every Computable Copy of ℓ^p Has a C.E. Degree of Isometry

Suppose $p \geq 1$ is computable, and let $(\ell^p)^\#$ be a computable presentation of ℓ^p . If $p = 2$, then, as mentioned in the introduction, there is a computable isometric isomorphism of ℓ^p onto $(\ell^p)^\#$. So, suppose $p \neq 2$. Let ϕ be a computable disintegration of $(\ell^p)^\#$, and let $S = \text{dom}(\phi)$.

For each $\nu \in S$, let $\epsilon(\nu) = \min\{2^{-|\nu|}, \|\phi(\nu)\|_p^p\}$. Thus, ϵ is computable. It follows from Theorem 8 that there is a partition $\{C_n\}_{n \in \mathbb{N}}$ of S into uniformly computable chains so that for every n and every nonterminal $\nu \in C_n$, C_n contains a child ν' of ν so that

$$\max_{\mu} \|\phi(\mu)\|_p^p - \|\phi(\nu')\|_p^p < \epsilon(\nu)$$

where μ ranges over the children of ν in S . Thus, each C_n is almost norm-maximizing. Let g_n denote the \preceq -infimum of $\phi[C_n]$.

The proof of Theorem 1 uses the following lemmas.

Lemma 3. *If $\{\|g_n\|_p\}_{n \in \mathbb{N}}$ is an X -computable sequence of reals, then X computes an isometric isomorphism of ℓ^p onto $(\ell^p)^\#$.*

Lemma 4. *If X computes an isometric isomorphism of ℓ^p onto $(\ell^p)^\#$, then $\{\|g_n\|_p\}_{n \in \mathbb{N}}$ is an X -computable sequence of reals.*

Proof (Proof of Lemma 3). Suppose $\{\|g_n\|_p\}_{n \in \mathbb{N}}$ is an X -computable sequence of reals.

We first claim that $\{g_n\}_{n \in \mathbb{N}}$ is an X -computable sequence of $(\ell^p)^\#$. For, let $n, k \in \mathbb{N}$ be given. For each $\sigma \in C_n$, $g_n \preceq \phi(\sigma)$, and so $\|\phi(\sigma) - g_n\|_p = \sqrt[p]{\|\phi(\sigma)\|_p^p - \|g_n\|_p^p}$. Thus, for each $\sigma \in C_n$, X computes $\|\phi(\sigma) - g_n\|_p$ uniformly in σ, n . By Theorem 5, there is a $\sigma \in C_n$ so that $\|\phi(\sigma) - g_n\|_p < 2^{-(k+1)}$; using oracle X , such a σ can be found by a search procedure. Since ϕ is computable, we can additionally compute a rational vector f of $(\ell^p)^\#$ so that $\|f - \phi(\sigma)\|_p < 2^{-(k+1)}$. Thus, we have computed a rational vector f of $(\ell^p)^\#$ so that $\|f - g_n\|_p < 2^{-k}$.

Let G denote the set of all $n \in \mathbb{N}$ so that g_n is nonzero. Thus, G is c.e. relative to X . By Theorem 5, for each $j \in \mathbb{N}$ there is a unique $n \in G$ so that $\text{supp}(g_n) = \{j\}$. Thus, G is infinite. So, X computes a one-to-one enumeration $\{n_k\}_{k \in \mathbb{N}}$ of G . Let $h_k = \|g_{n_k}\|_p^{-1} g_{n_k}$. Thus, $\{h_k\}_{k \in \mathbb{N}}$ is an X -computable sequence of $(\ell^p)^\#$.

Again, by Theorem 5, for each $j \in \mathbb{N}$, there is a unique $k \in \mathbb{N}$ so that $\text{supp}(h_k) = \{j\}$. So, there is a permutation ϕ of \mathbb{N} so that $\text{supp}(h_k) = \{\phi(k)\}$ for each $k \in \mathbb{N}$. Since $\|h_k\|_p = 1$, it follows that there is a unimodular scalar λ_k so that $h_k = \lambda_k e_{\phi(k)}$. It then follows from Theorem 4 there is a unique isometric isomorphism T of ℓ^p so that $T(e_k) = h_k$ for all $k \in \mathbb{N}$. So, by Theorem 6, T is an X -computable map of ℓ^p onto $(\ell^p)^\#$.

Proof (Proof of Lemma 4). Let $n, k \in \mathbb{N}$ be given. We compute a rational number q so that $|q - \|g_n\|_p| < 2^{-k}$ as follows. Using oracle X , we search for $\nu \in C_n$ so that either $\|\phi(\nu)\|_p < 2^{-k}$ or so that for some $j \in \mathbb{N}$

$$\|\phi(\nu) - T(e_j)^*(\phi(\nu))T(e_j)\|_p^p + \epsilon(\nu) < \|T(e_j)^*(\phi(\nu))T(e_j)\|_p^p.$$

By Theorem 4, if $g_n \neq \mathbf{0}$, then there exists $j \in \mathbb{N}$ so that $T(e_j)$ and g_n have the same support and so $T(e_j)^*(g_n)T(e_j) = g_n$. So, by Lemma 1.1, this search must terminate. If $\|\phi(\nu)\|_p < 2^{-k}$, since $g_n \preceq \phi(\nu)$, it follows that $\|g_n\|_p < 2^{-k}$ and so we output 0. Otherwise, it follows from Lemma 1.2 that $T(e_j)^*(\phi(\nu))T(e_j) = g_n$. So, using oracle X , we compute and output a rational number q so that $|q - \|T(e_j)^*(\phi(\nu))T(e_j)\|_p| < 2^{-k}$.

Let $r_n = \|g_n\|_p$. Since $g_n \preceq \phi(\nu)$ for all $\nu \in C_n$, $r_n \leq \|\phi(\nu)\|_p$ for all $\nu \in C_n$. Since $g_n = \inf \phi[C_n]$, it follows from Theorem 5 that r_n is right-c.e. uniformly in n . So, by Theorem 9, there is a right-c.e. real r so that the left Dedekind cut of r is enumeration-equivalent to the join of the left Dedekind cuts of the r_n 's. Let D denote the left Dedekind cut of r , and let \mathbf{d} denote the Turing degree of D . Thus, \mathbf{d} is c.e.

We claim that \mathbf{d} is the degree of isometric isomorphism of $(\ell^p)^\#$. For, since $\|g_n\|_p$ is right-c.e. uniformly in n , $\{\|g_n\|_p\}_{n \in \mathbb{N}}$ is a D -computable sequence. Thus, by Lemma 3, D computes an isometric isomorphism of ℓ^p onto $(\ell^p)^\#$. Conversely, suppose an oracle X computes an isometric isomorphism of ℓ^p onto $(\ell^p)^\#$. By Lemma 4, X computes $\{\|g_n\|_p\}_{n \in \mathbb{N}}$. Thus, X computes the left Dedekind cut of r . Therefore, X computes D .

7 Conclusion

For a computable real $p \geq 1$ with $p \neq 2$, we have investigated the least powerful Turing degree that computes a surjective linear isometry of ℓ^p onto one of its computable presentations. We have shown that this degree always exists, and, somewhat surprisingly, that these degrees are precisely the c.e. degrees. Thus computable analysis yields a computability-theoretic property that characterizes the c.e. degrees.

The isometry degree of a pair of computable copies of ℓ^p is an instance of a more general notion of the isomorphism degree of an isomorphic pair of computable structures which is related to the concept of a degree of categoricity. Since there exist computable structures for which there is no degree of categoricity, this leads to the question “Is there a computable structure \mathcal{A} for which there is no degree of computable categoricity but with the property that any two of its computable copies possess a degree of isomorphism?”.

Acknowledgments. We thank U. Andrews, R. Kuyper, S. Lempp, J. Miller, and M. Soskova for very helpful conversations during the first author’s visit to the University of Wisconsin; in particular for suggesting the use of enumeration reducibility. We also thank Diego Rojas for proofreading and making several very useful suggestions. Finally, we thank the reviewers for helpful comments and suggestions.

References

1. Anderson, B., Csima, B.: Degrees that are not degrees of categoricity. *Notre Dame J. Form. Log.* **57**, 389–398 (2016)
2. Banach, S.: *Theory of Linear Operations*. North-Holland Mathematical Library, vol. 38. North-Holland Publishing Co., Amsterdam (1987). Translated from the French by F. Jellett, With comments by A. Pełczyński and Cz. Bessaga
3. Cooper, S.B.: *Computability Theory*. Chapman & Hall/CRC, Boca Raton (2004)
4. Csima, B.F., Franklin, J.N.Y., Shore, R.A.: Degrees of categoricity and the hyperarithmetic hierarchy. *Notre Dame J. Form. Log.* **54**(2), 215–231 (2013)
5. Fokina, E.B., Kalimullin, I., Russell, M.: Degrees of categoricity of computable structures. *Arch. Math. Log.* **49**(1), 51–67 (2010)
6. Hanner, O.: On the uniform convexity of L^p and l^p . *Ark. Mat.* **3**, 239–244 (1956)
7. Lamperti, J.: On the isometries of certain function-spaces. *Pac. J. Math.* **8**, 459–466 (1958)
8. McNicholl, T.H.: Computable copies of ℓ^p . *Computability Preprint (Preprint)*, pp. 1–18
9. McNicholl, T.H.: A note on the computable categoricity of ℓ^p spaces. In: Beckmann, A., Mitrana, V., Soskova, M. (eds.) *CiE 2015. LNCS*, vol. 9136, pp. 268–275. Springer, Cham (2015)
10. Melnikov, A.G.: Computably isometric spaces. *J. Symb. Log.* **78**(4), 1055–1085 (2013)
11. Melnikov, A.G., Ng, K.M.: Computable structures and operations on the space of continuous functions. *Fundamenta Mathematicae* **233**(2), 1–41 (2014)
12. Miller, R.: d -computable categoricity for algebraic fields. *J. Symb. Log.* **74**(4), 1325–1351 (2009)
13. Pour-El, M.B., Richards, J.I.: *Computability in Analysis and Physics. Perspectives in Mathematical Logic*. Springer, Berlin (1989)
14. Weihrauch, K.: *Computable analysis. Texts in Theoretical Computer Science. An EATCS Series*. Springer, Berlin (2000). <https://doi.org/10.1007/978-3-642-56999-9>
15. Ziegler, M.: Effectively open mappings. *J. Complex.* **22**, 827–849 (2006)



Algorithmic Statistics and Prediction for Polynomial Time-Bounded Algorithms

Alexey Milovanov^(✉)

National Research University Higher School of Economics,
Moscow Institute of Physics and Technology, Moscow, Russia
almas239@gmail.com

Abstract. Algorithmic statistics studies explanations of observed data that are good in the algorithmic sense: an explanation should be simple i.e. should have small Kolmogorov complexity and capture all the algorithmically discoverable regularities in the data. However this idea can not be used in practice as is because Kolmogorov complexity is not computable.

In recent years resource-bounded algorithmic statistics were created [7,8]. In this paper we prove a polynomial-time version of the following result of ‘classic’ algorithmic statistics.

Assume that some data were obtained as a result of some unknown experiment. What kind of data should we expect in similar situation (repeating the same experiment)? It turns out that the answer to this question can be formulated in terms of algorithmic statistics [6]. We prove a polynomial-time version of this result under a reasonable complexity theoretic assumption. The same assumption was used by Antunes and Fortnow [1].

1 Introduction

Here we give some basic notation and present results about algorithmic statistics and prediction for general (without resource restrictions) algorithms.

1.1 Algorithmic Statistics

Let x be a binary string, and let A be a finite set of binary strings containing x . Considering A as an “explanation” (statistical model) for x , we want A to be as simple and small as possible. This approach can be made formal in the framework of algorithmic information theory, where the notion of Kolmogorov complexity of a finite object is defined. The definition and basic properties of Kolmogorov complexity can be found in [5,9,11]. Informally Kolmogorov complexity $C(x)$ of a string x is defined as the minimal length of a program that produces x .

We also use another basic notion of the algorithmic information theory, the *discrete a priori probability*. Consider a probabilistic machine V without input that outputs some binary string and stops. It defines a probability distribution

on binary strings: $m_V(x)$ is the probability to get x as the output of V . There exists a universal machine U [5, 11] such that m_U is maximal up to $O(1)$ -factor among all m_V . We fix some U with this property and call $m_U(x)$ the *discrete a priori probability of x* , denoted as $m(x)$. The function m is closely related to Kolmogorov complexity: the value $-\log_2 m(x)$ is equal to $C(x)$ with $O(\log C(x))$ -precision.

Now we can define two parameters that measure the quality of a finite set A as a model for its element x : the complexity $C(A)$ of A and the binary logarithm $\log |A|$ of its size. The first parameter measures how simple is our explanation; the second one measures how specific it is. We use binary logarithms to get both parameters in the same scale: to specify an element of a set of size N we need $\log N$ bits of information.

There is a trade-off between two parameters. The singleton $A = \{x\}$ is a very specific description, but its complexity may be high. On the other hand, for a n -bit string x the set $A = \{0, 1\}^n$ of all n -bit strings is simple, but it is large. To analyze this trade-off, following [3, 4], let us note that every set A containing x leads to a *two-part description of x* : first we specify A using $C(A)$ bits, and then we specify x by its ordinal number in A , using $\log |A|$ bits. In total we need $C(A) + \log |A|$ bits to specify x (plus logarithmic number of bits to separate two parts of the description). This gives the inequality

$$C(x) \leq C(A) + \log |A| + O(\log C(A)).$$

The difference $\delta(x, A) = C(A) + \log |A| - C(x)$

is called *optimality deficiency of A* (as a model for x). As usual in algorithmic statistic, all our statements are made with logarithmic precision (with error tolerance $O(\log n)$ for n -bit strings), so we ignore the logarithmic terms and say that $\delta(x, A)$ is positive and measures the overhead caused by using two-part description based on A instead of the optimal description for x .

One could wonder why we consider only sets as explanations and not general probability distributions (in other terms, why we restrict ourselves to uniform probability distributions). The reason is that this extension is not essential: for every string x and for every distribution μ there exists a set $A \ni x$ explaining x that is almost as good as μ , as the following observation shows:

Proposition 1 ([14]). *For every string x and for every distribution μ there exists a set $A \ni x$ such that $C(A|\mu) \leq O(\log |x|)$ and $\frac{1}{|A|} \geq \frac{1}{2}\mu(x)$.*

There exists another approaches to algorithmic statistics (see [10, 13, 15]) however they are essentially equivalent.

1.2 Prediction Hierarchy

Assume that we have some experimental data represented as a binary string x . We look for a good statistical model for x and find some set A that has small optimality deficiency $\delta(x, A)$. The problem, however, is that many different models with small optimality deficiency may exist for a given x . If we want to cover

all the possibilities, we need to consider the union of all these sets, so we get the following definition.

Definition 1. *Let $x \in \{0, 1\}^n$ be a binary string and let d be some integer. The union of all finite sets of strings $A \subset \{0, 1\}^n$ such that $x \in A$ and $\delta(x, A) \leq d$ is called algorithmic prediction d -neighborhood of x .*

Obviously d -neighborhood increases as d increases.

There is another natural approach to prediction. Since we treat the experiment as a black box (the only thing we know is its outcome x), we assume that the possible models $A \subset \{0, 1\}^n$ are distributed according to their a priori probabilities, and consider the following two-stage process. First, a finite set is selected randomly: a non-empty set A is chosen with probability $m(A)$. Second, a random element x of A is chosen uniformly. In this process every string x is chosen with probability

$$\sum_{A \ni x} m(A)/|A|.$$

For a given pair of strings x and y consider the conditional probability

$$P_x(y) := \Pr[y \in A \mid \text{the output of the two-stage process is } x].$$

Having some string x and some threshold d , we now can consider all strings y such that $P_x(y) \geq 2^{-d}$ (we use the logarithmic scale to facilitate the comparison with algorithmic prediction). These strings could be considered as plausible ones to appear when repeating the experiment of unknown nature that once gave x .

Definition 2. *Let x be a binary string and let d be an integer. The set of all strings y such that $p_x(y) \geq 2^{-d}$ is called probabilistic prediction d -neighborhood of x .*

It turns out that this approach is essentially equivalent to algorithmic prediction neighborhood.

Theorem 1 ([6]). (a) *For every n -bit string x and for every d the algorithmic prediction d -neighborhood is contained in probabilistic prediction $d + O(\log n)$ -neighborhood.*

(b) *For every n -bit string x and for every d the probabilistic prediction d -neighborhood of x is contained in algorithmic prediction $d + O(\log n)$ -neighborhood.*

Our main result is a version of this theorem for time-bounded algorithms.

2 Algorithmic Statistics for Polynomial Time

Here we present our approach to polynomial time-bounded algorithmic statistics. As explanations for strings we consider probability distributions over the set of binary strings. We can not limit ourself by sets (uniform distributions) since an analogue of Proposition 1 for polynomial time-bounded algorithms is unknown.

Let a probability distribution μ be an explanation for a string x . There is a natural parameter measuring how good is μ as an explanation for x , namely $\mu(x)$. Also we need to measure simplicity of μ . A probability distribution is called simple, if it can be sampled by a short probabilistic program with no input in polynomial time. A formal definition can be done by using the notion of *universal machines*—see [8]. There are other ways to measure acceptability of a distribution as explanation to strings [8]. However the way discussed above is the most usable for our investigation.

To measure “simplicity” we will use the notion of time-bounded *prefix-free* Kolmogorov complexity $K^t(x)$. Informally it is defined as the minimal length of a prefix-free program that produces x in at most t steps (see [5] for more details). In fact the difference between prefix free and plain time-bounded complexities is not essential (the plain complexity bounded by time t of a string x is denoted by $C^t(x)$).

Proposition 2 ([5]). *For every string x and for every t there exists c such that:*

- (a) $C^t(x) \leq K^t(x) + c$.
- (b) $K^{ct \log^2 t}(x) \leq C^t(x) + c \log |x|$.

Models of Restricted Type

So far we considered arbitrary distributions as models (statistical hypotheses). However, in practice we usually have some a priori information about the data. We know that the data was obtained by sampling with respect to an unknown probability distribution from a known family of distributions \mathcal{M} .

For example, we can consider the family of uniform distribution on Hamming balls as \mathcal{M} . (That means we know a priori that our string was obtained by flipping certain number of bits in an unknown string.) Restricting the class of allowed hypotheses was initiated in [15].

In our paper we will consider families with the following properties:

- Every element from \mathcal{M} is a distribution on the strings of the same length. The family of distribution on $\{0, 1\}^n$ that belong to \mathcal{M} is denoted by \mathcal{M}_n .
- There exists a polynomial q such that $|\mathcal{M}_n| = 2^{q(n)}$ for every n .
- There exists a polynomial t such that for every $\mu_i \in \mathcal{M}_n$ there exists a program p_i that samples μ_i in time $t(n)$. (This means that for every x of length n the probability of the event “ p_i outputs x ” equals $\mu_i(x)$ and the running time of p_i is at most $t(n)$ for all outcomes of coin tossing.) Moreover there exists a deterministic program $p_{\mathcal{M}}$ that for $i \in \{0, 1\}^{q(n)}$ outputs the program p_i in time $t(n)$.
- For every string x there exists $\mu \in \mathcal{M}$ such that $\mu(x) = 1$. Moreover the program that samples this distribution can be obtained as $p_{\mathcal{M}}(x0^{q(n)-n})$ where n is the length of x .

Any family of distributions that satisfies these four conditions is called *acceptable*. For example, the family of uniform distribution on Hamming balls is acceptable.

If a probability distribution $\mu \in \mathcal{M}$ is sampled by a program $p_i = p(i)$ then it is natural to compare $K^{\text{poly}}(i) - \log \mu(x)$ with $K^{\text{poly}}(x)$ (the difference between these values is an analogue of optimality deficiency in ‘classic’ algorithmic statistics). If $K^{\text{poly}}(i) - \log \mu(x) - K^{\text{poly}}(x) \approx 0$ then μ is called *optimal distribution* for x . Here is a formal definition.

Definition 3. *A distribution μ in an acceptable family \mathcal{M} is called \mathcal{M}, d, t_1, t_2 -optimal for a string x if the distribution μ can be sampled by a probabilistic program $p_{\mathcal{M}}(i) \in \mathcal{M}$ in time t_1 such that*

$$K^{t_1}(i) - \log \mu(x) - K^{t_2}(x) \leq d.$$

3 Prediction Hierarchy in Polynomial Time

Here for a given acceptable family \mathcal{M} we introduce notions of algorithmic and probabilistic prediction neighborhoods. For simplicity first we will consider only families of **uniform** distributions.

Definition 4. *Let $x \in \{0, 1\}^n$, let d, t_1, t_2 be some integers and let \mathcal{M} be an acceptable family of uniform distributions. The set of all strings y such that there exists $\mu \in \mathcal{M}$ such that*

- $\mu(y) > 0$,
- μ is d, t_1, t_2 -optimal for x

is called \mathcal{M} -algorithmic prediction d, t_1, t_2 -neighborhood of x .

Such d, t_1, t_2 -neighborhood increases as d and t_1 increases and t_2 decreases.

To define probabilistic prediction neighborhood we need first to recall the time-bounded version of discrete a priori probability. The t -bounded discrete a priori probability of string x is defined as

$$m^t(x) = 2^{-K^t(x)}.$$

Now we present results that show that this definition is consistent with the unbounded definition.

Definition 5. *A probability distribution σ over $\{0, 1\}^*$ is called P -samplable, if there is randomized machine M so that $\Pr[M \text{ output } x] = \sigma(x)$ and M runs a polynomial time of the length of the output.*

Theorem 2 ([2]). *For every polynomial p , there are a P -samplable distribution σ and a constant c such that for every string x*

$$\sigma(x) \geq \frac{1}{|x|^c} m^p(x).$$

The inequality in the opposite direction holds under the following assumption.

Assumption 1. *There is a set which is decidable by deterministic Turing machines in time $2^{O(n)}$ but is not decidable by deterministic Turing machines in space $2^{o(n)}$ for almost all n .*

Theorem 3 (Lemma 3.2 in [1]). *Under Assumption 1 for every P-samplable probability distribution σ there is number d such that for all x of length n ,*

$$m^{n^d}(x) \geq \frac{\sigma(x)}{n^d}.$$

Now we are ready to define \mathcal{M} -probabilistic prediction neighborhood. Recall that by the 2nd and the 4th properties of acceptability of \mathcal{M} there exists a polynomial q such that every string in $\{0, 1\}^{q(n)}$ defines a distribution in \mathcal{M} .

Consider the following two-stage process for given polynomial t . First a string $s \in \{0, 1\}^{q(n)}$ is selected randomly with probability $m^{t(n)}(s)$. This string s defines a distribution $\mu_s \in \mathcal{M}$. Then a string $x \in \{0, 1\}^n$ is randomly chosen according the distribution μ_s . In this process every string x is chosen with probability

$$\sum_s m^{t(n)}(s)\mu_s(x).$$

Consider the following probability.

$$P_{x,\mathcal{M}}^t(y) = \Pr[\mu_s(y) > 0 \mid \text{the output of the two-stage process is } x]. \tag{1}$$

Note that μ_s is a uniform distribution (now we consider only such families \mathcal{M}).

Definition 6. *Let x be a binary string, let d be an integer, t be a polynomial and \mathcal{M} be an acceptable family. The set of all strings y such that $P_{x,\mathcal{M}}^t(y) \geq 2^{-d}$ is called \mathcal{M} -probabilistic prediction d, t -neighborhood of x .*

Our main result is the following

Theorem 4. (a) *Under Assumption 1 the following holds. For every polynomial t there exists polynomial r such that for every n -bit string x and for every d the \mathcal{M} -algorithmic prediction $d, t(n), r(n)$ -neighborhood of x is contained in \mathcal{M} -probabilistic prediction $d + O(\log n), t$ -neighborhood of x .*

(b) *Under Assumption 1 the following holds. For every polynomial t there exists a polynomial r such that for every n -bit string x and for every d the \mathcal{M} -probabilistic prediction d, t -neighborhood of x is contained in \mathcal{M} -algorithmic prediction $d + O(\log n), r(n), t(n)$ -neighborhood.*

Non-uniform Distribution

Here we extend the notions of algorithmic and probabilistic prediction neighborhoods to arbitrary acceptable family of distribution M . Before we define algorithmic neighborhood note that now the condition $\mu(y) > 0$ is very weak (it is possible that for every y the value $\mu(y)$ is very small but positive). By this reason we have to add a new parameter.

Definition 7. Let $x \in \{0, 1\}^n$, let d, k, t_1, t_2 be some integers and let \mathcal{M} be an acceptable family of distributions. The set of all strings y such that there exists $\mu \in \mathcal{M}$ such that

- $\mu(y) > 2^{-k}$,
- μ is d, t_1, t_2 -optimal for x

is called \mathcal{M} -algorithmic prediction d, k, t_1, t_2 -neighborhood.

Such d, k, t_1, t_2 -neighborhood increases as d, k and t_1 increases and t_2 decreases. To define the probability neighborhood we consider the same 2-stage process. However now we consider another the conditional probability for given x and y .

$$p_{x,h,\mathcal{M}}^r(y) = \Pr[\mu_s(y) > 2^{-h} \mid \text{the output of the two-stage process is } x]. \quad (2)$$

Definition 8. Let x be a binary string, let λ, h be integers, r be a polynomial and \mathcal{M} be an acceptable family. The set of all strings y such that $p_{x,h,\mathcal{M}}^r(y) \geq 2^{-\lambda}$ is called \mathcal{M} -probabilistic prediction λ, h, r -neighborhood of x .

The generalization of Theorem 4 is the following.

Theorem 5. (a) Under Assumption 1 the following holds. For every polynomials t there exists polynomial r such that for every n -bit string x and for every d and k the \mathcal{M} -algorithmic prediction $d, k, t(n), r(n)$ -neighborhood of x is contained in \mathcal{M} -probabilistic prediction λ, h, t -neighborhood of x if $\lambda \geq d - \min(0, h - k) + O(\log n)$.

(b) Under Assumption 1 the following holds. For every polynomial t there exists a polynomial r such that for every n -bit string x and for every d the \mathcal{M} -probabilistic prediction λ, h, t -neighborhood of x is contained in \mathcal{M} -algorithmic prediction $d, k, t(n), r(n)$ -neighborhood of x if $\lambda \geq d + \min(0, h - k) + O(\log n)$.

4 Proof of Theorem 4

Proof (of Theorem 4(a)). This direction is simple. Assume that y belongs to \mathcal{M} -algorithmic prediction $d, t(n), r(n)$ -neighborhood of x . Here r is a polynomial that we will define later. By definition this means that there exists $\mu \in \mathcal{M}$ such that $\mu(y) > 0$ and μ is $d, t(n), r(n)$ -optimal. The later means that for some i the following inequality holds:

$$K^{t(n)}(i) - \log \mu_i(x) - K^{r(n)}(x) \leq d. \quad (3)$$

We need to show that y belongs to \mathcal{M} -probabilistic prediction $d + O(\log n), t$ -neighborhood of x , i.e. $P_{x,\mathcal{M}}^t(y) \geq 2^{-d-O(\log n)}$ (see (1)). By definition (1) can be rewritten as

$$P_{x,\mathcal{M}}^t(y) = \frac{\sum_{s:\mu_s(y)>0} m^{t(n)}(s)\mu_s(x)}{\sum_s m^{t(n)}(s)\mu_s(x)}. \quad (4)$$

Now we choose polynomial r such that the denominator of (4) is not greater than $m^{r(n)}(x)2^{O(\log n)}$. Under Assumption 1 such polynomial r exists. Indeed,

the denominator of (4) defines a P-sample distribution that can be dominated by a polynomial-time bounded discrete a priori probability by Theorem 3.

The sum at the numerator of (4) is not less than one term that obtained by taking $s = i$. So, from (3) it follows that the numerator is not less than $m^{r(n)}(x)2^{-d}$. Hence, $P_{x,\mathcal{M}}^t(y) \geq 2^{-d-O(\log n)}$. Therefore y belongs to \mathcal{M} -probabilistic prediction $d + O(\log n)$, t -neighborhood of x .

We will derive Theorem 4(b) from the following lemma.

Lemma 1. *For every polynomial t under Assumption 1 there exists polynomial r such the following hold. Let x and y be strings of length n and let \mathcal{M} be an acceptable family of distributions. Then there exists string i s. t. $\mu_i(y) > 0$ and*

$$\sum_{s:\mu_s(y)>0} m^{t(n)}(s)\mu_s(x) \leq m^{r(n)}(i)\mu_i(x)2^{O(\log n)}.$$

Proof (of Theorem 4(b) from Lemma 1). Let string y belongs to \mathcal{M} -probabilistic prediction d, t -neighborhood of x , i.e. $P_{x,\mathcal{M}}^t(y) \geq 2^{-d}$. Let us estimate $P_{x,\mathcal{M}}^t(y)$. First note that the denominator of (4) is less than $m^{t(n)}(x)$. Indeed, by the last property of acceptability there exists $\mu_s \in \mathcal{M}$ such that $\mu_s(x) = 1$ and $m^{O(t(n))}(s) = m^{t(n)}(x) + O(1)$. The numerator of (4) can be estimated by Lemma 1 as $m^{r(n)}(i)\mu_i(x)2^{O(\log n)}$ for some string i and polynomial r . So, $\log P_{x,\mathcal{M}}^t(y)$ is less than

$$K^{r(n)}(i) - \log \mu_i(x) - K^{t(n)}(x) + O(\log n).$$

This value is not smaller than d . Hence, y belongs to \mathcal{M} -algorithmic prediction $d + O(\log n), r(n), t(n)$ -neighborhood of x .

Lemma 2. *Let \mathcal{H} be a set of functions from $\{0, 1\}^l$ to $\{0, 1\}^m$ with the following properties.*

- For every l at least $\frac{3}{4}$ of all functions from $\{0, 1\}^l$ to $\{0, 1\}^n$ are in \mathcal{H} .
- For some k there is a Σ_k^P machine with oracle access to a function H on input 1^l will accept exactly when H is in \mathcal{H} .

Then under Assumption 1 there is a polynomial-time computable function $H'(x, r)$ with $x \in \{0, 1\}^l$ and $|r| = O(\log l)$ such that for at least $\frac{2}{3}$ of the possible r , $H_r(x) = H'(x, r)$ is in \mathcal{H} .

Proof (of Lemma 1 from Lemma 2).

The sum $\sum_{s:\mu_s(y)>0} m^{t(n)}(s)\mu_s(x)$ is equal to the sum over all k and j of sums

$$\sum_{\substack{s:\mu_s(y)>0 \\ m^{t(n)}(s)=2^{-k} \\ \mu_s(x)=2^{-j}}} m^{t(n)}(s)\mu_s(x). \tag{5}$$

In fact only $\text{poly}(n)$ of such sums are positive. Indeed, from acceptability of \mathcal{M} it follows that $K^{\text{poly}(n)}(\mu_s)$ is bounded by $\text{poly}(n)$. Also, if $\mu_s(x) < 2^{-\text{poly}(n)}$ then $\mu_s(x) = 0$ since μ_s is sampled by a polynomial time-bounded program. Hence it is enough to show that for every j and k there exists i and polynomial r such that $m^{r(n)}(i)\mu_i(x)2^{O(\log n)}$ is greater than (5) and $\mu_i(y) > 0$.

Denote by $U^{t(n)}$ a function that works as a universal Turing machine U but if U does not outputs anything in $t(n)$ steps then it outputs empty string. Denote by w the logarithm of the number of terms in the sum (5). Denote by \mathcal{H} the set of all functions h from $\{0, 1\}^{k-w+8\log n}$ to $\{0, 1\}^k$ with the following property:

If for a pair of strings (x', y') of length n there exist at least 2^w strings s such that $\mu_s(y') > 0$, $m^{t(n)}(s) = -k$ and $\mu_s(x') = 2^{-j}$ then one of this string is in the image of $U(h)$.

Lemma 3. *At least $\frac{3}{4}$ of all possible functions from $\{0, 1\}^{k-w+8\log n}$ to $\{0, 1\}^k$ belongs to \mathcal{H} .*

Using Lemma 3 we prove the existence of i such that $m(i)\mu_i(x)2^{O(\log n)}$ is greater than (5). (Note that here m is not polynomial-time bounded, so this is not really what we want.) By Lemma 3 *there exists* a function that belongs to \mathcal{H} . The lexicographically first such function has small complexity, because it can be computed given j, k, n and w . Since $h \in \mathcal{H}$ there exists $i' \in \{0, 1\}^{k-w+8\log n}$ such that $\mu_{i'}(x) = 2^{-j}$ and $\mu_{i'}(y) > 0$ where $i = U(h(i'))$. Since $i' \in \{0, 1\}^{k-w+8\log n}$ the complexity of i is not greater than $K(i) \leq k - w + O(\log n)$. A simple calculation shows that $m(i)\mu_i(x)2^{O(\log n)}$ is greater than (5).

To prove the existence of such string i which *polynomial-time bounded* complexity is less than $k - w + O(\log n)$ we need a simple and polynomial-time computable function in \mathcal{H} . To find it we use Lemma 4 for $l = k - w + 8\log n$ and $m = k$. We claim that family \mathcal{H} satisfies properties of Lemma 4. For the first property it is true by Lemma 3. For the second property note that the property $h \in \mathcal{H}$ can written as

$$\forall(x', y')\exists 2^w s : ((\mu_s(x') = 2^{-j}) \wedge (\mu_s(y') > 0)) \Rightarrow \exists i : ((\mu_{U(h(i))}(x') = 2^{-j}) \wedge (\mu_{U(h(i))}(y') > 0)).$$

This property belongs to $P^{\Sigma_p^l}$ for some l since the approximation of the number of certificates belongs to Σ_p^2 [12]. So, there exists polynomial-time computable $H'(x, r)$ such that for some fixed r function $H_r(x) = H'(x, r)$ is in \mathcal{H} . Since $|r| = O(\log n)$ we conclude that there exists simple and polynomial time computable function in \mathcal{H} that complete the proof.

Acknowledgments. This work is supported in parts by the RFBR grant 16-01-00362, by the Young Russian Mathematics award, MK-5379.2018.1 and the RaCAF ANR-15-CE40-0016-01 grant. The study has also been funded by the Russian Academic Excellence Project ‘5-100’.

References

1. Antunes, L., Fortnow, L.: Worst-case running times for average-case algorithms. In: Proceedings of the 24th IEEE Conference on Computational Complexity, pp. 298–303 (2009)
2. Antunes, L., Fortnow, L., Vinodchandran, N.V.: Using depth to capture average-case complexity. In: Lingas, A., Nilsson, B.J. (eds.) FCT 2003. LNCS, vol. 2751, pp. 303–310. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45077-1_28
3. Koppel, M.: Complexity, depth and sophistication. *Complex Syst.* **1**, 1087–1091 (1987)
4. Kolmogorov, A.N.: Talk at the Information Theory Symposium in Tallinn, Estonia (then USSR) (1974)
5. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications. TCS, vol. 3. Springer, New York (2008). <https://doi.org/10.1007/978-0-387-49820-1>
6. Milovanov, A.: Algorithmic statistic, prediction and machine learning. In: Proceedings of 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016). Leibniz International Proceedings in Informatics (LIPIcs), vol. 47, pp. 54:1–54:13 (2016)
7. Milovanov, A.: On algorithmic statistics for space-bounded algorithms. In: Weil, P. (ed.) CSR 2017. LNCS, vol. 10304, pp. 232–244. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58747-9_21
8. Milovanov A., Vereshchagin N.: Stochasticity in algorithmic statistics for polynomial time. In: 32nd Computational Complexity Conference (CCC 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 79, pp. 17:1–17:18 (2017)
9. Shen, A.: Around kolmogorov complexity: basic notions and results. In: Vovk, V., Papadopoulos, H., Gammernan, A. (eds.) Measures of Complexity: Festschrift for Alexey Chervonenkis, pp. 75–115. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21852-6_7. ISBN: 978-3-319-21851-9
10. Shen, A.: The concept of (α, β) -stochasticity in the Kolmogorov sense, and its properties. *Sov. Math. Dokl.* **271**(1), 295–299 (1983)
11. Shen, A., Uspensky, V., Vereshchagin, N.: Kolmogorov Complexity and Algorithmic Randomness. ACM, New York (2017)
12. Stockmeyer, L.: On approximation algorithms for $\#P$. *SIAM J. Comput.* **14**(4), 849–861 (1985)
13. Vereshchagin, N., Shen, A.: Algorithmic statistics: forty years later. In: Day, A., Fellows, M., Greenberg, N., Khousainov, B., Melnikov, A., Rosamond, F. (eds.) Computability and Complexity. LNCS, vol. 10010, pp. 669–737. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-50062-1_41
14. Vereshchagin, N., Vitányi, P.M.B.: Kolmogorov’s structure functions with an application to the foundations of model selection. *IEEE Trans. Inf. Theory* **50**(12), 3265–3290 (2004). Preliminary Version: Proceedings of 47th IEEE Symposium on the Foundations of Computer Science, pp. 751–760 (2002)
15. Vereshchagin, N., Vitányi, P.M.B.: Rate distortion and denoising of individual data using Kolmogorov complexity. *IEEE Trans. Inf. Theory* **56**(7), 3438–3454 (2010)



A C.E. Weak Truth Table Degree Which Is Array Noncomputable and R-maximal

Martin Monath^(✉)

Institut für Informatik, Universität Heidelberg, 69120 Heidelberg, Germany
martin.monath@informatik.uni-heidelberg.de

Abstract. In 1990, Downey, Jockusch and Stob introduced array non-computable sets which capture certain multiple permitting arguments. They completely characterize the simplicity notions which are compatible with array noncomputability. Here we generalize this question and ask for which simplicity properties P there exist c.e. wtt-degrees that contain both sets with property P and array noncomputable sets. By showing that there exists an r-maximal set which is wtt-equivalent to an array noncomputable set we obtain a complete answer to this question for the standard simplicity notions.

1 Introduction

In 1944, Emil Post addressed in his paper [8] his famous question whether there exists an incomplete and noncomputable computably enumerable (c.e.) Turing degree. Today, this is known as *Post's problem*. In his paper, he also introduced the notions of simple, hypersimple and hyperhypersimple sets. Although it is known that one cannot solve Post's problem using (hyper-/hyperhyper-) simple sets, these properties have later been investigated independently and other notions of simplicity have been proposed, e.g. maximal sets by Myhill [7]. An overview of the best known simplicity notions and their relations among each other is given in Fig. 1 (see Soare [9, p. 211]).

Now given a subclass \mathcal{C} of the c.e. sets, it is an interesting question which of the simplicity notions may hold for sets in \mathcal{C} . An example of a class for which this question was investigated is the class of the *array noncomputable* (a.n.c.) sets. These sets were introduced by Downey et al. in [6] and they were classified there

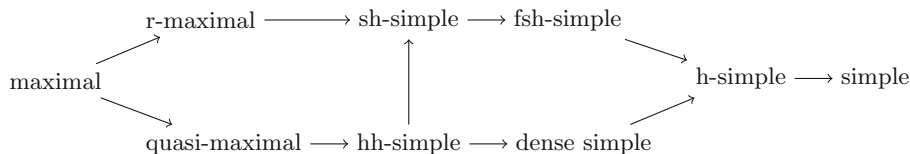


Fig. 1. Most common simplicity properties and their relations among each other. An arrow $P \rightarrow Q$ indicates that property P implies property Q but not vice versa.

as the c.e. sets that allow certain permitting constructions below them, called *multiple permitting*. It turned out that many constructions in computability theory are compatible with the multiple permitting technique and thus can be performed precisely below the a.n.c. degrees, i.e., the c.e. Turing degrees that contain a.n.c. sets. For instance, a.n.c. degrees have deep connections to maximal pairs both in the c.e. and the left-c.e. computable Lipschitz (cl) degrees as shown in [3, 4], respectively.

Now in [6], Downey et al. describe exactly which of the properties in Fig. 1 may hold for a.n.c. sets and which not. On the negative side, they showed that a.n.c. sets cannot be dense simple nor strongly hypersimple. On the positive side, they showed that there is a finitely strongly hypersimple a.n.c. set.

In this paper we address the question if the boundaries are the same if we replace a.n.c. sets by their weak truth table (wtt) degrees. On the negative side, recently Ambos-Spies [1] showed that no a.n.c. wtt-degree contains a dense simple set, thereby extending the corresponding result in [6] from sets to wtt-degrees. In contrast to [6], however, we show here that the positive bound turns out to be stronger for a.n.c. wtt-degrees than for a.n.c. sets. More precisely, we prove the following.

Theorem 1. *There exists a c.e. wtt-degree \mathbf{a} which is a.n.c. and r-maximal.*

By Theorem 1 and the result by Ambos-Spies [1], we completely describe which of the simplicity properties in Fig. 1 may hold for any a.n.c. wtt-degree.

The outline of the paper is as follows. In Sect. 2, we give the basic definitions that are needed for the proof of Theorem 1. In Sect. 3, we give the basic idea of the proof and give the formal construction. Finally, in Sect. 4, we verify that the construction yields a set with the required properties.

2 Preliminaries

In this paper, we follow the notation as given in [9]. In particular, $\{\varphi_e\}_{e \geq 0}$ denotes the standard enumeration of all unary partial computable functions and $\{W_e\}_{e \geq 0}$, where $W_e = \text{dom}(\varphi_e)$ – the domain of φ_e – the standard enumeration of all c.e. sets. Moreover, $\varphi_{e,s}$ denotes the approximation of φ_e within s steps and $W_{e,s} = \text{dom}(\varphi_{e,s})$; finally, we follow the usual convention on converging computations, i.e., for all $e, x, s \geq 0$, if $\varphi_{e,s}(x) \downarrow$ holds then $e, x, \varphi_e(x) < s$. So $W_{e,s} \subseteq \omega \upharpoonright s$ for all e, s .

Now first of all, recall the definition of an r-maximal set.

Definition 1. *A c.e. set A is r-maximal if it is coinfinite and for every computable set R , either $R \cap \bar{A}$ or $\bar{R} \cap \bar{A}$ is finite.*

Next, we give the definition of an array noncomputable set. To that end, we need the notion of a very strong array.

Definition 2 ([6]). A very strong array (v.s.a. for short) is a sequence of finite sets $\mathcal{F} = \{F_n\}_{n \geq 0}$ such that there exists a computable function $f : \omega \rightarrow \omega$ such that for all n , it holds that $F_n = D_{f(n)}$ (i.e., $f(n)$ is the canonical index of F_n), $0 < |F_n| < |F_{n+1}|$ and $F_m \cap F_n = \emptyset$ for all $m \neq n$. Moreover, a v.s.a. $\mathcal{F} = \{F_n\}_{n \geq 0}$ is called a very strong array of intervals (v.s.a.i.) if each F_n is an interval such that $\max(F_n) < \min(F_{n+1})$ holds for all n .

Based on very strong arrays, array noncomputability is defined for c.e. sets as follows.

Definition 3. Given a v.s.a. $\mathcal{F} = \{F_n\}_{n \geq 0}$, a c.e. set A is \mathcal{F} -array non-computable (\mathcal{F} -a.n.c.) if for all c.e. sets W , there exists $n \geq 0$ such that $W \cap F_n = A \cap F_n$; A is called array noncomputable (a.n.c.) if it is \mathcal{F} -a.n.c. for some v.s.a. \mathcal{F} ; and a wtt-degree \mathbf{a} is array noncomputable if \mathbf{a} contains an array noncomputable set.

Note that in the original definition of very strong arrays given in [6], it is required that very strong arrays partition ω . Definition 2 follows the one given in [5]. However, this does not affect the notion of array noncomputability for wtt-degrees. Namely, as shown in [2], every a.n.c. set A in the sense of Definition 3 is \mathcal{F} -a.n.c. for a v.s.a. \mathcal{F} with $\bigcup_{n \in \omega} F_n = \omega$. For the proof of Theorem 1, we use a characterization of the a.n.c. wtt-degrees given by [1].

Definition 4 ([1]). Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be a v.s.a., let f be a computable function, let A be c.e. set, and let $\{A_s\}_{s \geq 0}$ be computable enumeration of A . Then A is \mathcal{F} -permitting via f and $\{A_s\}_{s \geq 0}$ if, for any partial computable function ψ ,

$$\exists^\infty n \forall x \in F_n (\psi(x) \downarrow \Rightarrow A \upharpoonright f(x) + 1 \neq A_{\psi(x)} \upharpoonright f(x) + 1) \tag{1}$$

holds. A is \mathcal{F} -permitting via f if there is a computable enumeration $\{A_s\}_{s \geq 0}$ of A such that A is \mathcal{F} -permitting via f and $\{A_s\}_{s \geq 0}$; A is \mathcal{F} -permitting if A is \mathcal{F} -permitting via some computable f ; and A is multiply permitting if A is \mathcal{F} -permitting for some v.s.a. \mathcal{F} . Finally, a c.e. wtt-degree \mathbf{a} is multiply permitting if there is a multiply permitting set $A \in \mathbf{a}$.

As shown in [1], the c.e. multiply permitting wtt-degrees coincide with the a.n.c. wtt-degrees.

Theorem 2. For a c.e. wtt-degree \mathbf{a} , the following are equivalent.

1. \mathbf{a} is a.n.c.
2. \mathbf{a} is multiply permitting.
3. Every c.e. set $A \in \mathbf{a}$ is multiply permitting.

So for the sake of Theorem 1, it suffices to construct a c.e. set A which is both multiply permitting and r-maximal. Before we state the formal construction of such a set A , let us give some idea of the proof.

3 Idea of the Proof of Theorem 1

The construction of A is divided into two parts. First, we construct a c.e. set B such that B is r -maximal and such that the complement \bar{B} is “big enough” (which is made precise in (2) below). Then we define a v.s.a. \mathcal{F} and construct A as a c.e. superset of B such that A is \mathcal{F} -permitting via the identity function. So first let us make precise what B looks like.

Lemma 1. *There exists a v.s.a.i. $\mathcal{G} = \{G_n\}_{n \geq 0}$ and an r -maximal set B such that*

$$\exists^\infty n (|G_n \cap \bar{B}| \geq (n + 1)^2). \tag{2}$$

We claim that from Lemma 1, we can define a c.e. set A as required.

Proof (of Theorem 1 using Lemma 1). Fix $\mathcal{G} = \{G_n\}_{n \geq 0}$ and B as in Lemma 1 and fix a computable enumeration $\{B_s\}_{s \geq 0}$ of B . Let $\mathcal{F} = \{F_n\}_{n \geq 0}$ be the unique v.s.a. such that $F_n = \{x_0^n, \dots, x_n^n\}$, where x_0^n, \dots, x_n^n are the first $n + 1$ elements of G_{n+1} in order of magnitude (note that $F_n \subseteq G_{n+1}$ since $|G_n| \geq n + 1$ for all n ; hence, $\max(G_n) < \min(F_n)$ as \mathcal{G} is a v.s.a.i.). Then we define a computable enumeration $\{A_s\}_{s \geq 0}$ of A in stages s as follows, where A_s denotes the finite set of numbers that are enumerated into A by stage s .

Construction of A .

Stage 0. $A_0 = \emptyset$.

Stage $s + 1$. Given A_s , let N_s be the set of all $n \in \omega$ such that

- (a) $G_n \not\subseteq A_s$ and
 - (b) $\exists e, x (e \leq n \ \& \ x \in F_n \ \& \ \varphi_{e,s}(x) \uparrow \ \& \ \varphi_{e,s+1}(x) \downarrow)$
- hold (note that $N_s \subseteq \omega \upharpoonright s$). Let $A_{s+1} = B_{s+1} \cup A_s \cup \{\min(G_n \setminus A_s) : n \in N_s\}$.

We claim that the so constructed set A is multiply permitting and r -maximal. Clearly, it suffices to show that A is multiply permitting (namely, every multiply permitting set is noncomputable; hence, \bar{A} is infinite, and every coinfinite c.e. superset of an r -maximal set is r -maximal as well). To this end, let e be given. By (2), fix $n \geq e$ such that $|G_n \cap \bar{B}| \geq (n + 1)^2$. We claim that any such n witnesses that (1) holds for φ_e in place of ψ with $f(x) = x$. Since by Lemma 1, there exist infinitely many such n , this proves the claim. By construction and by convention on converging computations, it suffices to show that $G_n \not\subseteq A_s$ holds for any stage s such that (b) holds. For a proof by contraposition, let s be a stage such that $G_n \subseteq A_s$. Now for any stage s' , a number may enter $A \cap G_n$ only if (b) holds at stage $s' + 1$ or if x enters B at stage $s' + 1$. But on the one hand, there are at most $|F_n| \cdot (n + 1) = (n + 1)^2$ numbers that may enter $A \cap G_n$ via (b). On the other hand, $G_n \cap \bar{B} \geq (n + 1)^2$ by choice of n . So $G_n \subseteq A_s$ can only hold if $\varphi_{e,s}(x) \downarrow$ holds for all $x \in F_n$; hence, (b) cannot hold for any stage $t \geq s$. □

Thus, it remains to show that Lemma 1 holds.

3.1 Proof of Lemma 1: Construction of B

For the proof of Lemma 1, we effectively construct a c.e. set B in stages s where B_s denotes the finite set of numbers which are enumerated into B by stage s . Before we give the formal construction, let us discuss some of the ideas behind it and introduce some of the concepts to be used in the construction.

We give the definition of the v.s.a.i. $\mathcal{G} = \{G_n\}_{n \geq 0}$ in advance. We define $\{G_n\}_{n \geq 0}$ as the unique v.s.a.i. such that $\min(G_0) = 0, \min(G_{n+1}) = \max(G_n) + 1$ and

$$|G_n| = 2^{\frac{n(n+1)}{2}}(n + 1)^2 \tag{3}$$

holds. Then it suffices to construct B such that (2) holds and such that B meets for all e the requirements

$$\mathcal{Q}_e : V_e^0 \cup V_e^1 = \omega \Rightarrow \exists i \leq 1 \forall^\infty n (V_e^i \cap G_n \subseteq B). \tag{4}$$

where $\{(V_e^0, V_e^1)\}_{e \geq 0}$ is an effective enumeration of all pairs of disjoint c.e. sets. Such an enumeration can be easily obtained as follows. Given $e = \langle e_0, e_1 \rangle$ and stage s , let $t_{e,s}$ be the largest stage $t \leq s$ such that $W_{e_0,t} \cap W_{e_1,t} = \emptyset$ and let $V_{e,s}^i = W_{e_i,t_{e,s}}$ for $i \leq 1$. Then $V_{e,s}^0 \cap V_{e,s}^1 = \emptyset$ for all $e, s \geq 0$ and if W_{e_0} and W_{e_1} are disjoint then $V_e^i = W_{e_i}$ for all $i \leq 1$. We call a requirement \mathcal{Q}_e *infinitary* if the hypothesis of \mathcal{Q}_e holds.

Clearly, it is undecidable whether a requirement is infinitary (in fact, it is not hard to show that this question is Π_2 -complete). So we have to effectively approximate this question in the course of the construction. For this we define T as the full binary tree as a priority tree. A node $\alpha \in T$ of length n codes a guess at which of the first n requirements are infinitary where, for $e < n, \alpha(e) = 0$ codes that \mathcal{Q}_e is infinitary. Correspondingly we call e an *infinitary edge* of α in this case. Then the *true path* TP is the infinite path through T satisfying $TP(e) = 0$ iff \mathcal{Q}_e is infinitary. In order to approximate TP at stage s of the construction we use the following *length of agreement function*

$$l(e, s) = \mu y (V_{e,s}^0(y) = V_{e,s}^1(y) = 0). \tag{5}$$

By choice of the sequence $\{(V_e^0, V_e^1)\}_{e \geq 0}$, $l(e, s)$ is nondecreasing in s for all e and, for fixed e , it is unbounded iff \mathcal{Q}_e is infinitary. Based on $l(e, s)$, we define the set of α -stages by induction on $|\alpha|$ as follows. Every stage is a λ -stage. An α -stage s is called *α -expansionary* if $s = 0$ or $l(|\alpha|, s) > l(|\alpha|, t)$ for all α -stages $t < s$. Then a stage s is an $\alpha 0$ -stage if it is α -expansionary and an $\alpha 1$ -stage if it is an α -stage but not α -expansionary. The current approximation δ_s of $TP \upharpoonright s$ at the end of stage s is the unique node α of length s such that s is an α -stage, and we say that α is *accessible* at stage $s + 1$ if α is an initial segment of δ_s , i.e., $\alpha \sqsubseteq \delta_s$. Note that $TP = \liminf_{s \rightarrow \infty} \delta_s$, i.e., $TP \upharpoonright n$ is the leftmost node of length n which is accessible infinitely often for every n . As usual, we say for two nodes α and β that α has *higher priority than* β and denote it by $\alpha < \beta$ iff $\alpha \sqsubset \beta$ (i.e., α is a proper initial segment of β) or α is to the left of β , denoted by $\alpha <_{left} \beta$, i.e., there exists $\gamma \in T$ such that $\gamma 0 \sqsubseteq \alpha$ and $\gamma 1 \sqsubseteq \beta$.

Now the strategy for meeting the requirements \mathcal{Q}_e which at the same time satisfies (2) is based on a variant of the e -state definition used in the construction of a maximal set as e.g. given in [9]. We assign the intervals G_n to the nodes $\alpha \in T$ where at each stage at most one interval is assigned to α . An unused interval is assigned to α only at a stage where α is accessible, and the interval assigned to α is cancelled if α is to the right of δ_s . In this case, G_n is *deleted*, i.e., all elements of G_n are enumerated into B . So an interval may be permanently assigned to α only if α is on the true path or to the left of it (we also make sure that intervals that are never assigned to any node are deleted as well). Moreover, for any number e , there will be only finitely many nodes to the left of $TP \upharpoonright e + 1$ which get a permanent interval assigned since only finitely many such nodes are ever accessible. So almost all intervals which are never deleted are assigned to nodes extending $TP \upharpoonright e + 1$ and hence have the correct guess about the type of the first $e + 1$ requirements.

Now, for any interval G_n , any node α and any stage s , the α -state of G_n at stage s , denoted by $\sigma(\alpha, n, s)$ is a binary string of length $\leq k$ where k is the number of infinitary edges of α where in the following, let $e_0 < e_1 < \dots < e_{k-1}$ be the infinitary edges of α . Then $|\sigma(\alpha, n, s)|$ is the greatest $j \leq k$ such that for any $j' < j$, $l(e_{j'}, s) > \max(G_n)$; hence, $V_{e_{j'},s}^0$ and $V_{e_{j'},s}^1$ partition G_n (note that for α on the true path $|\sigma(\alpha, n, s)| = k$ for sufficiently large s). Moreover, for $j' < j$, we choose the values $i_{j'}$ of $\sigma(\alpha, n, s)(j')$ inductively in such a way that enumerating $V_{e_{j'}}^{i_{j'}} \cap G_n$ into B will keep $|G_n \cap \bar{B}|$ at least as big as when we would enumerate $V_{e_{j'}}^{1-i_{j'}} \cap G_n$ into B (for the precise definition of the inductive step of $\sigma(\alpha, n, s)$, see (7) below). As we will show, this ensures that the inner clause of (2) holds for any n such that G_n is never deleted.

Finally, in order to guarantee that requirement \mathcal{Q}_e is met it suffices to ensure that (in the limit) almost all of the states $\sigma(\alpha, n, s)$ of intervals G_n permanently assigned to α extending $TP \upharpoonright e + 1$ agree on the first $k' + 1$ arguments where $k' + 1 = |\{e' \leq e : TP(e') = 0\}|$. Namely, for infinitary \mathcal{Q}_e this ensures that there is $i \leq 1$ such that $\sigma(\alpha, n, s)(k') = i$ in almost all of the cases above whence $V_e^i \cap G_n \subseteq B$ for almost all undeleted intervals G_n ; hence, $V_e^i \subseteq^* B$.

Now the states can be unified in the above way as follows. Whenever intervals G_n and $G_{n'}$ are assigned to α and β , respectively, where $\alpha < \beta$, $|\alpha| < |\beta|$ and $\sigma(\alpha, n', s) <_{left} \sigma(\alpha, n, s)$ holds then the interval $G_{n'}$ is assigned to α in place of G_n . Note that this replacement must be done even if α is to the left of δ_s . For the formal definition of the states we first introduce an auxiliary notion.

For finite subsets $E, F \subseteq \omega$ the *density of E inside F* , denoted by $\rho(E, F)$, is defined as

$$\rho(E, F) = \frac{|E \cap F|}{|F|}, \tag{6}$$

where we set $\rho(E, \emptyset) = 0$. Then given $\alpha \in T$, stage s and $n \geq 0$, we let $\sigma(\alpha, n, s)$ denote the α -state of G_n at stage s and define it to be the longest string $\sigma \in \{0, 1\}^{<\omega}$ such that $|\sigma| \leq |\{e < |\alpha| : \alpha(e) = 0\}|$ and for all $j < |\sigma|$ such that $\alpha(e_j) = 0$ (where e_j is the $(j + 1)$ th infinitary edge of α in order of magnitude), $l(e_j, s) > \max(G_n)$ holds and $\sigma(j)$ is the least $i \leq 1$ such that

$$\rho(V_{e_j,s}^i, \bar{B}_s \cap G_n \setminus \bigcup_{l < j} V_{e_l,s}^{\sigma(l)}) \leq \frac{1}{2}. \tag{7}$$

We let

$$V_{\sigma(\alpha,n,s)} = G_n \cap \bigcup_{j < |\sigma(\alpha,n,s)|} V_{e_j,s}^{\sigma(\alpha,n,s)(j)}. \tag{8}$$

Then the construction is as follows.

Construction of B .

Stage 0. $B_0 = \emptyset$.

Stage $s + 1$. Let B_s be given. We say that a node α *requires attention* at stage $s + 1$ if $|\alpha| \leq s$ and either

- (i) $\alpha \sqsubseteq \delta_s$ and no interval is assigned to α , or
- (ii) $\alpha \leq \delta_s$, G_n is assigned to α and $\sigma(\alpha, n, s) \sqsupset \sigma(\alpha, n, s - 1)$, or
- (iii) $\alpha \leq \delta_s$, G_n is assigned to α , (ii) does not hold and there exists $\beta > \alpha$ and n' such that $|\beta| > |\alpha|$, $G_{n'}$ is assigned to β and $\sigma(\alpha, n', s) <_{left} \sigma(\alpha, n, s)$.

Let α be the node of highest priority which requires attention at stage $s + 1$. Say that α *receives attention* and *acts via* the clause via which α requires attention.

If (i) holds, assign G_s to α at stage $s + 1$.

If (ii) holds, enumerate all of $V_{\sigma(\alpha,n,s)}$ into B at stage $s + 1$.

If (iii) holds, let β be the highest priority node which makes (iii) true and let $G_{n'}$ be its assigned interval. Cancel the assignment of G_n to α , assign $G_{n'}$ to α and enumerate all of $V_{\sigma(\alpha,n',s)}$ into B at stage $s + 1$.

At stage $s + 1$, *initialize* all nodes $\beta > \alpha$, i.e., cancel their assigned interval (if any). After α has received attention and the assignment of intervals to nodes has been declared at stage $s + 1$, for all $n \leq s$, do the following: if G_n is not assigned to any node at stage $s + 1$, *delete* G_n at stage $s + 1$, i.e., enumerate all of G_n into B at stage $s + 1$.

This ends the formal construction.

4 Verification

We prove in a series of claims that the so constructed set B has the required properties. Before, let us give some general remarks about the construction which we will tacitly use in the proofs below. Unless otherwise stated, they can be easily shown by induction on the stage s .

The construction is effective and $\{B_s\}_{s \geq 0}$ is a computable enumeration of B ; hence, B is a c.e. set. At any stage s , there is a unique node $\alpha \leq \delta_s$ which requires attention at stage $s + 1$. For all nodes α and stages s , α is assigned at most one interval at stage s , if G_n is the interval that is assigned to α at stage s then $|\alpha| \leq n < s$ and if α gets G_n assigned via (i) at stage $s + 1$ then $n = s$ and $G_n \cap B_s = \emptyset$ since all intervals that are assigned to nodes by stage s have index less than s .

Moreover, the assignment of intervals to nodes is nondecreasing in s and strictly increasing with respect to the priority ordering, i.e., if $\alpha < \beta$, α is assigned G_n and β is assigned $G_{n'}$ at stage s then $n < n'$. Furthermore, $\sigma(\alpha, n, s) \sqsubseteq \sigma(\beta, n, s)$ holds whenever $\alpha \sqsubseteq \beta$ holds and if G_n is assigned to α at stage $s + 1$ then

$$\sigma(\alpha, n, s) \sqsubseteq \sigma(\alpha, n, s + 1). \tag{9}$$

Hence, $V_{\sigma(\alpha, n, s)} \subseteq V_{\sigma(\alpha, n, s+1)}$ in this case.

Finally, and importantly, for any α and $n, s \geq 0$, if α gets G_n assigned at stage $s + 1$ then G_n has not been deleted at any stage $t \leq s$. In particular, if G_n is never deleted then from stage $n + 1$ on, it is always assigned to a node and it is eventually permanently assigned to a node which is on or to the left of the true path. Now the first claim states that nodes on or to the left of the true path act only finitely often and that the former eventually get a permanent interval assigned. Fix e in the following.

Claim 1. Every $\alpha \leq TP \upharpoonright e$ requires attention only finitely often and if $\alpha = TP \upharpoonright e$ then α eventually gets a permanent interval assigned.

Clearly, nodes to the left of the true path may act via (ii) or (iii) after they have been accessible. However, since this action is finitary and since nodes to the left of TP are accessible only finitely often, Claim 1 can be shown for given $\alpha \leq TP \upharpoonright e$ by induction on the number of nodes that have higher priority than α and which are ever accessible on the tree.

For the next claim, we show that \bar{B} is infinite by proving that clause (2) holds for all intervals G_n that are never deleted during the course of the construction. Note that there are infinitely many such intervals. Namely, by Claim 1, any node $\alpha \sqsubset TP$ gets a permanent interval assigned which by construction is never deleted and different from intervals which are permanently assigned to proper initial segments of α .

Claim 2. For all n , if G_n is never deleted then $|G_n \cap \bar{B}| \geq (n + 1)^2$.

Proof. Let $n \geq 0$ be given such that G_n is never deleted. By construction, for any stage $s \geq n$, there is a node γ such that G_n is assigned to γ at stage $s + 1$. Moreover, if G_n is assigned to γ at stage s and to γ' at stage $s + 1$ then γ' requires attention via (iii) which implies that $\gamma' < \gamma$ and $|\gamma'| < |\gamma|$. So we can argue that there exists $k \geq 0$ and a sequence of stages $s_0 = n < s_1 < \dots < s_k$ and nodes $\gamma_0 > \gamma_1 > \dots > \gamma_k$ such that $G_n \cap B_{s_0} = \emptyset$, G_n is assigned to γ_k at any stage $s > s_k$ and, moreover, for any $i < k$, G_n is assigned to γ_i at stage $s + 1$ for any $s \in [s_i, s_{i+1})$ and

$$|\gamma_i| > |\gamma_{i+1}|, \tag{10}$$

$$G_n \cap B_{s_{i+1}} \subseteq (G_n \cap B_{s_i}) \cup V_{\sigma(\gamma_i, n, s_{i+1})} \tag{11}$$

hold. Note that $k \leq n$ since $k \leq |\gamma_0|$ by (10) and $|\gamma_0| \leq n$ since G_n is assigned to γ_0 at stage $s_0 + 1$. In particular, $|\gamma_i| \leq n - i$ by (10) for all $i \leq k$. Furthermore, γ_k must be on or to the left of the true path. Otherwise, γ_k would be initialized after stage s_k and G_n would be deleted. So by Claim 1, fix the least stage $s_{k+1} > s_k$ such that γ_k does not require attention after stage s_{k+1} . Then $G_n \cap B = G_n \cap B_{s_{k+1}}$ and $G_n \cap B_{s_{k+1}} \subseteq (G_n \cap B_{s_k}) \cup V_{\sigma(\beta, n, s_{k+1})}$. So since for all nodes α , all $n, s \geq 0$,

$$\rho(V_{\sigma(\alpha, n, s)}, G_n \cap \bar{B}_s) \leq 1 - \left(\frac{1}{2}\right)^{|\sigma(\alpha, n, s)|} \tag{12}$$

holds, by (9) and (11), we may easily deduce that

$$|G_n \cap \bar{B}_{s_{i+1}}| \geq 2^{-|\gamma_i|} |G_n \cap \bar{B}_{s_i}|$$

holds for all $i \leq k$. However, since $G_n \cap B_{s_0} = \emptyset$, this yields

$$|G_n \cap \bar{B}| \geq 2^{-(|\gamma_k| + |\gamma_{k-1}| + \dots + |\gamma_0|)} |G_n| \geq (n + 1)^2.$$

This completes the proof. □

Finally, we show that all \mathcal{Q} -requirements are met.

Claim 3. \mathcal{Q}_e is met.

Proof. If the hypothesis of \mathcal{Q}_e does not hold, \mathcal{Q}_e is trivially met. So we may assume that \mathcal{Q}_e is infinitary, i.e., $TP(e) = 0$ holds. We have to show that there exists $i \leq 1$ and $m \geq 0$ such that for all $n \geq m$, $V_e^i \cap G_n \subseteq B$. For the proof of this statement, suppose that $e = e_k$ where e_k denotes the $(k + 1)$ th infinitary edge of TP in order of magnitude and let M be the set of all nodes $\beta \sqsupseteq TP \upharpoonright e + 1$ which eventually get a permanent interval assigned. For $\beta \in M$, let n_β denote the index of the last interval assigned to β and let $\sigma(\beta, n_\beta) = \lim_{s \rightarrow \infty} \sigma(\beta, n_\beta, s)$. Note that $\sigma(TP \upharpoonright e + 1, n_\beta, s) \sqsubseteq \sigma(\beta, n_\beta, s)$; hence, $\sigma(TP \upharpoonright e + 1, n_\beta) = \lim_{s \rightarrow \infty} \sigma(TP \upharpoonright e + 1, n_\beta, s)$ exists, too and $\sigma(TP \upharpoonright e + 1, n_\beta) \sqsubseteq \sigma(\beta, n_\beta)$. Moreover, $|\sigma(TP \upharpoonright e + 1, n_\beta)| = k + 1$ since $TP \upharpoonright e + 1$ lies on the true path; hence, $|\sigma(\beta, n_\beta)| \geq k + 1$ for any $\beta \in M$. Now let τ be the leftmost binary string of length $k + 1$ such that $\tau \sqsubseteq \sigma(\beta, n_\beta)$ for infinitely many $\beta \in M$. We claim that

$$\forall^\infty \beta \in M (\tau \sqsubseteq \sigma(\beta, n_\beta)) \tag{13}$$

holds and that (13) suffices to prove Claim 3. First, assume that (13) holds. Let $\beta_0 \in M$ be such that (13) holds for all $\beta \in M$ with $\beta \geq \beta_0$. Let $m = n_{\beta_0}$ and $i = \tau(k)$. We claim that for all $n \geq m$, $V_e^i \cap G_n \subseteq B$ holds. Fix $n \geq m$. We distinguish between the following two cases. If G_n is eventually deleted then $G_n \subseteq B$. So the claim trivially holds for n . Otherwise, G_n is never deleted. Then by construction, there is a unique $\beta \in M$ such that $n = n_\beta$. Namely, β must be on or to the left of the true path. But by choice of m , β cannot be to the left of $TP \upharpoonright e + 1$ nor can $\beta < \beta_0$ hold because by construction, $n_\beta < m$ holds in

both cases. Thus, by (13), $\sigma(\beta, n_\beta)(k) = i$. So by (9) and since $\beta \sqsupseteq TP \upharpoonright e + 1$, β eventually enumerates $V_{\sigma(\beta, n_\beta)} \supseteq V_e^i \cap G_n$ into B .

Thus, to complete the proof, we show that (13) holds. By definition of τ , let $\beta'_0 \in M$ be such that for all $\beta \in M$ with $|\beta| \geq |\beta'_0|$, $\tau \leq \sigma(TP \upharpoonright e + 1, n_\beta)$ holds. Fix any such node β and by Claim 1, fix s_0 such that β does not require attention after stage s_0 . In particular, $\sigma(\beta, n_\beta) = \sigma(\beta, n_\beta, s)$ for all $s \geq s_0$. We claim that $\sigma(\beta, n_\beta) \sqsupseteq \tau$. Otherwise, $\tau <_{left} \sigma(\beta, n_\beta)$ by choice of β . By choice of τ , let $\beta' \in M$ with $|\beta'| > |\beta|$ and $\beta' > \beta$ be such that $\sigma(\beta', n_{\beta'}) \sqsupseteq \tau$ and let $s_1 > s_0$ be a stage such that $\sigma(\beta', n_{\beta'}, s_1) = \sigma(\beta', n_{\beta'})$. Such a β' exists because by construction, there are only finitely many nodes in M which have higher priority than β . But then β requires attention via (iii) at stage $s_1 + 1$, contrary to choice of s_0 . \square

By Claim 3, this completes the proof of Lemma 1 and hence of Theorem 1.

Acknowledgements. We would like to thank Klaus Ambos-Spies for suggesting this problem and for numerous helpful discussions on the topic of this paper.

References

1. Ambos-Spies, K.: Multiple permitting and array noncomputability. In: Manea, F., et al. (eds.) CiE 2018. LNCS, vol. 10936, pp. 30–39 (2018)
2. Ambos-Spies, K., Fang, N., Losert, N., Merkle, W., Monath, M.: Array noncomputability: a modular approach. Unpublished Notes
3. Ambos-Spies, K., Ding, D., Fan, Y., Merkle, W.: Maximal pairs of computably enumerable sets in the computably Lipschitz degrees. *Theory Comput. Syst.* **52**, 2–27 (2013)
4. Barpalias, G., Downey, R., Greenberg, N.: Working with strong reducibilities above totally ω -c.e. and array computable degrees. *Trans. Am. Math. Soc.* **362**, 777–813 (2010)
5. Downey, R., Hirschfeldt, D.: *Algorithmic Randomness and Complexity. Theory and Applications of Computability.* Springer, New York (2010). <https://doi.org/10.1007/978-0-387-68441-3>
6. Downey, R., Jockusch, C., Stob, M.: Array nonrecursive sets and multiple permitting arguments. In: Ambos-Spies, K., Müller, G.H., Sacks, G.E. (eds.) *Recursion Theory Week.* LNM, vol. 1432, pp. 141–173. Springer, Heidelberg (1990). <https://doi.org/10.1007/BFb0086116>
7. Myhill, J.: The lattice of recursively enumerable sets. *J. Symb. Log.* **21**, 220 (1966)
8. Post, E.: Recursively enumerable sets of positive integers and their decision problems. *Bull. Am. Math. Soc.* **50**, 284–316 (1944)
9. Soare, R.: *Recursively Enumerable Sets and Degrees. Perspectives in Mathematical Logic.* Springer, Chicago (1987)



The Complexity of Tukey Types and Cofinal Types

Marie Nicholson^(✉)

Cork Institute of Technology, Cork, Ireland
marie.nicholson@cit.ie

Abstract. This paper studies how difficult it is to determine whether two computable partial orders share the same Tukey type and the same cofinal type. For Tukey types, we show the index set is $\mathbf{0}^{(3)}$. For cofinal types, we show the index set is computable from $\mathbf{0}^{(4)}$. This is in sharp contrast to the isomorphism problem for computable partial orders, which is Σ_1^1 .

1 Introduction

The isomorphism relation on countable posets preserves all the structural information but there are 2^{\aleph_0} many isomorphism classes of countable posets and the isomorphism relation is Σ_1^1 -complete. One way of dealing with the complexity of classifying posets up to isomorphism is to look at coarser equivalence relations that preserve some but not all of the structural information. Two examples of this for posets are Tukey equivalence and cofinal equivalence. These equivalences are defined in terms of reducibility notions. In both cases, the equivalence relations give rise to just countably many equivalence classes and their equivalence types are defined arithmetically.

Both Tukey and cofinal types have been studied on more restrictive classes than posets, such as directed posets, and on more general classes, such as oriented systems. Tukey introduced the Tukey ordering to develop the notion of Moore-Smith convergence in topology [9]. After its initial success in helping develop general topology, Tukey reducibility was studied as a means of classifying algebraic structures related to partially ordered sets in, for example, Day [1], Isbell [5] and Todorčević [8]. Cofinal equivalence is closely related to Tukey equivalence of posets and although it is finer than Tukey equivalence, in the context of directed posets the two notions are the same. Here we examine the complexity of Tukey types and cofinal types of countable posets.

In Sect. 2, we introduce Tukey reducibility for partially ordered sets and the resulting equivalence classes which are called Tukey types. Tukey types are themselves partially ordered by the Tukey reduction \leq_{ty} and we outline the maximal and minimal Tukey types of computable partial orders. In Sect. 3, we outline the conditions under which we can decompose a partially ordered set into directed components and in Sect. 4, we show that the complexity of the index

set of Tukey types is $\mathbf{0}^{(3)}$. In Sect. 5, we show that cofinal type of a computable partial order can be uniformly computed by $\mathbf{0}^{(4)}$.

Our notation is standard and follows Soare [7].

2 Tukey Types of Computable Partial Orders

We begin by introducing Tukey reducibility for partially ordered sets.

Definition 1. We say that $f : D \rightarrow E$ is a convergent map from D into E if for each $e \in E$ there is a $d \in D$ such that $f(c) \geq_E e$ for all $c \geq_D d$.

Definition 2. We say a partial ordering (E, \leq_E) is Tukey reducible to a partial ordering (D, \leq_D) , written as $E \leq_{ty} D$, if and only if there is a convergent map from D into E .

There are various classically equivalent definitions for the notion of Tukey reducibility.

Definition 3. We say a function $f : D \rightarrow E$ is cofinal if the image of each cofinal subset of D is cofinal in E .

Definition 4. A subset $X \subseteq D$ is called unbounded if there is no single $d \in D$ which simultaneously bounds every member of X . That is, for each $d \in D$, there is some $x \in X$ such that $d \not\leq_D x$. A map $g : E \rightarrow D$ is called a Tukey map or an unbounded map if the g -image of each unbounded subset of E is an unbounded subset of D .

The following useful facts are proved in Tukey [9] and in Schmidt [6].

Lemma 1 (Tukey [9], Schmidt [6]). For partially ordered sets D and E , the following are equivalent:

1. $E \leq_{ty} D$.
2. There is a cofinal map from D into E .
3. There is an unbounded map from E into D .

Definition 5. If $P \leq_{ty} Q$ and $Q \leq_{ty} P$ we say that P is Tukey equivalent to Q , written as $P \equiv_{ty} Q$.

The relation \equiv_{ty} is an equivalence relation and the equivalence classes are called *Tukey types*. Tukey types are themselves partially ordered by the Tukey reduction \leq_{ty} . We now consider the Tukey types of directed posets.

Definition 6. We say that a poset P is directed if for all $a, b \in P$ there exists $c \in P$ such that $a \leq_P c$ and $b \leq_P c$.

In what follows, we let $1 = \{e\}$ denote the directed set with just one element and we let ω denote \mathbb{N} under the usual ordering. Maximal and greatest elements will be important in this context.

Definition 7. Let (P, \leq_P) be a poset. We say that $a \in P$ is a maximal element if for all $p \in P$ such that $p \geq_P a$ we have $p = a$.

Definition 8. Let (P, \leq_P) be a poset. We say that $a \in P$ is a greatest element if for all $p \in P$ we have $p \leq_P a$.

Note that in directed posets a maximal element is also a greatest element.

Tukey showed in [9] that the Tukey types of countable directed partial orders are 1 and ω . First we consider directed sets with a greatest element.

Lemma 2 (Tukey [9]). *The Tukey type of a directed poset with a greatest element is 1.*

Any pair of functions between these partially ordered sets, which send the greatest element of one set to the greatest element of the other set, witnesses this equivalence.

In particular, any finite directed set is Tukey equivalent to 1. Also notice that two partial orders do not need to have the same cardinality to be Tukey equivalent. Next we consider directed sets without greatest elements.

Lemma 3 (Tukey [9]). *The Tukey type of a countable directed set without a greatest element is ω .*

This equivalence is witnessed by functions which send cofinal ω -chains in one set to cofinal ω -chains in the other set. More generally, Lemma 3 holds for any directed set whose cofinality is \aleph_0 . We also note that 1 is strictly less than ω in the Tukey ordering.

Hence the Tukey type 1 is characterised by the directed posets with a greatest element and the Tukey type ω contains all countable directed sets without a greatest element. Alternatively we may think of the posets of type ω as those which have a cofinal ω -chain.

Remark 1. So, given a computable directed poset, if we can determine whether or not it has a greatest element, then we know its Tukey type. This has Σ_2^0 complexity.

We now consider an arbitrary computable poset. Given a poset (P, \leq_P) , the function $f : P \rightarrow 1$ defined by $f(p) = e$ is cofinal and so $1 \leq_{t_y} P$ for all posets. As before, unless (P, \leq_P) has a greatest element, 1 is strictly less than P in the Tukey ordering.

We now ask if the partial ordering of Tukey types of countable posets have a greatest element.

Definition 9. Let (P, \leq_P) be a poset. We say that the elements a and b are incompatible in P , if there is no $c \in P$ such that $a <_P c$ and $b <_P c$.

Definition 10. A strong antichain in a poset (P, \leq_P) is a subset A of P in which each pair of distinct elements are incompatible in P .

The existence of a strong antichain characterises the maximal element in the partial ordering of countable posets.

Lemma 4 (Day [1]). *If (P, \leq_P) and (Q, \leq_Q) are countable posets and P has an infinite strong antichain, then $Q \leq_{ty} P$.*

Also, if P has an infinite strong antichain and $P \leq_{ty} Q$, then Q also has an infinite strong antichain.

Lemma 5 (Day [1]). *Suppose P is a countable poset with an infinite strong antichain and let $g : P \rightarrow Q$ be an unbounded map. Then Q also has an infinite strong antichain.*

Hence a poset (P, \leq_P) is in the maximal Tukey type of countable partial orders if and only if (P, \leq_P) has an infinite strong antichain. It was shown by Frittation and Marcone in [4] that the existence of an infinite strong antichain in a partial order is equivalent to the existence of arbitrarily large finite strong antichain. So (P, \leq_P) has an infinite strong antichain if and only if for all $n \in \mathbb{N}$ there exists a strong antichain of size n . That is,

$$(\forall n \in \mathbb{N})(\exists a_1, a_2, \dots, a_n \in P)(\forall i \neq j \leq n)(\forall p \in P)(a_i \not\leq_P p \text{ or } a_j \not\leq_P p).$$

This is a Π_3^0 statement. In fact, the index set of computable posets with an infinite strong antichain is Π_3^0 -complete.

Lemma 6. *There is a computable sequence of posets $\langle P_n \mid n \in \mathbb{N} \rangle$ such that P_n has an infinite strong antichain if and only if W_n is co-infinite.*

Proof. We construct the poset (P_n, \leq_{P_n}) in stages. At each stage we enumerate two elements, a_s and x_s , into P_n . We let $x_i \leq_{P_n} x_s$ for each $i \leq s$. In addition, if $i \leq s$ is enumerated into W_n by stage s , we let $a_i \leq_{P_n} x_s$. The a_i 's will form a antichain and the x_i 's will form an infinite chain as follows:

$$x_0 \leq_{P_n} x_1 \leq_{P_n} x_2 \leq_{P_n} \dots x_i \leq_{P_n} x_{i+1} \leq_{P_n} \dots$$

We claim P_n has an infinite strong antichain if and only if W_n is co-infinite. The poset P_n has an infinite strong antichain if and only if there are infinitely many a_i such that $a_i \not\leq_{P_n} x_s$ for any s . This happens if and only if there are infinitely many i such that $i \notin W_n$. That is, W_n is co-infinite.

We let ∞ denote the Tukey type which consists of all posets with an infinite strong antichain. Hence given a countable poset (P, \leq_P) we have $1 \leq_{ty} P \leq_{ty} \infty$.

3 Decomposing Partial Orders into Directed Components

We now seek to compute the Tukey type of posets which are not directed and do not have an infinite strong antichain. As a first step, we decompose posets into directed components, where possible. It is always possible to decompose a

poset into directed subsets by decomposing it into single points. However, the unrelated disjoint union of two ω -chains clearly has two directed parts. So we would like to partition a set into as few directed subsets as possible. We may think of this as decomposing a poset into paths and stars. In [2, 3], Diestel and Pikhurko captured this notion in a *essential directed* subset; a directed subset which every cofinal subset must intersect with. Following their notation, we begin with some definitions.

Definition 11. *If $A \subseteq P$ is not cofinal in P , we say A is small in P . If A is small in P , then we say that the complement of A in P is essential.*

Hence any cofinal subset of P intersects all of the essential subsets of P . The canonical example of an essential subset is the up-closure of a single point x in P , that is $\lfloor x \rfloor_P$. In fact, it is straightforward to check that a subset of P is essential if and only if it contains the up closure of at least one point in P . In [2], Diestel showed that any poset without an infinite antichain may be decomposed into finitely many essential directed sets and the decomposition is unique up to Tukey equivalence. We will strengthen this result by showing that a partially ordered set may be decomposed into finitely many essential directed sets if and only if it does not contain a strong antichain.

Lemma 7. *If (P, \leq_P) is a countable partially ordered set, then P can be partitioned into finitely many essential directed subsets if and only if P does not contain an infinite strong antichain.*

Proof. Suppose P contains an infinite strong antichain. By definition, a directed subset of P may contain at most one element of the strong antichain. Hence P cannot be partitioned into finitely many directed sets.

Conversely, suppose P cannot be partitioned into finitely many directed sets. We will show by induction on $n \geq 2$ that P has a strong antichain of size n . By Frittaion and Marcone [4], this suffices to show that P has an infinite strong antichain. For the base case, we show that P has a strong antichain of size 2. Take $a_0 \in P$. Then $\lfloor a_0 \rfloor_P$ is an essential set. Either $\lfloor a_0 \rfloor_P$ is directed or not. First consider the case when $\lfloor a_0 \rfloor_P$ is directed. Then there must be $a_1 \in P$ such that $a_1 \not\leq_P a$ for any $a \in \lfloor a_0 \rfloor_P$. If such an a_1 does not exist, then P is a directed set and hence may be partitioned into one directed essential set. Hence $\{a_0, a_1\}$ forms a strong antichain of size two. For the second case, suppose that $\lfloor a_0 \rfloor_P$ is not directed. Then there exists $a_1 >_P a_0$ and $a_2 >_P a_0$ such that a_1 and a_2 are incompatible. So $\lfloor a_1 \rfloor_P$ and $\lfloor a_2 \rfloor_P$ are disjoint essential sets. Now $\{a_1, a_2\}$ forms an strong antichain of size two.

For the induction case, suppose we have a strong antichain a_0, a_1, \dots, a_{n-1} of size n . We show that there exists a strong antichain of size $n + 1$. As a_0, a_1, \dots, a_{n-1} forms a strong antichain $\lfloor a_0 \rfloor_P, \lfloor a_1 \rfloor_P, \lfloor a_2 \rfloor_P, \dots, \lfloor a_{n-1} \rfloor_P$ are disjoint essential sets. If $\lfloor a_0 \rfloor_P, \lfloor a_1 \rfloor_P, \lfloor a_2 \rfloor_P, \dots, \lfloor a_{n-1} \rfloor_P$ are also directed sets, then there exists a_n such that $a_n \not\leq_P a$ for any a in $\lfloor a_i \rfloor_P$ where $0 \leq i < n$. If this were not the case, it would be possible to partition P into n many essential directed sets. Hence $\{a_0, a_1, \dots, a_{n-1}, a_n\}$ is a strong antichain of size

$n + 1$. Otherwise, suppose that at least one of $[a_0]_P, [a_1]_P, [a_2]_P, \dots, [a_{n-1}]_P$ is not directed. Take the least such a_i . Since $[a_i]_P$ is not directed we have $a_n >_P a_i$ and $a_{n+1} >_P a_i$ such that a_n and a_{n+1} are not compatible in P . Hence $\{a_0, a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n, a_{n+1}\}$ is a strong antichain of size $n + 1$.

As there exists a strong antichain of size n for every $n \in \mathbb{N}$, there exists an infinite strong antichain.

Remark 2. To construct an infinite strong antichain in Lemma 1 we need to determine when the up-closure of a point is directed, which is a Π_2^0 statement. We also need to determine if a point x is below any point in the up-closure of another point $[a]_P$. This is a Σ_1^0 statement.

Lemma 8 (Diestel [2]). *If $P = \bigcup_{0 \leq i \leq n} A_i$ is a partition of P into essential directed subsets, then every essential directed subset of P is Tukey equivalent to one of the A_i .*

Furthermore, the number of essential directed subsets n in the partition is unique.

Lemma 9 (Diestel [2]). *Let $P = \bigcup_{0 \leq i \leq n} A_i$ and $B = \bigcup_{0 \leq i \leq m} B_i$ be two partitions of P into essential directed subsets. Then $m = n$.*

Therefore there is a most one partition of a partially ordered set into directed essential sets, up to Tukey equivalence.

4 Complexity of Tukey Types

Recall that the Tukey type of a directed set is 1 if that set has a greatest element and the Tukey type of a directed set is ω otherwise. Given a partition of a poset P into essential directed sets, one would hope that the Tukey type of the essential directed subsets would determine the Tukey type of the poset. To this end, we say that a poset P has type (n, m) when P partitions into $n + m$ many directed sets and n of those sets have greatest elements where $n, m \in \mathbb{N}$. Day showed that two posets are Tukey equivalent if and only if they have the same type.

Lemma 10 (Day [1]). *If P has type (n_1, m_1) and Q has type (n_2, m_2) , then $P \leq_{ty} Q$ if and only if $n_1 + m_1 \leq n_2 + m_2$ and $m_1 \leq m_2$.*

So, once we have partitioned a poset into essential directed sets, we can determine the Tukey type of the poset by checking whether or not each essential directed set in the partition has a greatest element. Hence we have the following theorem, initially proved by Day [1] in the context of oriented systems.

Theorem 1 (Day [1]). *If (P, \leq_P) is a computable poset, then the Tukey type is that of an infinite strong antichain, ∞ , or (n, m) where P decomposes into $n + m$ many essential directed sets and n of them have a greatest element.*

For $i \in \mathbb{N}$, let \leq_i be the binary relation computable by the i th partial computable function φ_i on \mathbb{N} . We let $P_i = (\mathbb{N}, \leq_i)$ and note that saying that P_i is a partial order is a Π_2^0 statement since it requires saying φ_i is total. We denote the index set of Tukey types as

$$I_{ty} = \{\langle e, i \rangle \mid P_e \text{ and } P_i \text{ are partial orders and } P_e \equiv_{ty} P_i\}.$$

Lemma 11. *I_{ty} is Turing computable from $\mathbf{0}^{(3)}$.*

Proof. We have $\langle e, i \rangle \in I_{ty}$ if and only if φ_e and φ_i define partial orders on \mathbb{N} (which are Π_2^0 conditions) plus one of the following holds:

1. both P_e and P_i have infinite strong antichains (which are Π_3^0 conditions), or
2. there are n and m such that each of P_e and P_i decompose into exactly $n + m$ many essential directed sets and n of them have maximal elements.

Condition 2 is Σ_3^0 : there exist $m, n \in \mathbb{N}$ such that

- for all x_1, \dots, x_{n+m+1} , there are $u, v \leq n + m + 1$ with $u \neq v$, such that x_u and x_v are compatible (this is a Π_2^0 condition), and
- there exist elements x_1, \dots, x_{n+m} such that
 - for all $u \neq v \leq n + m$, x_u and x_v are incompatible (Π_1^0 condition), and
 - for all $u \leq n$, there is a maximal element above x_u (Σ_2^0 condition), and
 - for all $n < v \leq n + m$, there is not a maximal element above x_v (Π_2^0 condition).

So condition 2 has the form

$$\exists(\Pi_2^0 \wedge \Sigma_3^0)$$

and hence is Σ_3^0 . Altogether, the statement defining I_{ty} has the form

$$\Pi_2^0 \wedge (\Pi_3^0 \vee \Sigma_3^0).$$

As $\mathbf{0}^{(3)}$ can answer Π_3^0 and Σ_3^0 questions, we have $I_{ty} \leq_T \mathbf{0}^{(3)}$.

We also have the following corollary of Lemma 6.

Lemma 12. *Let Q be a fixed computable poset consisting of an infinite antichain. There is a computable sequence of posets $\langle P_n \mid n \in \mathbb{N} \rangle$ such that $\{n \mid P_n \equiv_{ty} Q\} \equiv_T \mathbf{0}^{(3)}$.*

Proof. Let $\langle P_n \mid n \in \mathbb{N} \rangle$ be the sequence constructed in Lemma 6. By construction, the set $\{n \mid P_n \text{ has an infinite strong antichain}\} \equiv_T \mathbf{0}^{(3)}$. The current lemma now follows from the fact that $P_n \equiv_{ty} Q$ if and only if P_n has a strong infinite antichain.

Theorem 2. *The complexity of the index set of Tukey types I_{ty} is $\mathbf{0}^{(3)}$*

Proof. This follows from the previous lemmas.

In comparison the isomorphism problem,

$$\{(a, b) \mid P_a, P_b \text{ are computable posets and } P_a \cong P_b\},$$

is Σ_1^1 -complete. Clearly the isomorphism type of a partial order gives us more information than the Tukey type. However if the Tukey type of a partial order is (n, m) , where $n, m \in \mathbb{N}$, then we have a very good idea what the partial order looks like. However if the Tukey type is ∞ then, for example, it may contain a binary tree, \aleph_0 many ω -chains, or an infinite antichain. We turn to cofinal similarity to give us a better picture in this case.

5 Cofinal Similarity of Computable Partial Orders

Definition 12. *Two partially ordered sets P, Q are cofinally similar if there exists a third partial order into which they both embed cofinally. That is, there is a partially ordered set R with cofinal sets $C_P, C_Q \subseteq R$ such that P is order isomorphic to C_P and Q is order isomorphic to C_Q .*

Cofinal similarity turns out to be an equivalence relation, and the equivalence classes are called *cofinal types*. If P is cofinally similar to Q , then we write $P \equiv_{cf} Q$. Day showed in [1] that for directed sets cofinal similarity coincides with Tukey equivalence. He also showed that cofinal similarity and Tukey equivalence coincide when a partially ordered set can be partitioned into finitely many essential directed sets. However, not all posets with an infinite strong antichain are cofinally similar. For example, an infinite strong antichain cannot preserve order and embed cofinally into a poset which has a cofinal binary tree or cofinal set which is isomorphic to \aleph_0 many ω -chains. Similarly a partial order with \aleph_0 many ω -chains cannot embed cofinally into a partial order with a cofinal binary tree.

In what follows, let (P, \leq_P) be a partially ordered set.

Definition 13. *We say that B is an essential cofinal subset of P if B is cofinal in some essential subset C of P .*

Everywhere branching trees are important in this context.

Definition 14. *Let P be a poset. We call $A \subseteq P$ an everywhere branching tree if for all $a \in A$ there exists incompatible elements $a_1, a_2 \geq_P a$.*

Following Day’s notation in [1] we make the following definitions.

Definition 15. *Let $P_1 = \{p \in P \mid [p] \text{ is directed}\}$ and $P_2 = \{q \in P \mid \forall p \geq_P q (p \notin P_1)\}$.*

So we have determined two disjoint subsets of P . The first part P_1 consists of directed subsets of P and the second part P_2 consists of a maximal everywhere branching tree in P . Note that if P is computable, then P_1 is Π_2^0 and P_2 is Π_3^0 .

As already mentioned, every poset with no infinite antichain can be partitioned into finitely many essential directed sets. In addition to this, Diestel and Pikhurko showed in [3] that a countable partially ordered set (P, \leq_P) does not have an essential cofinal subset isomorphic to an ever-branching tree if and only if it admits a partition into essential directed sets.

As infinitely many copies of $2^{<\omega}$ or $\omega^{<\omega}$ can embed into a single copy, we need only consider the existence or absence of an everywhere branching tree. Again, following Day’s notation, we make the following definition.

Definition 16. *Let \mathcal{E}_0 denote the partial order that consists of countably many disjoint copies of $\omega^{<\omega}$.*

Day proved in [1] Lemma 7.2 that a countable poset P is ever-branching (meaning that $P_2 = P$) if and only if P has a cofinal subset isomorphic to \mathcal{E}_0 . Also in Theorem 4.8 [1], Day gives the following properties, which outline how a countable poset P can be decomposed to understand its cofinal parts.

- P_1 and P_2 are disjoint and at least one is non-empty.
- P_1 and P_2 are each upward closed.
- $P_1 + P_2$ is cofinal in P .
- If P_2 is non-empty, then (P_2, \leq_P) is an everywhere branching poset and hence has a cofinal subset isomorphic to \mathcal{E}_0 by Lemma 7.2 in Day.
- If P_1 is non-empty, then (P_1, \leq_P) can be partitioned into essential directed sets, each of which either has a maximal element or has a cofinal ω -chain. However, unlike the Tukey type case there may be infinitely many of each of these types of essential directed subsets.

So, to determine the cofinal type of a countable partially ordered set we need to determine if it contains any essential cofinal subsets isomorphic to a everywhere branching tree and then partition the remainder of the poset into essential directed sets. Hence, with each countable poset P we associate a tuple (n_P, m_P, l_P) such that n_P , where $0 \leq n_P \leq \aleph_0$, is the number of essential directed subsets with a greatest element in a partition of P , and m_P , where $0 \leq m_P \leq \aleph_0$, is the number of essential directed subsets without a greatest element in a partition of P , and l_P , where $l_P \in \{0, 1\}$ denotes the existence of a essential cofinal subset isomorphic to an ever-branching tree. If $l_P = 1$, then P contains an essential cofinal subset isomorphic to an ever-branching tree and if $l_P = 0$ it does not. Day found a classification of countable oriented systems under cofinal similarity in terms of these three elements.

Lemma 13 (Day [1, Corollary 7.4]). *Two countable partially ordered sets are cofinally similar if and only if their associated triples are the same.*

Now that we know the tuple (n_P, m_P, l_P) associated with a computable partial order P determines the cofinal type, we ask how complicated it is to find the tuple (n_P, m_P, l_P) . To do this we need to determine if a computable poset P contains an essential cofinal subset which is isomorphic to an everywhere branching tree. First, we make the following observation.

Theorem 3. *The complexity of finding the cofinal type of a computable poset is at most $\Sigma_4^0 \wedge \Pi_4^0$.*

Proof. Given a computable poset P , to compute the tuple (n_P, m_P, l_P) we need to find out if P contains a cofinal essential everywhere branching tree and then decompose the remainder of P into essential directed posets.

Firstly, we need to determine if P contains a cofinal essential everywhere branching tree. This is equivalent to asking if P_2 is non-empty, which is Σ_4^0 question. This determines the third component of the cofinal type.

Secondly, we need to determine the number of directed components in P_1 and whether or not these directed components have a maximal element. Note that an element is maximal in (P_1, \leq_P) if and only if it is maximal in (P, \leq_P) . To be a maximal point in P is a Π_1^0 condition. Therefore, to ask if there are infinitely many is a Π_3^0 question and to ask if there are exactly n many maximal points is a $\Sigma_2^0 \wedge \Pi_2^0$ question (i.e. there are n many maximal points and there are not $n + 1$ many maximal points). The answers to these questions determine the first component of the cofinal type.

Finally, we need to determine how many maximal directed sets in P_1 have cofinal type ω . To ask if there are infinitely many, we need to ask whether for every n , there are n many distinct elements x_1, \dots, x_n in P_1 such that each pair $x_i \neq x_j$ is incompatible. Since being in P_1 is Π_2^0 and being incompatible is Π_1^0 , this question is Π_4^0 . To ask if there are exactly n many such points, we need to ask whether there are n many pairwise incompatible elements of P_1 (a Σ_3^0 condition) but not $n + 1$ such elements (a Π_3^0 condition).

It follows that the triple describing the cofinal type of a computable partial order can be uniformly computed by $\mathbf{0}^{(4)}$. We conjecture that this bound is sharp. That is, the complexity of the index set

$$I_{cf} = \{\langle e, i \rangle \mid P_e, P_i \text{ are posets and } P_e \equiv_{cf} P_i\}$$

is exactly $\mathbf{0}^{(4)}$. The proof of this conjecture is ongoing research.

References

1. Day, M.M.: Oriented systems. *Duke Math. J.* **11**, 201–229 (1944). MR 0009970
2. Diestel, R.: Relating subsets of a poset, and a partition theorem for WQOs. *Order* **18**(3), 275–279 (2001). MR 1867238
3. Diestel, R., Pikhurko, O.: On the cofinality of infinite partially ordered sets: factoring a poset into lean essential subsets. *Order* **20**(1), 53–66 (2003). MR 1993410
4. Frittaion, E., Marcone, A.: Reverse mathematics and initial intervals. *Ann. Pure Appl. Log.* **165**(3), 858–879 (2014). MR 3142390
5. Isbell, J.R.: The category of cofinal types. II. *Trans. Am. Math. Soc.* **116**, 394–416 (1965). MR 0201316
6. Schmidt, J.: Konfinalität. *Z. Math. Logik Grundlagen Math.* **1**, 271–303 (1955). MR 0076836 (17,951e)

7. Soare, R.I.: Recursively enumerable sets and degrees. *Bull. Am. Math. Soc.* **84**(6), 1149–1181 (1978)
8. Todorčević, S.: Directed sets and cofinal types. *Trans. Am. Math. Soc.* **290**(2), 711–723 (1985). MR 792822 (87a:03084)
9. Tukey, J.W.: *Convergence and Uniformity in Topology*. *Annals of Mathematics Studies*, vol. 2. Princeton University Press, Princeton (1940). MR 0002515 (2,67a)



Functionals of Type 3 as Realisers of Classical Theorems in Analysis

Dag Normann^(✉)

Department of Mathematics, The University of Oslo,
P.O. Box 1053, Blindern, 0316 Oslo, Norway
dnormann@math.uio.no

Abstract. We investigate the relative computational strength of combinations of four higher order functionals, the jump and hyperjump seen as functionals of type 2 and realisers for the compactness of Cantor space and the Lindelöf property of Baire space seen as functionals of type 3. We compare them with the closure operator for non-monotone inductive definitions of sets of integers, also seen as a functional of type 3.

1 Background

The current note is a spin-off from a joint project with Sam Sanders, based on his original initiative. In [17], Sanders used the Kohlenbach-inspired axiomatisation of *nonstandard analysis* from [1] to obtain results in computability theory from nonstandard analysis. His proof-theoretical analysis of the nonstandard proofs of Weak König's Lemma and the less known Weak Weak König's Lemma, combined with methods of *term extraction*, led to the discovery of two classes of type three functionals, the Θ -functionals (which will be discussed further in this note), and the Λ -functionals. For the latter, see also [13–15]. The original classes of Θ - and Λ -functionals were obtained using term extraction from proofs, but brief formulations equivalent from the point of view of computability theory are:

If we view a functional F of type two as representing an open covering of the Cantor space (see details in Sect. 2.2), then $\Theta(F)$ will be a finite subcovering, while $\Lambda(F)$ will be a countable subcovering of a set of measure 1.

Sanders got the author of this note interested in the computational strength of these objects, and a fruitful cooperation started. As a preliminary result, it was established that these objects are genuinely of type 3, no Θ - or Λ -functional is computable, in the sense of Kleene, in any object of type 2. Moreover, it became clear that there will be Θ - and Λ -functionals that are *countably based*. A functional Ψ of type 3 is countably based if the value of $\Psi(F)$ will be determined from the restriction of F to a countable set. The initial investigations of countably based functionals were described by Hartley [6–8]. At the time, there were few natural examples beyond the Superjump. The fan functional and the Gandy-Hyland functional are important objects of type three, but they are nowadays

mainly considered as functionals that are restricted to the class of continuous objects of type 2. Our project now has shown that many natural functionals of type 3 belong to the class of countably based functionals.

Next, it became clear that there are Θ - and Λ -functionals that are sequential in the sense that the value can be “computed” via a transfinite recursive procedure. It also became clear that these functionals are intimately related to some classical theorems in mathematics, such as the Heine-Borel theorem and the Vitali covering theorem. In particular, we observed that the sequential transfinite procedure used to construct one particular Θ -functional turned out to be an almost line-by-line translation from $[a, b]$ to the Cantor space of a construction due to Borel [2]. These connections are to some extent explored in [16], are brought further in [18] and will also be investigated in the continuation of our project.

We now see Θ -functionals as realisers for the compactness of the Cantor space, and we have introduced another class of functionals, the Ξ -functionals, that can be seen as realisers of the Lindelöf property of the Baire space $\mathbb{N}^{\mathbb{N}}$. The technical aim of this note is to show how the computational strengths of these objects are related, and also to link this to the complexity of (non-monotone) inductive definitions of subsets of \mathbb{N} . As a part of this, we will use the Borel-method to produce a realiser for the Lindelöf property, and indirectly, give a kind of Reverse Mathematics characterisation of the strength of the Borel method from 1895.

2 The Five Functionals

2.1 Two Functionals of Type 2

We will consider the functional μ , known as *Feferman’s μ* , and the Suslin functional S as follows:

$$\mu(f) := \begin{cases} 0 & \text{if } f \text{ is constant zero} \\ \text{the least } a & \text{such that } f(a) > 0 \text{ otherwise.} \end{cases}$$

$$S(f) := \begin{cases} 0 & \text{if } \forall g \exists n (f(\bar{g}(n)) = 0) \\ 1 & \text{if } \exists g \forall n (f(\bar{g}(n)) > 0) \end{cases}$$

where f and g are functions on \mathbb{N} , and $\bar{g}(n)$ is the *number* $\langle g(0), \dots, g(n - 1) \rangle$, which we identify with the corresponding sequence.

We use μ instead of \exists^2 , also known as 2E : modulo full Kleene computability, they are equivalent, while in fragments like e.g. Gödel’s T, μ is more expressive than \exists^2 . Arithmetical definitions can be transformed to terms using μ , and our use of S will mainly be to isolate the well-founded segment of a relation.

2.2 Specifications for Realisers of Type 3

Without digging too deep into the theory of realisers, we can say that a *realiser* of a theorem of the form $\forall x \exists y (A(x) \rightarrow B(x, y))$ is a total function ϕ that maps

an x satisfying A to some y such that $B(x, y)$. We analyse the complexity of realisers of this kind for two classical theorems. We will use $C = 2^{\mathbb{N}} = \{0, 1\}^{\mathbb{N}}$ as the Cantor space.

1. Compactness of the Cantor space

For each $x \in C$, let $O_x \subseteq C$ be an open set containing x . Then there is a finite set $\{x_1, \dots, x_n\} \subseteq C$ such that $C \subseteq O_{x_1} \cup \dots \cup O_{x_n}$.

2. The Lindelöf Property of the Baire space

For each $x \in \mathbb{N}^{\mathbb{N}}$, let $O_x \subseteq \mathbb{N}^{\mathbb{N}}$ be an open set containing x . Then there is a sequence $\{x_i\}_{i \in \mathbb{N}}$ in $\mathbb{N}^{\mathbb{N}}$ such that $\mathbb{N}^{\mathbb{N}} \subseteq \bigcup_{i \in \mathbb{N}} O_{x_i}$.

In [16] we discuss how these statements relate to the classical theorems known as the *Heine-Borel theorem* and the *Lindelöf property* for subsets of \mathbb{R} . Here we will consider equivalent, styled versions suitable for interpretations over the standard typed structure of functionals with base type \mathbb{N} .

For the sake of simplicity, we will identify a finite sequence s of non-negative integers with its sequence number. If s is a finite binary sequence of length n , we let C_s be the set of functions $f \in C$ with $s = \bar{f}(n) = \langle f(0), \dots, f(n-1) \rangle$, while if s is a finite sequence of non-negative numbers, still of length n , we let B_s be the set of $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $\bar{f}(n) = s$. Any $F : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ will define open coverings of C resp. $\mathbb{N}^{\mathbb{N}}$. They are $\{C_{\bar{f}(F(f))} \mid f \in C\}$ resp. $\{B_{\bar{f}(F(f))} \mid f \in \mathbb{N}^{\mathbb{N}}\}$. Moreover, any general cover allocating an open set O_x containing x to each x in the spaces C or $\mathbb{N}^{\mathbb{N}}$ can, with an arithmetical construction, be replaced by one obtained from a functional as above, without simplifying the task of verifying compactness or the Lindelöf property.

We can then define what we mean by a realiser Θ for the compactness of C or a realiser Ξ for the Lindelöf property of $\mathbb{N}^{\mathbb{N}}$ by

Definition 1.

- Θ An instance of Θ will be a functional, which we also denote Θ , of type 3 such that if $F : C \rightarrow \mathbb{N}$, then $\Theta(F)$ is a finite sequence f_1, \dots, f_n in C such that the $C_{\bar{f}_i(F(f_i))}$'s cover C .
- Ξ An instance of Ξ will be a functional, which we also denote Ξ , of type 3 such that if $F : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$, then $\Xi(F)$ is an infinite sequence $\{f_i\}_{i \in \mathbb{N}}$ from $\mathbb{N}^{\mathbb{N}}$ such that $\{B_{\bar{f}_i(F(f_i))}\}_{i \in \mathbb{N}}$ covers $\mathbb{N}^{\mathbb{N}}$.

It is routine to rewrite these definitions to fit pure type 3, but for the sake of readability, we prefer not to do this routine.

Remark 1. It is common to define an open covering of C as a class \mathcal{O} of open sets such that

$$\forall f \in C \exists O \in \mathcal{O} (f \in O).$$

Then a functional F of type 2 such that

$$\forall f \in C \exists n \in \mathbb{N} \exists O \in \mathcal{O} (C_{\bar{f}(F(f))} \subseteq O)$$

can be considered as a realiser of the fact that \mathcal{O} is an open covering. In this sense, our realisers will take realisers of the assumptions and provide witnesses to the conclusions.

The definitions of Θ and Ξ may look similar, but there is an important difference: while checking if a finite set s_1, \dots, s_n of binary sequences induce a finite covering of C is computable at a low level, the set of countable sequences $\{s_i\}_{i \in \mathbb{N}}$ inducing a countable covering of $\mathbb{N}^{\mathbb{N}}$ will be complete Π_1^1 . This is reflected in Theorem 1(b) and can be seen as the cause of the increased computational strength of Ξ , compared to that of Θ . Even Θ will have some computational strength, though, as any instance of Θ will compute, in conjunction with μ , a realiser of transfinite recursion, see [13, 15], and will in particular compute functions that are not hyperarithmetical. However, any function uniformly computable in all instances of Θ will be hyperarithmetical, see [15] for a proof.

Remark 2. Our interpretation of the Lindelöf property of the Baire space may seem a bit odd, since we actually *start* with a countable cover $\{B_{\bar{f}(F(f))} \mid f \in \mathbb{N}^{\mathbb{N}}\}$. The original assumption in Lindelöf [10] was essentially as follows:

For $x \in E \subseteq \mathbb{R}^n$, let D_x be a disc with centre in x and a positive radius.

Then, selecting a countable set of such discs is equivalent to selecting a countable set of centres, so our formulation captures the original formulation of the Lindelöf property correctly.

The easy proof of the Lindelöf property will use $\Sigma_1^1[F]$ -comprehension and countable choice, and for proving that every subset of \mathbb{R}^n has the Lindelöf property, there do not seem to be any good alternatives. Our choice-free proof for $\mathbb{N}^{\mathbb{N}}$ can be seen as a special proof for the set of irrational numbers. It is well known that $\mathbb{N}^{\mathbb{N}}$ and the set of irrationals between 0 and 1 are homeomorphic via continued fractions.

2.3 Non-monotone Inductive Definitions

Let $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$, where $2^{\mathbb{N}}$ denotes the powerset of \mathbb{N} , identified with the set of characteristic functions. We may view F as a functional of type 2, ignoring the necessary coding. We may also consider F as an inductive definition:

Definition 2. Let $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$, and let α be an ordinal number. By recursion on α we define

$$\Gamma_\alpha(F) = \bigcup_{\beta < \alpha} F(\Gamma_\beta(F)).$$

The sequence $\Gamma_\alpha(F)$ indexed by ordinals is an increasing sequence of subsets of \mathbb{N} , with $\Gamma_0(F) = \emptyset$. For cardinality reasons, there is a countable ordinal α_0 , depending on F , such that $\Gamma_{\alpha_0}(F) = \Gamma_{\alpha_0+1}(F)$, i.e. $F(\Gamma_{\alpha_0}(F)) \subseteq \Gamma_{\alpha_0}(F)$.

Definition 3. We let Γ be the functional of type 3 defined by $\Gamma(F) = \Gamma_{\alpha_0}(F)$ where α_0 , depending on F , is as above.

We observe that if we let $G(X) = F(X) \setminus X$ for $X \in 2^{\mathbb{N}}$, then $\Gamma_\alpha(F) = \Gamma_\alpha(G)$ for all α , and in particular, $\Gamma(F) = \Gamma(G)$.

Remark 3. The functional F is *monotone* if $A \subseteq B \Rightarrow F(A) \subseteq F(B)$. Our inductive definitions are not necessarily monotone, so we call them *non-monotone* in general. Non-monotone inductive definitions were in particular studied in the late 60's and early 70's, see e.g. Sect. III of [4] with papers by Stål Aanderaa, Douglas Cenzer, Robin O. Gandy and Wayne Richter/Peter Aczel.

3 The Main Theorem

We will use Kleene's definition via the schemes S1–S9 (see [9], or alternatively [11]) as our formal notion of computability in a higher order context.

Theorem 1. *The five functionals are related as follows:*

- (a) *For any instance of Ξ , there is an instance of Θ uniformly computable in Ξ and μ .*
- (b) *S is uniformly computable in μ and any instance of Ξ .*
- (c) *Γ is uniformly computable in S and any instance of Θ .*
- (d) *There is an instance of Ξ computable in Γ .*

Proof. (a) is easy. Computing a finite covering from a countable covering of C is computable even in the Turing sense. Given a covering of C we extend this to a covering of $\mathbb{N}^{\mathbb{N}}$ such that each $f \notin C$ is covered by an open set disjoint from C , and then Ξ provides a countable sub-covering of the original covering of C .

In order to prove (b), observe that if $A \subseteq \mathbb{N}$ is Σ_1^1 , there will, for each $a \in \mathbb{N}$ be a closed set $T_a \subseteq \mathbb{N}^{\mathbb{N}}$ such that $a \in A \Leftrightarrow T_a \neq \emptyset$. For each a , we define F_a such that $B_{\bar{f}(F_a(f))} \cap T_a = \emptyset$ whenever $f \notin T_a$, and zero on T_a . Then $a \in A$ if and only if $\Xi(F_a)$ contains an element in T_a .

The proof of (c), which is the main technical achievement of this note, will be given in a separate section, see below, so let us prove (d).

We need the *Kleene-Brouwer ordering*: If s and t are finite sequences of integers, then $s \prec_{KB} t$ if s is an *extension* of t or if s is below t in the lexicographical ordering. If $F : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ and A is a set of finite sequences s , we let $G(A) = \{\bar{f}_A(F(f_A))\}$, where f_A is the least function that properly bounds $\bigcup_{s \in A} B_s$, provided that this set is bounded in the lexicographical ordering of $\mathbb{N}^{\mathbb{N}}$. If not, we let $G(A) = \emptyset$. f_A is definable from A in an arithmetical manner as follows:

Let $g_A(0)$ be the maximal value of s_0 for $s \in A$. By recursion, and for as long as it is possible, let $g_A(n)$ be the maximal integer a such that for some $s \in A$, s extends $\langle g_A(0), \dots, g_A(n-1), a \rangle$.

If this process goes on forever, let $f_A = g_A$. If step n cannot be carried out, let $f_A(m) = g_A(m)$ for $m < n-1$, $f_A(n-1) = g_A(n-1) + 1$ and $f_A(m) = 0$ for $n \leq m$.

Each set $\Gamma_\alpha(G)$ will be a set of finite sequences well-ordered by the Kleene-Brouwer ordering, and if $\alpha < \beta$, then $\Gamma_\beta(G)$ will be an end extension of $\Gamma_\alpha(G)$ with respect to this ordering. Moreover, the union $\bigcup_{s \in \Gamma_\alpha(G)} B_s$ will be an initial segment O_α of $\mathbb{N}^{\mathbb{N}}$ under the lexicographical ordering of $\mathbb{N}^{\mathbb{N}}$. Finally, the function

$f_\alpha = f_{\Gamma_\alpha(G)}$ is definable from $\Gamma(G)$ uniformly in the finite sequence s_α added at stage α , following the procedure for f_A where $A = \{s \in \Gamma(G) \mid s \prec_{\text{KB}} s_\alpha\}$. Thus we may arithmetically extract an enumeration of the set $\{f_\alpha \mid s_\alpha \in \Gamma(G)\}$, and this will be the output of our $\Xi(F)$. \square

Remark 4. Borel’s proof in [2] essentially starts with an open covering of $[0, 1]$, builds up a larger and larger half-open interval $[0, x_\alpha]$ by, at each stage adding an open set containing x_α . Borel assumes that the original covering is countable, but this is not needed in order to show that the process must stop at some countable ordinal. Finally, through backtracking the process, Borel uses that there are no infinite descending sequences of ordinals in order to extract the finite covering. The point is that Borel’s construction is deterministic and choice free, the finite sub-covering is fully determined by the map $x \mapsto O_x$ through his proof.

Remark 5. We see that modulo computability relative to μ there is a minimal instance of Ξ , and this one is equivalent to Γ . On the other hand, the analysis of computability in Θ in [13, 15] shows that there is no minimal instance of Θ . However, if one takes the instance Θ_0 that is naturally obtained using Borel’s proof, then S will be computable in Θ_0 and μ (see [13]), and consequently, Θ_0 is also equivalent to Γ modulo μ . A closer inspection of the proofs will show that we only need a tiny fragment of Kleene-computability to establish these equivalences, we do not need the scheme for primitive recursion and the scheme of enumeration.

3.1 The Proof of Theorem 1(c)

We will see how we may compute the inductive closure $\Gamma(F)$ from F , Θ and S . So, let F be given. For the sake of notational simplicity, we assume that $F(A)$ is disjoint from A for all A .

Associated with the sequence $\Gamma_\alpha(F)$, we have the relation $\Delta(F)$ on $\Gamma(F)$ defined by $\Delta(F)(a, b)$ if the least β such that $a \in \Gamma_\beta(F)$ is less than or equal to the least γ such that $b \in \Gamma_\gamma(F)$. The relation $\Delta(F)$ is inductively definable as well, with approximations $\Delta_\alpha(F)$ defined on $\Gamma_\alpha(F)$.

To be more precise, we may use the operator \tilde{F} to define $\Delta(F)$ as follows:

If R is a binary relation, we let D_R be the domain of R , here defined as the set of a such that for some b , either $(a, b) \in R$ or $(b, a) \in R$. Then $\tilde{F}(R)$ will, by definition, consist of all pairs (a, b) where $a \in D_R$ and $b \in F(D_R)$ together with all pairs (a, b) where both a and b are in $F(D_R)$. Instead of defining $\Gamma(F)$ directly with the help of Θ , we extract $\Delta(F) = \Gamma(\tilde{F})$ from F , S , μ and Θ .

We now need to establish some notation and general machinery: The advantage of $\Delta(F)$ is that it is a *prewellordering*. A *preordering* is a binary relation R that is both transitive and reflexive, and such that for all $a, b \in D_R$ we have $R(a, b)$ or $R(b, a)$ (or both). The corresponding *strict* relation will be $R_s(a, b) \leftrightarrow R(a, b) \wedge \neg R(b, a)$, and a is in the *well founded part* W_R if $a \in D_R$ and there is no R_s -descending sequence starting with a .

If R is a preordering and $a \in D_R$, we let $D_R[a] = \{b \in D_R \mid R_s(b, a)\}$ and we let $[a]_R = \{b \mid R(a, b) \wedge R(b, a)\}$. These are the *initial segments* and *layers* (or *equivalence classes*) of D_R .

We will now see how we can code all this with functions $f : \mathbb{N} \rightarrow \{0, 1\}$:

Definition 4

- (a) Let $f : \mathbb{N} \rightarrow \{0, 1\}$. Let $R_f(a, b)$ if $f(\langle a, b \rangle) = 1$. We write D_f for D_{R_f} .
- (b) We say that f is in PREO if R_f is a preordering.
- (c) If $f \in \text{PREO}$ and $a \in D_f$, we let $D_f[a] = D_{R_f}[a]$ and $[a]_f = [a]_{R_f}$
- (d) If $f \in \text{PREO}$, we write W_f for W_{R_f} , $R_f[a]$ for the restriction of R_f to $D_f[a]$ and $R_f[W]$ for the restriction of R_f to W_f .

We now link this to the inductive operator F , which we still assume to be fixed:

Lemma 1. *Let $f \in \text{PREO}$. The following are equivalent:*

- 1. $W_f = \Gamma(F)$ and $R_f[W] = \Delta(F)$
- 2. (i) For all $a \in W_f$ we have that $[a]_f = F(D_f[a])$
 (ii) $F(W_f) = \emptyset$.

The proof is trivial.

Lemma 2. *Let F be as above. There is a type 2 functional $G : \{0, 1\}^{\mathbb{N}} \rightarrow \mathbb{N}$, uniformly computable in S and F , such that for any instance of Θ , $\Theta(G)$ will contain an element f such that $f \in \text{PREO}$ and such that W_f and $R_f[W]$ either satisfy (i) and (ii) in Lemma 1, or has a proper initial segment that does so.*

Proof. We define G by cases, and, at the same time, explain what is achieved in that case. For each case, we assume that the previous cases do not apply. Whenever we write “let a, b etc. be such and such”, we can make the definition precise by applying numerical search, i.e. using μ , which is computable in S .

The aim is to define G such that whenever $G(f) > 0$, then $\bar{f}(G(f))$ has no extension g that is in PREO and such that $(\Gamma(F), \Delta(F))$ is equal to (D_g, R_g) , while when $G(f) = 0$, we can define $(\Gamma(F), \Delta(F))$ from f , using μ , S and F . Thus, in order to induce a finite set of neighbourhoods covering the one g coding $(\Gamma(F), \Delta(F))$, $\Theta(G)$ must contain an f with $G(f) = 0$, and from which we can compute $(\Gamma(F), \Delta(F))$ using μ and S .

The first part of the aim is met by, for each f with $G(f) > 0$, letting $G(f) > \langle a, b \rangle$ for some pair (a, b) such that $f(\langle a, b \rangle) = 0 \Leftrightarrow \Delta(F)(a, b)$.

- 1. If f is not in PREO, there will be a number n such that $\bar{f}(n)$ cannot be extended to any $f' \in \text{PREO}$. We let $G(f) = n$ for the least such n .
- 2. If $(\Gamma(F), \Delta(F))$ is all of, or an initial segment of (D_f, R_f) , then $G(f) = 0$. This can be checked using F , S and μ , since if this is the case, it will be the case also for $(W_f, R_f[W])$, and from this relation, all initial segments are arithmetically defined. This case clearly satisfies our aim.
- 3. For some $a \in W_f$, we have that $[a]_f \neq F(D_f[a])$. Select an R_f -minimal such a . Then $(D_f[a], R_f[a])$ is a proper initial segment of $(\Gamma(F), \Delta(F))$, proper since we are not in Case 2. There will be two subcases:

- (i) $[a]_f \subset F(D_f[a])$. Since the sets are not equal, let b be in $F(D_f[a])$, but not in $[a]_f$. If $f(\langle b, b \rangle) = 0$, it suffices to let $G(f) = \langle b, b \rangle + 1$ to achieve our aim. If $f(\langle b, b \rangle) = 1$, we have that $b \in D_f$ while b is neither R_f -equivalent to, nor R_f -less than a , so we must have $f(\langle a, b \rangle) = 0$. Since a and b appear at the same stage in the F -recursion, it suffices to let $G(f) = \langle a, b \rangle + 1$.
 - (ii) Otherwise. We may then as well assume that $a \notin F(D_f[a])$. Since we are not in Case 2, there is a $b \in F(D_f[a])$. If $f(\langle b, b \rangle) = 0$, we may let $G(f) = \langle b, b \rangle + 1$. If $f(\langle b, b \rangle) = 1$, $b \in D_f$, and we will use that R_f is a preordering. We must have that $f(\langle a, b \rangle) = 1$, since $b \notin D_f[a]$. But we will not have $\Delta(F)(a, b)$ since $b \in D_f[a] \cup F(D_f[a])$, while a is not. Thus it suffices, in this case, to let $G(f) = \langle a, b \rangle + 1$.
4. In this remaining case, we have that W_f is one of the initial segments of $\Gamma(F)$, but not $\Gamma(F)$ itself. Let $a \in F(W_f)$. If $f(\langle a, a \rangle) = 0$, we let $G(f) = \langle a, a \rangle + 1$. This suffices since $\Delta(F)(a, a)$. If $f(\langle a, a \rangle) = 1$, we use that R_f is a preordering, $a \in D_f$, but a is not in the well founded part of R_f . In particular, there will be some b not in W_f such that $f(\langle b, a \rangle) = 1$ while $f(\langle a, b \rangle) = 0$. If we let $G(f) = \max\{\langle b, a \rangle, \langle a, b \rangle\} + 1$, we have ensured that we are in conflict with $\Delta(F)$.

We have defined $G(f)$ for all f , and the construction of G is such that if $G(f) > 0$, then either $f \notin \text{PREO}$ or for some (a, b) we have that $\langle a, b \rangle < G(f)$ and $f(\langle a, b \rangle) = 0 \Leftrightarrow \Delta(F)(a, b)$. Thus we cannot cover the Cantor space just by neighbourhoods given by $\hat{f}(G(f))$ for f with $G(f) > 0$. Consequently, $\Theta(G)$ must contain some f with $\Theta(f) = 0$. We can compute $\Gamma(F)$ uniformly in any such f , using μ and S . □

Theorem 1(c) is an immediate consequence of this lemma.

4 Speculations on Functionals of Type 3

Prior research on the computational strength of functionals of type 3 has, to our knowledge, been concentrated on the normal functionals (with 3E computable in them), the Superjump and continuous functionals like the fan functional and the Gandy-Hyland functional, also denoted Γ . Our project has exposed that the class of non-normal, countably based functionals of type 3 in which μ or 2E is computable contains objects that reflect actual mathematical strength in some sense. For instance, consequences of the uncountable Heine-Borel theorem HBU will have realisers of type 3 of their own, and the computational strength of these realisers may reflect, at least in some sense, the strength of these consequences.

The functional Λ , only briefly discussed in this note, is also of this nature. Λ will not be computable in any type 2 functional, and actually neither in the Superjump, but it is strictly weaker than Θ in the sense that any instance of Θ computes an instance of Λ while the converse does not hold, even modulo μ .

One obvious challenge will be to classify the 1-section of Γ , i.e. the class of functions computable in Γ . A reasonable conjecture is that this 1-section

is generated in a gap-free manner, and that it corresponds to a class of functions computable by an Infinite Time Turing Machine, ITTM, within some time bound. A *gap* will be an interval of ordinals in which a generalised computing device computes no new subsets of \mathbb{N} , but will do so again at the end of the gap. The ITTM's were introduced by Hamkins and Kidder, but first appeared in published form in [5]. For a recent survey, see [19]. A more ambitious, but vague, conjecture is that there is some total, countably based functional Ψ of type 3 such that Kleene-computations relative to Ψ somehow reflects computations using ITTM's. For this to make sense, there should at least be a gap-structure in the generation of the 1-section of Ψ resembling that of ITTM's. We know that there is a gap-structure for computations relative to 3E , (see [12]), and by a Löwenheim-Skolem argument there will be countably based functionals with a similar structure, but beyond this, little is known. More is known about the gap-structure for ITTM's, see e.g. the recent [3].

Acknowledgements. I am grateful to Sam Sanders for initiating the project of which this note can be seen as an outcome, and for commenting fruitfully on the content of the note.

I will also thank John P. Hartley for sharing his thoughts around the early days of countably based functionals with me.

Finally, I will thank the two reviewers. They both gave valuable advice, and helped me improve the exposition and correct some typos and linguistic errors.

References

1. van den Berg, B., Briseid, E., Safarik, P.: A functional interpretation for nonstandard arithmetic. *Ann. Pure Appl. Log.* **163**, 1962–1994 (2012)
2. Borel, E.: Sur quelques points de la théorie des fonctions. *Ann. Sci. École Norm. Sup.* **3**(12), 9–55 (1895)
3. Carl, M., Durand, B., Lafitte, G., Ouazzani, S.: Admissibles in gaps. In: Kari, J., Manea, F., Petre, I. (eds.) *CiE 2017*. LNCS, vol. 10307, pp. 175–186. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58741-7_18
4. Fenstad, J.E., Hinman, P.: *Generalized Recursion Theory*. North-Holland/American Elsevier, Amsterdam (1974)
5. Hamkins, J.D., Lewis, A.: Infinite time turing machines. *J. Symb. Log.* **65**, 567–604 (2000)
6. Hartley, J.P.: *Recursion on Countably Based Functionals*. Thesis, University of Leeds (1982)
7. Hartley, J.P.: The countably based functionals. *J. Symb. Log.* **48**, 458–474 (1983)
8. Hartley, J.P.: Effective discontinuity and a characterisation of the superjump. *J. Symb. Log.* **50**, 349–358 (1985)
9. Kleene, S.C.: Recursive functionals and quantifiers of finite types I. *Trans. Am. Math. Soc.* **91**, 1–52 (1959)
10. Lindelöf, E.: Sur Quelques Points De La Théorie Des Ensembles. *Comptes Rendus*, pp. 697–700 (1903)
11. Longley, J., Normann, D.: *Higher-Order Computability*. Springer, Heidelberg (2015). <https://doi.org/10.1007/978-3-662-47992-6>

12. Moldestad, J.: Computations in Higher Types. LNM, vol. 574. Springer, Heidelberg (1977). <https://doi.org/10.1007/BFb0087822>
13. Normann, D., Sanders, S.: Nonstandard Analysis, Computability Theory, and Their Connections. arXiv: <https://arxiv.org/abs/1702.06556> (2017, submitted)
14. Normann, D., Sanders, S.: The strength of compactness in Computability Theory and Nonstandard Analysis. arXiv: <https://arxiv.org/abs/1801.08172> (2018, submitted)
15. Normann, D., Sanders, S.: Nonstandard Analysis, Computability Theory, and Metastability, in preparation
16. Normann, D., Sanders, S.: On the mathematical and foundational significance of the uncountable. arXiv: <http://arxiv.org/abs/1711.08939> (2017, submitted)
17. Sanders, S.: The Gandy-Hyland functional and a computational aspect of nonstandard analysis. *Computability* **7**, 7–43 (2018)
18. Sanders, S.: Metastability and higher-order computability. In: Artemov, S., Nerode, A. (eds.) LFCS 2018. LNCS, vol. 10703, pp. 309–330. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72056-2_19
19. Welch, P.D.: Transfinite machine models. In: Downey, R. (ed.) Turing’s Legacy. ASL Lecture Notes in Logic, vol. 42, pp. 493–529. Cambridge University Press, Cambridge (2014)



Enumeration Degrees and Topology

Arno Pauly^{1,2}(✉)

¹ Swansea University, Swansea, UK

Arno.M.Pauly@gmail.com

² University of Birmingham, Birmingham, UK

Abstract. The study of enumeration degrees appears prima facie to be far removed from topology. Work by Miller, and subsequently recent work by Kihara and the author has revealed that actually, there is a strong connection: Substructures of the enumeration degrees correspond to σ -homeomorphism types of second-countable topological spaces. Here, a gentle introduction to the area is attempted.

1 Enumeration Reducibility

Enumeration reducibility is a computability-theoretic reducibility for subsets of \mathbb{N} introduced by Friedberg and Rogers [10].

Definition 1. $A \leq_e B$ iff there is a c.e.-set W such that:

$$A = \{n \in \mathbb{N} \mid \exists k \in \mathbb{N} \exists m_0, \dots, m_k \in B \langle n, m_0, \dots, m_k \rangle \in W\}$$

where $\langle \rangle : \mathbb{N}^* \rightarrow \mathbb{N}$ is some standard coding for tuples of arbitrary length. We write \mathcal{E} for the collection of enumeration degrees.

The intuitive idea is that the elements of W are rules, that upon observing the presence of some finite collection of numbers in B let us conclude the presence of some number in A . Contrasted to Turing reducibility, we note that the symmetry between presence and absence is broken: We only receive positive information about B , and only need to provide position information about A .

Looking at individual sets, we find that enumeration reducibility and Turing reducibility appear very different: For any combination of enumeration and Turing reductions between A and B , we can construct sets realizing that combination. On the level of the degree structures, both \mathcal{E} and the Turing degrees (to be denoted by \mathcal{T}) are join-semilattices with the usual tupling function \oplus acting as join. We can make the following observation:

Observation 2. The map $A \mapsto A \oplus A^C$ induces a join-semilattice embedding of \mathcal{T} into \mathcal{E} .

A different intuitive interpretation of what enumeration reducibility is about is connected to this observation: Using the usual oracle Turing machines, we

obtain a reducibility not only between total functions, but between partial functions, too (e.g. [7, Sect. 11.3]). We obtain the degree structure \mathcal{E} like that, and the embedding just becomes the natural inclusion of the total functions into the partial functions.

A third notion of reducibility relevant for us is Medvedev reducibility. Rather than being about subsets of \mathbb{N} , we are dealing with subsets of $2^{\mathbb{N}}$ or $\mathbb{N}^{\mathbb{N}}$ here. We recall that computability for functions of type $F : \subseteq 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ can be defined via Type-2 Turing machines (which run forever, and have a write-once-only output tape), or equivalently via Turing functionals. In the latter case, we consider an oracle Turing machine M . We say that M computes F , if given oracle access to $p \in 2^{\mathbb{N}}$, it computes $n \mapsto F(p)(n)$.

Definition 3. $A \leq_M B$ if there is a computable $F : B \rightarrow A$. We denote the collection of Medvedev degrees by \mathcal{M} .

Both the Turing degrees and the enumeration degrees embed in natural ways into the Medvedev degrees: For the former, map $p \in 2^{\mathbb{N}}$ to $\{p\} \subseteq 2^{\mathbb{N}}$. For the latter, map $A \subseteq \mathbb{N}$ to $\{p \in \mathbb{N}^{\mathbb{N}} \mid \forall n \in \mathbb{N} \ n \in A \Leftrightarrow \exists k \in \mathbb{N} \ p(k) = n + 1\}$.¹ In words, to embed an enumeration degree into the Medvedev degrees, we move from a set $A \subseteq \mathbb{N}$ to the set of all enumerations of A . If we start with a Turing degree, embed it as enumeration degree, and then embed that as a Medvedev degree, we obtain the same degree (although not the same set) as when we move from Turing degrees directly to Medvedev degrees.

2 Represented Spaces and Generalized Turing Reductions

If we wish to delve deeper into the idea that the difference between Turing reducibility and enumeration reducibility is about having different information about the sets available, we are led to the notion of a *represented space*. Represented spaces are the means by which we introduce computability to the spaces of interest in computable analysis [44].

Definition 4. A *represented space* is a set X together with a partial surjection $\delta_{\mathbf{X}} : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow X$.

A function between represented spaces $\mathbf{X} = (X, \delta_{\mathbf{X}})$ and $\mathbf{Y} = (Y, \delta_{\mathbf{Y}})$ is just a set-theoretic function on the underlying sets. We say that $F : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$ is a *realizer* of $f : \subseteq \mathbf{X} \rightarrow \mathbf{Y}$, if $\delta_{\mathbf{Y}}(F(p)) = f(\delta_{\mathbf{X}}(p))$ for all $p \in \text{dom}(f \circ \delta_{\mathbf{X}})$. We then call a function between represented spaces *computable*, if it has a computable realizer. More on the theory of represented spaces is presented in [35].

A potential way to introduce Turing reducibility is to say that for $p, q \in \mathbb{N}^{\mathbb{N}}$ we have $p \leq_T q$ if there is a computable $F : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$ with $F(q) = p$. Not only does this approach align well with our embedding of \mathcal{T} into \mathcal{M} , it immediately suggests a generalization. Given that we have available to us a notion of partial computable functions between represented spaces in general, we can introduce:

¹ Using $p(k) = n + 1$ rather than $p(k) = n$ is necessary to deal with the empty set in a uniform way.

Definition 5 (Kihara and Pauly [24]). For represented spaces \mathbf{X} , \mathbf{Y} , and elements $x \in \mathbf{X}$, $y \in \mathbf{Y}$, say that x is reducible to y (written $x^{\mathbf{X}} \leq_T y^{\mathbf{Y}}$), if there is a computable $F : \subseteq \mathbf{Y} \rightarrow \mathbf{X}$ with $F(y) = x$.

Unraveling the definitions, we see that $x^{\mathbf{X}} \leq_T y^{\mathbf{Y}}$ is just a fancy way of saying $\delta_{\mathbf{X}}^{-1}(x) \leq_M \delta_{\mathbf{Y}}^{-1}(y)$. Considering the degree structure for arbitrary represented spaces just gives back \mathcal{M} again. However, we do gain two advantages: On the one hand, we can now deal with the degrees of meaningful mathematical objects, rather than faceless subsets of Baire space. On the other hand, we can look at the degrees of all points in more restricted classes of represented spaces than all of them at once, and investigate which degrees are present in those.

Two particular classes of represented spaces have received significant attention, computable metric spaces and countably-based spaces². In both, representations can be constructed in a canonical way from other data about the space:

Definition 6. A computable metric space is a metric space (X, d) together with a dense sequence $(a_i)_{i \in \mathbb{N}}$ such that $(i, j) \mapsto d(a_i, a_j) : \mathbb{N}^2 \rightarrow \mathbb{R}$ is computable. Its associated representation is defined via $\delta(p) = x$ iff $\forall i \in \mathbb{N} \ d(x, a_{p(i)}) < 2^{-i}$.

Definition 7. Given an enumeration $(B_n)_{n \in \mathbb{N}}$ of a basis of a T_0 topological space, its induced representation is given by $\delta(p) = x$ iff:

$$\{n \mid x \in B_n\} = \{n \mid \exists i \ p(i) = n\}$$

In effective descriptive set theory [32], a slightly different effectivization of metric spaces is used, namely recursively presented metric spaces. The difference disappears if we add a completeness-requirement and forget the specific choice of metric [14]. We then arrive at computable Polish spaces. The complete version of countably-based T_0 spaces are the quasi-Polish spaces [4].

The generalized reducibility restricted to computable metric spaces was studied by Miller [30]. He showed that there are indeed degrees of points in computable metric spaces (he called these *continuous* degrees) that are not Turing degrees, and obtained many results about them. Since there are universal Polish spaces (for example, the Hilbert cube $[0, 1]^\omega$ or the space of continuous functions $\mathcal{C}([0, 1], [0, 1])$), we can in particular define:

Definition 8. We call $\text{Spec}([0, 1]^\omega)$ the continuous degrees.

For countably-based spaces, we find that there too exists a universal space. Let $\mathcal{O}(\mathbb{N})$ be the space of open subsets of \mathbb{N} , i.e. of all subsets of \mathbb{N} represented via $\psi(p) = \{n \mid \exists i \ p(i) = n + 1\}$. Alternatively, $\mathcal{O}(\mathbb{N})$ can be seen as derived from the countably-based topology generated by the subbasis $\{\{U \subseteq \mathbb{N} \mid n \in U\} \mid n \in \mathbb{N}\}$, i.e. carrying the Scott topology.

² There are some variations here regarding what aspects are required to be effective. Typical names used in the literature are *effective topological space*, *computable topological space* or *effectively enumerable topological space*, see e.g. [25, 45]. These details do not matter for our purposes.

Observation 9. $\text{Spec}(\mathcal{O}(\mathbb{N})) = \mathcal{E}$

Thus, we see that the enumeration degrees are the degrees of points in countably-based spaces.

3 σ -Homeomorphisms

While we have seen that enumeration degrees can naturally be conceived of as the degrees of points in countably-based spaces, we have not yet discussed how topological properties of a space interact with the degrees of its points. The relevant notion here is that of a σ -homeomorphism. We have both a computable and a continuous version of σ -homeomorphism.

Definition 10. *A represented space \mathbf{X} σ -embeds into a represented space \mathbf{Y} , if there is a countable partition $\mathbf{X} = \bigcup_{i \in \mathbb{N}} \mathbf{X}_i$ such that any \mathbf{X}_i embeds into \mathbf{Y} .³ If \mathbf{X} σ -embeds into \mathbf{Y} and vice versa, we call \mathbf{X} and \mathbf{Y} σ -homeomorphic.*

Definition 11 (Kihara and Pauly [24]). *For a represented space \mathbf{X} , let $\text{Spec}(\mathbf{X})$ be the set of degrees of points in \mathbf{X} .*

Theorem 12 (Kihara and Pauly [24]). *The following are equivalent for represented spaces \mathbf{X}, \mathbf{Y} :*

1. $\text{Spec}(\mathbf{X}) \subseteq \text{Spec}(\mathbf{Y})$
2. \mathbf{X} computably σ -embeds into \mathbf{Y}

Thus, we see that the degrees present in a space (above some oracle) just characterize its σ -homeomorphism type. For Polish spaces, the question of their σ -homeomorphism types has received significant attention. Here, the partition of \mathbf{X} in Definition 10 can even be chosen to be Π_2^0 . Intricate arguments from descriptive set theory [13, 33, 38] then show that for Polish spaces, σ -homeomorphism agrees with second-level Borel isomorphism. Jayne had explored second-level Borel isomorphism of Polish spaces in 1974 [20] motivated by applications in Banach space theory. While it is well-known that $2^{\mathbb{N}}$ and $[0, 1]^{\omega}$ are not σ -homeomorphic⁴, it remained open whether there were more σ -homeomorphism types of uncountable Polish spaces. This question was reraised by Motto-Ros [33] and by Motto-Ros et al. [34]. The context of [34] is the generalization of Wadge degrees. The question was answered using recursion-theoretic methods by Kihara and Pauly, yielding:

Theorem 13 (Kihara and Pauly [24]). *The poset $(\omega_1^{\leq \omega}, \subseteq)$ of countable subsets of the first uncountable ordinal ω_1 ordered by set-inclusion embeds into the poset of uncountable Polish spaces ordered by σ -embeddability.*

³ We have slightly deviated from the usual definition here. In topology, we would typically demand that the \mathbf{X}_i can be *disjointly* embedded into \mathbf{Y} . The difference can be removed by replacing \mathbf{Y} with $\mathbf{Y} \times \mathbb{N}$. The results we need from topology hold for either version, and the present one makes the connection to degree theory more elegant.

⁴ We now realize that this means that there are continuous degrees which are not Turing degrees!

4 Non-total Continuous Degrees

Miller had observed in [30] that all continuous degrees share a peculiar property, namely being *almost-total* (the name was coined later, in [2]). The name is explained by noting that an enumeration degree is called *total*, if it is in the range of the embedding of the Turing degrees.

Definition 14. *An enumeration degree d is almost-total, if for every total degree p with $p \not\leq_e d$ we find that $p \oplus d$ is total.*

The question of existence of non-total almost-total degrees *prima facie* is purely recursion-theoretic question. Miller's result shows that the existence of continuous degrees which are not Turing degrees gives a positive answer. Miller's proof of the existence of continuous non-total degrees in [30] proceeds by constructing a multi-valued function on $[0, 1]^\omega$ whose fixed-points are non-total, and invoking a generalization of Brouwer's fixed point theorem to conclude existence. It is, in particular, relying heavily on topological arguments.

A different proof follows from the observation by Day and Miller [8] that Levin's neutral measures from [27] (see also [11]) have non-total continuous degrees. A measure μ is called weakly *neutral*, if every point is μ -random – this in particular requires every point computable from μ to have positive measure. The existence of neutral measures is obtained via the Kakutani fixed point theorem.

We already mentioned that by Theorem 12 the existence of non-total continuous degrees is equivalent to saying that $2^{\mathbb{N}}$ and $[0, 1]^\omega$ are not σ -homeomorphic. This is in turn a consequence of a classic result in topological dimension theory [19]: A Polish space is countably-dimensional iff it is σ -homeomorphic to $2^{\mathbb{N}}$ – and the Hilbert cube is not countably-dimensional. We thus have three different proofs of the existence of the recursion-theoretic theorem that non-total almost-total degrees exist – and all of them invoke classic topological theorems.

That the existence of non-total almost-total degrees is proven via a seeming detour through the continuous degrees is not accident: Andrews et al. proved that the almost-total degrees are exactly the continuous degrees [2]. Their proof proceeds via a number of characterizations, essentially showing that every almost-total degree has a certain representative, and that the collection of these representatives forms an effective regular topological space. Schröder's effective metrization theorem [16, 39] then enables the conclusion that all these representatives have continuous degree.

5 G_δ -Spaces and Cototal Degrees

That enumeration degrees correspond to countably-based spaces, and Turing degrees to Cantor space, or more generally, countably-dimensional spaces could still be put aside as a superficial resemblance. We shall thus present further examples of topological spaces and substructures of the enumeration degrees both studied in their own right, and explain how they link up.

The total enumeration degrees can be characterized as having a representative $A \subseteq \mathbb{N}$ such that $A^C \leq_e A$. We can dualize this to get the *cototal* enumeration degrees as those having a representative such that $A \leq_e A^C$. Every total degree is cototal, but not vice versa. McCarthy [29] revealed various characterizations of the cototal degrees: They are the degrees of complements of maximal antichains in $\mathbb{N}^{<\omega}$, of (uniformly) e-pointed trees and of the languages of minimal subshifts. Here, a (uniformly) e-pointed tree is an infinite binary tree T such that every infinite path through T is (uniformly) \leq_e -above T . For more on degrees and minimal subshifts, see [21]. The cototal enumeration degrees are further studied in [1, 31].

A topological space is a G_δ -space, if every closed subset can be written as a countable intersection of open sets. If we consider only countably-based spaces, this is equivalent to saying that every closed subset is equal to the intersection of all open sets containing it. Every Polish space is G_δ , while neither the Sierpiński space \mathbb{S} nor $\mathcal{O}(\mathbb{N})$ are.

Theorem 15 (Kihara et al. [23]). *The degrees of points in countably-based G_δ -spaces are exactly the co-total degrees.*

In particular, we can conceive of the space of complements of maximal antichains in $\mathbb{N}^{<\omega}$ or the space of languages of minimal subshifts to be a universal G_δ -space, taking into account McCarthy’s results.

6 Graph-Cototal Degrees and the Cofinite Topology

Call an enumeration degree graph-cototal, if it contains a representative of the form $\text{Graph}(f)^C$ for some $f : \mathbb{N} \rightarrow \mathbb{N}$. Graph-cototal enumeration degrees were studied by Solon [42]⁵ in the context of quasi-minimal enumeration degrees⁶.

To find their topological counterpart, we turn to the cofinite topology on \mathbb{N} . Here, a subset of \mathbb{N} is open iff it is empty or cofinite. As a representation, we find that $\delta_{\text{cof}}(p) = n$ iff $\{p(i) \mid i \in \mathbb{N}\} = \mathbb{N} \setminus \{n\}$ produces the desired represented space \mathbb{N}_{cof} . The space \mathbb{N}_{cof} is perhaps the simplest example of a topological space satisfying the T_1 separation axiom (every singleton is closed), but not the T_2 separation axiom (being Hausdorff, i.e. every two points are separated by disjoint open sets).

Observation 16 (Kihara et al. [23]). *$\text{Spec}(\mathbb{N}_{\text{cof}}^\omega)$ contains exactly the graph-cototal enumeration degrees.*

The question has been raised⁷ whether all almost-total degrees are graph-cototal. Via the aforementioned results and Theorem 12, we can rephrase this question to a topological one:

⁵ Solon uses the name cototal instead of graph-cototal, which we have already used for a different concept above.

⁶ An enumeration degree is quasi-minimal, if it is non-computable, but every total degree below is computable.

⁷ This open question was brought to the author’s attention by Joe Miller.

Question 17. Does $[0, 1]^\omega$ σ -embed into $\mathbb{N}_{\text{cof}}^\omega$?

What we can rule out easily is an actual embedding of $[0, 1]^\omega$ into $\mathbb{N}_{\text{cof}}^\omega$, due to the following:

Theorem 18 (Sierpiński, see e.g. [9, Theorem 6.1.27]). *Let \mathbf{X} be a connected compact metric space. Then every continuous $f : \mathbf{X} \rightarrow \mathbb{N}_{\text{cof}}^\omega$ is constant.*

By a classic theorem from topological dimension theory (see [19]), the Hilbert cube cannot be the countable union of disconnected spaces. At first glance, this may appear to answer Question 17. However, there are connected metric spaces containing no non-trivial connected subspaces. Spaces with the latter property are called *punctiform*, and a construction of a connected punctiform space is found as [26, Example 1.4.8]. For a further discussion of punctiform spaces and additional references, see [28].

7 The Lower Reals and Semirecursive Sets

The lower reals $\mathbb{R}_<$ are the real numbers, where $x \in \mathbb{R}$ is represented via an increasing sequence $(q_i)_{i \in \mathbb{N}}$ of rationals with $\sup_{i \in \mathbb{N}} q_i = x$. Equivalently, they are the reals equipped with the (countably-based) topology $\{\{y \in \mathbb{R} \mid y < x\} \mid x \in \mathbb{R}\} \cup \{\emptyset, \mathbb{R}\}$. This spaces appears naturally when performing the Dedekind-construction of the reals in a constructive setting, and has a central role in the development of measure theory via valuations (e.g. [6, 37]).

Recall from [22] that a set $A \subseteq \mathbb{N}$ is called *semirecursive*, if there is a computable function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ such that for all $n, m \in \mathbb{N}$ we find $f(n, m) \in \{n, m\}$, and if $n \in A$ or $m \in A$, then $f(n, m) \in A$. Combining results by Jockusch [22] and by Ganchev and Soskova [12] shows:

Theorem 19. *The semirecursive enumeration degrees are precisely $\text{Spec}(\mathbb{R}_<)$.*

For most natural spaces, taking finite products does not change their σ -homeomorphism type. In other words, the products of any two degrees of their corresponding spectra will lie in the spectrum again. The situation is different for semirecursive sets. It is readily seen that $2^{\mathbb{N}}$ does not embed into $\mathbb{R}_<$, whereas $\mathbb{R}_< \times \mathbb{R}_>$ contains a copy of \mathbb{R} (and thus of $2^{\mathbb{N}}$). In degree language, almost all semirecursive degrees are not total, whereas any total degree can be written as a product of two semirecursive degrees. Using geometric reasoning, one can obtain the following general result:

Theorem 20 (Kihara and Pauly [24]). *Let \mathbf{X} be uncountable, countably-based and T_1 . Then the spectra of $\mathbf{X} \times \mathbb{R}_<^n$ and $\mathbb{R}_<^{n+1}$ are incomparable.*

One recursion-theoretic corollary is that for any n , there is a degree arising as the product of $n + 1$ semirecursive degrees, but not of n semirecursive degrees and one graph-cototal degree, and vice versa. Studying the spectrum of spaces $\mathbf{X} \times \mathbb{R}_<$ also led to a generalization ([24, Lemma 8.2]) of Arslanov et al. result [3] that if a real x is neither left-c.e. nor right-c.e., then $x \in \mathbb{R}_<$ is quasi-minimal.

8 And More...

In [23], several countably-based spaces from “Counterexamples in topology” [43] had their spectra classified in recursion-theoretic terms, including the Arens square, the Gandy-Harrington space, Roy’s space and the relatively-coprime topology on the integers. One can lift the notion of quasi-minimality to spaces: a non-computable point $x \in \mathbf{X}$ is called \mathbf{Y} -quasi-minimal, if x computes no non-computable point in \mathbf{Y} . Various existence result for such quasi-minimal points are provided in [23].

We can also leave behind the realm of enumeration degrees and countably-based spaces, and study degrees in non-countably-based spaces. The spectrum of $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ exceeds the enumeration degrees; we can show this by lifting a diagonalization argument from Hinman [17] from partial functions on $2^{\mathbb{N}}$ to partial functions on $\mathcal{O}(\mathbb{N})$.

Non-countably-based spaces can be very resistant to the usual descriptive set theoretic methods. Hoyrup [18] has shown that already the lowest levels of the Borel hierarchy behave very differently for $\mathcal{O}(\mathbb{N})$ than their usual behaviour on quasi-Polish spaces⁸. In [5, 41] various hierarchies of non-countably-based represented topological spaces are explored. It is an open task to explore how these align with hierarchies of spectra.

Two further approaches to non-countably-based spaces in sight are to generate examples via the sequential de Groot dual [15] (essentially, given a T_1 -space, consider the space of singletons as a subspace of its space of closed subsets); or via countably cs-networks. It was shown by Schröder that the existence of these characterize the topological spaces than can be obtained as represented spaces in [40].

Acknowledgments. My understanding of the subject material has tremendously benefited from a multitude of discussions and talks. Foremost, I am grateful to Takayuki Kihara, but also to Matthew de Brecht, Mathieu Hoyrup, Steffen Lempp, Joseph Miller, Keng Meng Selwyn Ng and Mariya Soskova.

The author received support from the MSCA-RISE project “CID - Computing with Infinite Data” (731143) and the Marie Curie International Research Staff Exchange Scheme *Computable Analysis*, PIRSES-GA-2011-29496.

References

1. Andrews, U., Ganchev, H.A., Kuyper, R., Lempp, S., Miller, J.S., Soskova, A.A., Soskova, M.I.: On cototality and the skip operator in the enumeration degrees (preprint). <http://www.math.wisc.edu/~msoskova/preprints/cototal.pdf>
2. Andrews, U., Igusa, G., Miller, J.S., Soskova, M.I.: Characterizing the continuous degrees (2017, preprint). <http://www.math.wisc.edu/~jmiller/Papers/codable.pdf>
3. Arslanov, M.M., Kalimullin, I.S., Cooper, S.B.: Splitting properties of total enumeration degrees. *Algebra Log.* **42**(1), 1–13 (2003)


⁸ This raises the question how exactly one ought to define the Borel hierarchy in these spaces. One approach is found in [36].

4. de Brecht, M.: Quasi-Polish spaces. *Ann. Pure Appl. Log.* **164**(3), 354–381 (2013)
5. de Brecht, M., Schröder, M., Selivanov, V.: Base-complexity classifications of QCB_0 -spaces. In: Beckmann, A., Mitrana, V., Soskova, M. (eds.) *CiE 2015. LNCS*, vol. 9136, pp. 156–166. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20028-6_16
6. Collins, P.: Computable stochastic processes. [arXiv:1409.4667](https://arxiv.org/abs/1409.4667) (2014)
7. Cooper, S.B.: *Computability Theory*. Chapman and Hall/CRC, Boca Raton (2004)
8. Day, A., Miller, J.: Randomness for non-computable measures. *Trans. AMS* **365**, 3575–3591 (2013)
9. Engelking, R.: *General Topology*. Heldermann, Berlin (1989)
10. Friedberg, R., Rogers, H.: Reducibility and completeness for sets of integers. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **5**, 117–125 (1959)
11. Gács, P.: Uniform test of algorithmic randomness over a general space. *Theor. Comput. Sci.* **341**(1), 91–137 (2005). <http://www.sciencedirect.com/science/article/pii/S030439750500188X>
12. Ganchev, H.A., Soskova, M.I.: Definability via Kalimullin pairs in the structure of the enumeration degrees. *Trans. Am. Math. Soc.* **367**(7), 4873–4893 (2015)
13. Gregoriades, V., Kihara, T., Ng, K.M.: Turing degrees in Polish spaces and decomposability of Borel functions (2016, preprint)
14. Gregoriades, V., Kispéter, T., Pauly, A.: A comparison of concepts from computable analysis and effective descriptive set theory. *Math. Struct. Comput. Sci.* (2016). [arXiv:1403.7997](https://arxiv.org/abs/1403.7997)
15. de Groot, J., Strecker, G., Wattel, E.: The compactness operator in general topology. In: *General Topology and its Relations to Modern Analysis and Algebra*, pp. 161–163. Academia Publishing House of the Czechoslovak Academy of Sciences (1967). <http://eudml.org/doc/221016>
16. Grubba, T., Schröder, M., Weihrauch, K.: Computable metrization. *Math. Log. Q.* **53**(4–5), 381–395 (2007)
17. Hinman, P.G.: Degrees of continuous functionals. *J. Symb. Log.* **38**, 393–395 (1973)
18. Hoyrup, M.: Results in descriptive set theory on some represented spaces. [arXiv:1712.03680](https://arxiv.org/abs/1712.03680) (2017)
19. Hurewicz, W., Wallman, H.: *Dimension Theory*. Princeton Mathematical Series, vol. 4. Princeton University Press, Princeton (1948)
20. Jayne, J.E.: The space of class α Baire functions. *Bull. Am. Math. Soc.* **80**, 1151–1156 (1974)
21. Jeandel, E., Vanier, P.: Turing degrees of multidimensional SFTs. *Theor. Comput. Sci.* **505**, 81–92 (2013). <http://www.sciencedirect.com/science/article/pii/S0304397512008031>
22. Jockusch, C.: Semirecursive sets and positive reducibility. *Trans. AMS* **131**(2), 420–436 (1968)
23. Kihara, T., Ng, K.M., Pauly, A.: Enumeration degrees and non-metrizable topology. (201X, in preparation)
24. Kihara, T., Pauly, A.: Point degree spectra of represented spaces. [arXiv:1405.6866](https://arxiv.org/abs/1405.6866) (2014)
25. Korovina, M.V., Kudinov, O.V.: Towards computability over effectively enumerable topological spaces. *Electr. Notes Theor. Comput. Sci.* **221**, 115–125 (2008)
26. Krupka, D.: *Introduction to Global Variational Geometry*. Elsevier, Amsterdam (2000)
27. Levin, L.A.: Uniform tests of randomness. *Sov. Math. Dokl.* **17**(2), 337–340 (1976)

28. Lipham, D.: Widely-connected sets in the bucket-handle continuum. [arXiv:1608.00292](https://arxiv.org/abs/1608.00292) (2016)
29. McCarthy, E.: Cototal enumeration degrees and their application to computable mathematics. In: Proceedings of the AMS (to appear)
30. Miller, J.S.: Degrees of unsolvability of continuous functions. *J. Symb. Log.* **69**(2), 555–584 (2004)
31. Miller, J.S., Soskova, M.I.: Density of the cototal enumeration degrees. *Ann. Pure Appl. Log.* (2018). <http://www.sciencedirect.com/science/article/pii/S0168007218300010>
32. Moschovakis, Y.N.: *Descriptive Set Theory, Studies in Logic and the Foundations of Mathematics*, vol. 100. North-Holland, Amsterdam (1980)
33. Motto-Ros, L.: On the structure of finite level and omega-decomposable Borel functions. *J. Symb. Log.* **78**(4), 1257–1287 (2013)
34. Motto-Ros, L., Schlicht, P., Selivanov, V.: Wadge-like reducibilities on arbitrary quasi-polish spaces. *Math. Struct. Comput. Sci.* 1–50 (2014). http://journals.cambridge.org/article_S0960129513000339, [arXiv:1204.5338](https://arxiv.org/abs/1204.5338)
35. Pauly, A.: On the topological aspects of the theory of represented spaces. *Computability* **5**(2), 159–180 (2016). [arXiv:1204.3763](https://arxiv.org/abs/1204.3763)
36. Pauly, A., de Brecht, M.: Descriptive set theory in the category of represented spaces. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 438–449 (2015)
37. Pauly, A., Fouché, W.: How constructive is constructing measures? *J. Log. Anal.* **9** (2017). <http://logicandanalysis.org/index.php/jla/issue/view/16>
38. Pawlikowski, J., Sabok, M.: Decomposing Borel functions and structure at finite levels of the Baire hierarchy. *Ann. Pure Appl. Log.* **163**(12), 1748–1764 (2012)
39. Schröder, M.: Effective metrization of regular spaces. In: Ko, K.I., Nerode, A., Pour-El, M.B., Weihrauch, K., Wiedermann, J. (eds.) *Computability and Complexity in Analysis*. Informatik Berichte, vol. 235, pp. 63–80. FernUniversität, Hagen (1998)
40. Schröder, M.: Extended admissibility. *Theoret. Comput. Sci.* **284**(2), 519–538 (2002)
41. Schröder, M., Selivanov, V.L.: Some hierarchies of QCB₀-spaces. *Math. Struct. Comput. Sci.* 1–25 (2014). [arXiv:1304.1647](https://arxiv.org/abs/1304.1647)
42. Solon, B.: Co-total enumeration degrees. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J.V. (eds.) *CiE 2006. LNCS*, vol. 3988, pp. 538–545. Springer, Heidelberg (2006). https://doi.org/10.1007/11780342_55
43. Steen, L.A., Seebach Jr., J.A.: *Counterexamples in Topology*, 2nd edn. Springer, Heidelberg (1978). <https://doi.org/10.1007/978-1-4612-6290-9>
44. Weihrauch, K.: *Computable Analysis*. Springer, Heidelberg (2000). <https://doi.org/10.1007/978-3-642-56999-9>
45. Weihrauch, K., Grubba, T.: Elementary computable topology. *J. Univ. Comput. Sci.* **15**(6), 1381–1422 (2009)



A Taxonomy of Deviant Encodings

Paula Quinon^(✉) 

Philosophy Department, Lund University, Lund, Sweden

paula.quinon@fil.lu.se

<http://www.paulaquinon.com>

Abstract. The main objective of this paper is to design a common background for various philosophical discussions about adequate conceptual analysis of “computation”.

Keywords: The Church-Turing Thesis · Deviant encodings
Fixed points of conceptual analysis

1 Introduction: Circularity in the Definition of Computability

The core of the problem discussed in this paper is the following: the Church-Turing Thesis states that Turing Machines formally explicate the intuitive concept of computability. The description of Turing Machines requires description of the notation used for the **input** and for the **output**. The notation used by Turing in the original account and also notations used in contemporary handbooks of computability all belong to the most known, common, widespread notations, such as standard Arabic notation for natural numbers, binary encoding of natural numbers or stroke notation. The choice is arbitrary and left unjustified. In fact, providing such a justification and providing a general definition of notations, which are acceptable for the process of computations, causes problems. This is so, because the comprehensive definition states that such a notation or encoding has to be computable. Yet, using the concept of computability in a definition of a notation, which will be further used in a definition of the concept of computability yields an obvious vicious circle.

This argument appears in discussions about what is an adequate or correct conceptual analysis of the concept of computability. Its exact form depends on the underlying picture of mathematics that an author is working with. My objective in this paper is, firstly to discuss various versions and aspects of this

I am indebted to Liesbeth de Mol and Giuseppe Primiero for inviting me to the Special Session in History and Philosophy of Computing at CiE 2018. I am also grateful to Patrick Blackburn (Roskilde) and Nina Gierasimczuk (KTH) for helpful comments on an earlier version of this paper. I am finally indebted to the anonymous reviewer for the valuable insight into possible new openings to which the subject matter of the paper can, and certainly will, lead.

problem, and then, secondly, to point towards possible solutions, both those proposed explicitly, and these which are relevant for us only implicitly, because they target some disguised version of the problem. Finally, I will indicate related topics and formulate some ideas arising from this discussion.

2 Background: Philosophical Framework and Terminology Clarification

It was Gödel's ambition to explicate the concept of "absolute" computation, that is a computation which is not formulated for a particular domain. However, all know today explications of computability refer to some sort of ω -progression. Philosophical investigations into the concept of natural number, and its use as basis for analysis of the concept of computation, involve – more or less explicitly – one or both layers:

1. the *syntactical* layer of numerals and
2. the *semantical* layer of abstract natural numbers.

A *denotation* δ maps the elements from one layer to the elements of the other layer. See [16].

The syntactical layer of numerals consists of a sequence of inscriptions susceptible to be subjected to computational manipulations. Various syntactical systems have different syntactical properties. *A priori*, without an interpretation, a sequence of random inscriptions is as good as a binary notation based on recursive combinations of zeros and ones. At the semantical layer natural numbers are always understood as some sort of abstract entities. It does not belong to the current endeavor to analyze all possible modalities of existence of abstract natural numbers, neither to decide, which of these modalities is the correct one. That would complicate the picture unnecessarily. However, in order to mark the ground, let me say that you can think of variety of things starting from types of inscriptions, conceptual content associated with numerals in the process of learning, cognitive content issued from the empirical experience, a system of mental representations or Platonic ideas remaining in the Platonic world of ideas. The denotation function matches numerals from the syntactical layer with natural numbers from the semantical one, so denotation is a cross-modal function, mapping physical inscriptions on abstract objects.

Computations can be defined on the syntactical layer, or semantical layer or on both of them combined. Functions from syntactical layer are called string-theoretic functions, functions from semantical layer are called number-theoretic functions.

This picture was explicitly proposed by [16] and, in a more or less implicit manner, appears in various other papers studying philosophical correlations between computability and natural numbers of such authors as [1, 3] or [15]. The terminology that various authors use differ from one account to another. My very first objective in this paper is to clarify the basic vocabulary.

Three notions come back in accounts of deviations related to the concept of computability are the following: “notation”, “encoding” and “semantics”.

For instance, when it comes to “notation”, in his paper entitled “Acceptable notation” [16] calls “notation” a syntactic sequence of numerals *together* with a denotational function. This is definitely more handy in his context, as he considers only computable sequences of inscriptions. Shapiro’s concern consists in defining adequate ways of associating those sequences with semantics in such a way that string-theoretic functions have clear number-theoretic counterparts. [4] by “notation” mean the triple: a sequence of inscriptions with a denotation function and the standard semantics. One notation differs from another one by syntactical features of the sequence of inscriptions. In this paper, I speak of notation in the context of purely syntactical layer of inscriptions. I speak of notation with denotation function or notation with interpretations when I introduce other layers.

When the full picture (all layers) is in place, I call “deviant encodings” deviations that happens on the syntactic level; I call “deviant semantics” deviations that happens on the semantic level; finally I call “unacceptable denotation function” deviations of the denotation function.

I divide the discussion in this paper to three general perspectives:

- I make an attempt to talk about numerals without any reference to natural numbers formulating a purely syntactical version of the problem;
- I reflect on what happens when one starts associating names with abstract objects, and in this context I look at the denotation function;
- I look at those realistic positions where abstract objects are taken as existing before being named.

3 The Lightest Shade of the Problem: Purely Syntactical Approach

The general presentation of the problem, proposed at the beginning of this paper, addresses the syntactical version of the issue. In this version, computations are understood as manipulations of inscriptions, functions on inscriptions are string-theoretic functions, *etc.* This version is easily conceptualised in the context of a machine. When a human agent is involved at any stage the computational process, there exists at least a theoretical possibility that purely mechanical computations exist.¹

The problem in its purely syntactical version can be formulated as follows. In a definition of Turing computability, one of the aspects that needs to be clarified is the characterization of notation that can be used as an input for a machine to process. If a Turing Machine is supposed to explicate the intuitive concept of computability it is necessary to explain, which sequence of numerals can be used as an input without the use of the concept of computability. That means,

¹ It might be claimed that humans always associate some meaning with symbols.

we cannot simply say: “sequences that can be used as input are the computable ones” as we have not yet defined what means “to be computable”.

Study of symbol manipulation as a mathematical endeavor has its place in history of philosophy. A clear distinction between syntax and semantics dates from [17]. In a different context, [6] designs methods to deal with a purely syntactical calculus. [10] introduced philosophical problems related to the rejection of all abstract entities.

To understand this situation better, ask this question: what one could tell the Skeptic, who thinks that there is no way of distinguishing an acceptable notation from an unacceptable one?

3.1 Nominalist Platonism Solution: Syntactic Entwining

One could tell the Skeptic that sequences of inscriptions exist and are free of interpretations independently of human beings apprehending them. However, even if we cannot see that from our earthy perspective, there is something specific about certain sequences of inscriptions. Some of the sequences are computable. Hence, this primarily nominalistic position has a Platonist dimension. I call it “Platonist Nominalism”².

Imagine that, similarly to the librarians from the Library of Babel, described by Borges - I am going to get back to the original Borges story a little bit later - you wander in the land of inscriptions. You might be a robot equipped with means to process sequences of inscriptions and after processing some of those sequences you get an expected output. You can also imagine a “Chinese Room world”³. In any of those cases, you have no means to formulate a general theoretical law enabling you to distinguish processable sequences from non-processable ones.

“Nominalist Platonism” faces similar problems as those encountered by concrete computations. Consider, for instance, presentation of concrete computations by [13].

In our ordinary discourse, we distinguish between physical systems that perform computations, such as computers and calculators, and physical systems that don't, such as rocks. Among computing devices, we distinguish between more and less powerful ones. These distinctions affect our behaviour: if a device is computationally more powerful than another, we pay more money for it. What grounds these distinctions? What is the principled difference, if there is one, between a rock and a calculator, or between a calculator and a computer? Answering these questions is more difficult than it may seem.

² The name “Nominalist Platonism” has been used in a different context by George Boolos in “Nominalist Platonism”, *Philosophical Review* 94(3): 327–344 (1985). I do not want to get into comparison here.

³ John Searle (1980), “Minds, Brains and Programs”, *Behavioral and Brain Sciences*, 3(3): 417–457.

The difficulty for both, nominalist Platonism and concrete computations, consists in distinguishing sequences of physical objects (inscriptions in the first case and whatever physical systems, in the second) that can play the role of subjects of computations from those that cannot. We know the difference between a rock and a computer, in as much the same way as we know the difference between computable and non-computable sequences of inscriptions. We make a difference between notations for natural numbers, random sequences of random inscriptions, and even weird permutations of numerals.

However, when it comes to formulating an explanation of why is this so, there is no easy way out. Nominalist Platonism suffers from a severe and incurable epistemological problem of the sort described in [2] in the context of full-blooded Platonism: we always face a choice between realistic ontology and epistemological access to abstract objects. It is not impossible to get both. “[S]omething must be said to bridge the chasm, created by [...] [a] realistic [...] interpretation of mathematical propositions... and the human knower,” he writes. For prima facie “the connection between the truth conditions for the statements of [our mathematical theories] and [...] the people who are supposed to have mathematical knowledge cannot be made out”. [2, p. 675], see also [7].

3.2 Analytic Solution: The Turing Blank-Type Restriction and Turing Notational Thesis

Some Skeptics will obviously not agree to accept the Nominalist Platonism solution and we shall not take it against them. The position is plausible and coherent, but it is not philosophically very fruitful. The lack of epistemological access is a strong and non fixable defect rendering Nominalist Platonism useless for everyday reasonings, decision making and, most importantly, computational practice.

In this paper, I will recall a solution that has been proposed in the context of computability. The answer that might help overcome Skeptics’ worries was originally proposed by Turing and recently reevaluated by Copeland & Proudfoot [8]. I will call it the “analytic solution”.

Copeland & Proudfoot reconstruct the way in which Turing implicitly guarantees that sequences of symbols that are processed by a machine are computable. Computable sequences are subjects of two constraints.

Firstly, the *Turing’s Blank-Tape Restriction*: “If the [Turing] machine is supplied with a blank tape and set in motion [...] the subsequence of the symbols printed by it which are of the first kind [i.e. binary digits] will be called the sequence computed by the machine. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the number computed by the machine” [18, p. 232].

Secondly, *Turing Notational Thesis*: “Any job of work that can be done by a human computer engaged in numerical calculation can be carried out equivalently by a human computer employing Turing-unary notation”.

When these two constraints are put together, the sequence that is in the input and the sequence that is in the output of the process are necessarily computable.

4 The Problem Gets One Shade Darker: Numerals Get Meanings

Let's now imagine that the Skeptic is convinced by the arguments that one can generate the sequence of inscriptions - or, more precisely, of numerals - in such a way that this structure is recursive. She can do it on a basis that she finds the most convincing (social, cognitive, Blank-Type Restriction, *etc.*), it will not play any role in my further reflections.

Now, let us assume that the Skeptic thinks that numerals have meanings, but she refuses to rely on arbitrary choices of a "standard" notation, a "standard" denotation function, and a "standard" semantics. Instead, she asks for providing her a way to distinguish acceptable notations from unacceptable ones. In consequence, the problem of deviation extends to the semantical level.

4.1 Everyday Solution: Arbitrary Choice of Notation and Denotation

The solution commonly adopted in the real world is to take arbitrary decisions, possibly based on social, cultural or cognitive reasons, of which notation and which denotation to use. We simply use a commonly known, well-recognised notation, whose denotation does not leave to us any doubts. This is the case of [4]. When those authors introduce the question of changing notation they narrow the problem to the practical issue of using rather one notation than another. For instance, they say, "multiplying numbers given in decimal numerals (expressing the product in the same form) is easier in practice than multiplying numbers given in something like Roman numerals". They also claim that "it is possible in principle to do it in any other notation, simply by translating the data from the difficult notation into an easier one, performing the operation using the easier notation, and then translating the result back from the easier to the difficult notation", which means there is a computational translation between the notations. Ideas of "deciphering" a notation and "the system now in common use" show that, according to the authors, defining a notation require an external, non-mathematical, non-formal, non-effective knowledge [4, p. 24].

Indeed, when they define a Turing machine they say:

A Turing machine is a specific kind of idealised machine for carrying out computations, especially computations on positive integers represented in monadic notation.

There is no theoretical explanation that would justify the choice of one notation together with a denotation function over another. One can, obviously, try to justify the choice by saying that numerals are social creations that children learn in social interactions or that there is a necessity of learning them in this specific order because of cognitive process, but all these justifications go beyond the theoretical and conceptual context, which is ours.

4.2 Formal Problem: Deviant Encodings and Deviant Semantics

It is highly doubtful if any Skeptic gets convinced by an arbitrary argument. There is no theoretical reason to favour one notation, accompanied by a denotation, over another. In consequence, the “lacuna” remains, but at another level. And here finally comes time to recall the story of the Library of Babel. Under Jorge Luis Borges description [1941] “the Library is total and that its shelves register all the possible combinations of the twenty-odd orthographical symbols (a number which, though extremely vast, is not infinite). [...] When it was proclaimed that the Library contained all books, the first impression was one of extravagant happiness. All men felt themselves to be the masters of an intact and secret treasure. [...] As was natural, this inordinate hope was followed by an excessive depression. The certitude that some shelf in some hexagon held precious books and that these precious books were inaccessible, seemed almost intolerable”.

The librarians usually know which books make sense and which do not. It is ok if a copy contains typos. However, the librarians do not understand reasons why some sequences are special. *A priori*, no one is able to tell the difference between books that tell the stories from books containing illegible information. Similarly, no one is able to tell the difference between sequences that are computable from those that are not.

However, if no external justification is adapted, we fall once more into vicious circle of a conceptual analysis. This type of deviations is a deviation of the meaning associated to the denotation function. There is no formal way of defining such a function without referring to the concept of computability. A traditional way of presenting this problem is clearly visible on the example of the Semantical Halting Problem introduced by [12] and discussed by [8].

The classical formulation of the Halting Problem, first described by [18] and named by Davis, provides a negative reply to the question of whether there exists a general procedure to decide if a given Turing Machine, or more generally a given computer program, will eventually stop. The proof goes by showing that assuming the opposite leads to contradiction. The classical formulation uses some intended and arbitrary, recursive, notation with standard semantics, *e.g.*, Arabic notation with standard interpretations.

Formulating the Halting Problem in purely syntactical terms is not really possible, because the input, even if purely syntactical, is being generated by a recursive procedure of encoding Turing Machines. Each numeral stands for the Turing Machines it encodes. Problems arise however, when semantical layer gets involved in the Semantical Halting Problem.

The Semantical Halting Problem is a Skeptical issue of the same sort as the problem of non-computable, but ω -ordered models of the first-order Peano Arithmetic introduced by [11] and [14], and discussed by [5]. In the presentation of the classical Halting Problem, machines are encoded in a standard way. The Semantical Halting Problem opens up to the possibility of using deviant - non-computable - encodings. Imagine, you have encoded Turing machines with some standard encoding. Then, it is Funes de Memoriosus – a character from another

Borges' story – who inherits the job from you. He is not following any recursive rules. Since his memory is infinite and he has trouble synthesising information, he names subsequent machines by a random name, and symbolises by a random inscription.

He told me that in 1886 he had invented an original system of numbering and that in a very few days he had gone beyond the twenty-four-thousand mark. [...] In place of seven thousand thirteen, he would say (for example) *Máximo Pérez*, in place of seven thousand fourteen, *The Railroad*; other numbers were *Luis Melián Lafinur*, *Olimar*, *sulphur*, *the reins*, *the whale*, *the gas*, *the caldron*, *Napoleon*, *Augustn de Veida*. In place of five hundred, he would say *nine*. Each word had a particular sign, a kind of mark; the last in the series were very complicated... I tried to explain to him that this rhapsody of incoherent terms was precisely the opposite of a system of numbers. I told him that saying 365 meant saying three hundreds, six tens, five ones, an analysis which is not found in the “numbers” *The Negro Timoteo* or *meat blanket*. Funes did not understand me or refused to understand me.

Tapes with Funes encoding are subsequently given to the Halting Machine. What then happens? The Halting Machine that processes encodings of Turing Machines is designed to process information in an algorithmic manner. If inputted with a given non-standard enumeration of Turing machines, the machine will process those non-computable encodings as it were standard notation.

No one will, obviously, have the idea of encoding Turing Machines with a non-standard encoding. However, the problem of distinguishing one encoding from another is the same as it was in the case of purely syntactical version of the problem. If “being computable by a Turing machine” is how computable is defined, one cannot use the concept of being computable in the definition. Is there any formal general way of distinguishing standard encodings from deviant encodings?

4.3 Way Out: Constraints on Denotation Function (Shapiro)

Shapiro in [16] defines computability on inscriptions and then searches for ways of constraining the denotation function in such a way that no uncomputable semantics can be reached. The first constraint is that between the syntactic and the semantic layer there is a bijection (one-to-one and onto). He shows then that the class of number-theoretic functions which are computable relative to every notation is too narrow, containing only rather trivial functions, and that the class of number-theoretic functions which are computable relative to some notation is too broad (containing, for example, every characteristic function [p. 15]). Since these constraints does not single out any standard notation, Shapiro introduces human factor: “under normal circumstances, a person engaged in computation is not merely following an algorithm. It is usually important, in particular, that the computist know the number-theoretic goal of the algorithm” [p. 18].

4.4 Model-Realistic Solution: Model-Theoretic Entwining

Some Skeptics could feel disappointed by lack of a purely formal solution. What Shapiro proposes is, in the end, human-based and handwavy. If this is the case, the Skeptic might find some peace in a solution recently proposed by [9]. Dean develops a model-theoretic realism for the concept of computation. He claims that there is no point in trying to find external arguments to distinguish between various standard and non-standard models of arithmetic, nor of recursive theory. We should rather use the richness of the model-theoretic universe for studying structural properties of the concept of computability.

5 The Darkest Shade of the Problem: Computations Happen on Abstract Objects

There is finally another type of Skeptical worry. In his paper, [15] puts forward the idea that a full account of computability necessitated to define both syntactic and semantic computability. He then formulates a “crucial lacuna” indicating the intrinsic impossibility of defining computability on inscriptions first, and then, on its basis, computability on abstract numbers.

The lacuna states that in a realistic picture, when computations are defined on inscriptions, that there is no non-circular way of defining what computability on natural numbers is, if we want to take computability as first applying it to strings of characters. Let me remind you, that in an epistemologically plausible picture, computability on numbers as abstract objects is defined via notation. In an “epistemologically plausible picture” abstract objects are approached via language, and not via a private insight.

Rescorla’s objective is to give an account of what a number-theoretic computability is. He works under three hypotheses, first, computability refers to numbers via notation (via numerals and with help of denotation function); second, Turing Machines manipulate syntactic entities; third, to specify which number-theoretic function a Turing Machine computes, one must correlate these syntactic entities with numbers. The problem is that the correlation must itself be computable, otherwise the Turing machine would compute uncomputable functions. And the circularity arises: if we propose the intuitive notion of computable relation between syntactic entities and numbers, then our analysis of computability is circular.

In consequence, Rescorla claims that computability needs to be defined as a property of abstract objects and shall be defined as such. Computability on abstract objects is defined via Church’s thesis with the axioms of the theory of recursivity. This is where a Skeptic can be consoled again by the model-theoretic entwining proposed by [9].

5.1 Moderate Realism Solution: Any Old ω -Sequence Will Do After All

In [3] – sequel to the famous [1] – the author takes a structuralist position and claims that abstract objects playing the role of natural numbers in the structure

do not need to form a recursive progression. According to Benacerraf, there is no reason to proclaim computability of the series of abstract entities. This is so, because it is always possible to enumerate these entities with a recursive series of names. But, and here we get back to the beginning, how can we know which sequences of numerals are actually recursive.

6 Conclusions

There is no final answer that will fully satisfy our Skeptics. Each analysis of the concept of computation ends up in a vicious circle, it has a conceptual fixed point and suffers from a diagonal problem. We should keep that in mind when attempting to define computation and its twin concept of natural number.

References

1. Benacerraf, P.: What numbers could not be. *Philos. Rev.* **74**(1), 47–73 (1965)
2. Benacerraf, P.: Mathematical truth. *J. Philos.* **70**(19), 661–679 (1973)
3. Benacerraf, P.: Recantation, or: any old ω -sequence would do after all. *Philosophia Math.* **4**, 184–189 (1996)
4. Boolos, G., Burgess, J., Jeffrey, R.: *Computability and Logic*. Cambridge University Press, Cambridge (2007)
5. Button, T., Smith, P.: The philosophical significance of Tennenbaum’s theorem. *Philosophia Math.* **20**(1), 114–121 (2012)
6. Carnap, R.: *Foundations of Logic and Mathematics*. The University of Chicago Press, Chicago (1939)
7. Clarke-Doane, J.: What is the Benacerraf problem? In: Pataut, F. (ed.) *Truth, Objects, Infinity: New Perspectives on the Philosophy of Paul Benacerraf*. LEUS, vol. 28, pp. 17–43. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45980-6_2
8. Copeland, J., Proudfoot, D.: Deviant encodings and Turing’s analysis of computability. *Stud. Hist. Philos. Sci.* **41**, 247–252 (2010)
9. Dean, W.: Models and computability. *Philosophia Math.* **22**(2), 143–166 (2013)
10. Goodman, N., Quine, W.V.: Steps toward a constructive nominalism. *J. Symb. Log.* **12**(4), 105–122 (1947)
11. Halbach, V., Horsten, L.: Computational structuralism. *Philosophia Math.* **13**(2), 174–186 (2005)
12. van Heuveln, B.: *Emergence and consciousness*. Ph.D. thesis, Binghamton University (2000)
13. Piccinini, G.: *Physical Computation: A Mechanistic Account*. Oxford University Press, Oxford (2015)
14. Quinon, P., Zdanowski, K.: Intended model of arithmetic. Argument from Tennenbaum’s theorem. In: Cooper, S., Loewe, B., Sorbi, A. (eds.) *Computation and Logic in the Real World*, CiE Proceedings (2007)
15. Rescorla, M.: Church’s thesis and the conceptual analysis of computability. *Notre Dame J. Form. Log.* **48**, 253–280 (2007)
16. Shapiro, S.: Acceptable notation. *Notre Dame J. Form. Log.* **23**(1), 14–20 (1982)

17. Tarski, A.: The concept of truth in formalized languages. In: Tarski, A. (ed.) *Logic, Semantics, Metamathematics*, pp. 152–278. Oxford Univeraity Press, Oxford (1936)
18. Turing, A.: On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **42**(1), 230–265 (1936)



Elementary Bi-embeddability Spectra of Structures

Dino Rossegger^(✉) 

Institute of Discrete Mathematics and Geometry, Technische Universität Wien,
Vienna, Austria

dino.rossegger@tuwien.ac.at

Abstract. We study elementary bi-embeddability spectra, the collection of Turing degrees of elementary bi-embeddable structures. We give examples of such spectra and compare them with related notions.

1 Introduction

We investigate families of Turing degrees realized by the elementary bi-embeddability type of countable structures, *bi-embeddability spectra of structures*. The study of families of degrees realized by structures is a central topic in computable structure theory. Richter [15] was the first to study degrees realized by isomorphic copies of a structure. Knight [12] introduced the notion of the degree spectrum of a structure, the set of degrees of its isomorphic copies. Since then the question of which sets of degrees are realized as degree spectra of structures has seen a lot of interest, for an overview of the topic see [5, Sect. 2]. Andrews and Miller [1] studied theory spectra, the set of models of a complete theory; Fokina et al. [7] studied Σ_n spectra of structures, the set of degrees of structures satisfying the same Σ_n formulas as a given structure, and recently Fokina et al. [6] initiated the study of bi-embeddability spectra of structures, the collection of degrees of bi-embeddable copies of a structure. In [7] the authors defined the general notion of a degree spectrum under an equivalence relation.

Definition 1. *Given a structure \mathcal{A} and an equivalence relation \sim , the degree spectrum of \mathcal{A} under \sim is*

$$DgSp_{\sim}(\mathcal{A}) = \{deg(\mathcal{B}) : \mathcal{B} \sim \mathcal{A}\}.$$

Given a complete theory T and a model \mathcal{A} of T , the theory spectrum of T is the degree spectrum under elementary equivalence of \mathcal{A} . We study *degree spectra under elementary bi-embeddability*, or short *elementary b.e. spectra*.

Two structures \mathcal{A} and \mathcal{B} are *elementary bi-embeddable*, $\mathcal{A} \cong \mathcal{B}$, if either is isomorphic to an elementary substructure of the other. We will also refer to the

The author was supported by the Austrian Science Fund FWF through project P 27527. He thanks the logic group of the University of Wisconsin – Madison, Ekaterina Fokina, Luca San Mauro and Nikolay Bazhenov for useful discussions.

notion of bi-embeddability. Two structures \mathcal{A} and \mathcal{B} are *bi-embeddable*, $\mathcal{A} \approx \mathcal{B}$, if either is isomorphic to a substructure of the other.

In this article we give several examples of collections of degrees which are or are not elementary bi-embeddability spectra. One of the goals of this research is to separate elementary bi-embeddability spectra with spectra under other equivalence relations that have been investigated. Using our examples we obtain that there are collections of degrees that are elementary bi-embeddability spectra but are not theory spectra, Σ_n spectra or spectra of atomic theories and vice versa. It is open whether every elementary bi-embeddability spectrum is an isomorphism spectrum and vice versa.

1.1 Notation

Our computability theoretic notation is standard and as in [17]; for model theoretic background we suggest [13]. We denote structures by calligraphic letters and their universe by the corresponding capital letter. All our structures are countable and relational and we identify a structure \mathcal{A} by its atomic diagram: the set of atomic formulas and negations of atomic formulas true of \mathcal{A} expanded by a constant symbol for every element in A under a suitable Gödel numbering. Thus, \mathcal{A} is a set of natural numbers and we can apply computability theoretic tools as usual.

2 Elementary Bi-embeddability Spectra

For many arguments presented in this article it is useful to assume that our languages are finite. Indeed in our scenario this assumption is justified since graphs are universal for elementary bi-embeddability spectra.

Theorem 2. *Given a countable structure \mathcal{A} in any language we can compute a graph $\mathcal{G}_{\mathcal{A}}$ such that $DgSp_{\approx}(\mathcal{A}) = DgSp_{\approx}(\mathcal{G}_{\mathcal{A}})$.*

Proof (Sketch). We use the coding given in [1, Proposition 2.2]. We assume without loss of generality that \mathcal{A} is in a relational language $\langle R_i \rangle_{i \in I}$ where each R_i has arity i . Given \mathcal{A} the graph $\mathcal{G}_{\mathcal{A}}$ consists of 3 vertices a, b, c where to a we connect the unique 3-cycle in the graph, to b the unique 5-cycle, and to c the unique 7-cycle. For each element $x \in A$ we add a vertex v_x and an edge $a \rightarrow v_x$. For every i tuple $x_1, \dots, x_i \in A$ we add chains of length $i + k$ for $1 \leq k \leq i$ where for each such chain y_1, \dots, y_{i+k} the last vertex y_{i+k} is the same. We add an edge $v_{x_k} \rightarrow y_1$ only if y_1 is the first element of the chain of length $i + k$. If $\mathcal{A} \models R_i(x_1, \dots, x_i)$ then we add an edge $y_{i+k} \rightarrow b$ and if $\mathcal{A} \models \neg R_i(x_1, \dots, x_i)$ we add $y_{i+k} \rightarrow c$.

It is easy to see that any copy of $\mathcal{G}_{\mathcal{A}}$ computes a copy of \mathcal{A} and vice versa. Andrews and Miller [1] showed that $\mathcal{A} \equiv \mathcal{B}$ if and only if $\mathcal{G}_{\mathcal{A}} \equiv \mathcal{G}_{\mathcal{B}}$; we need to show that $\mathcal{A} \approx \mathcal{B}$ if and only if $\mathcal{G}_{\mathcal{A}} \approx \mathcal{G}_{\mathcal{B}}$. The proof makes use of the Tarski-Vaught test and Ehrenfeucht-Fraïssé games. Due to reasons of space we omit it. \square

A structure is *automorphically trivial* if there is a finite subset of its universe such that any permutation that fixes this subset pointwise is an automorphism. Knight [12] showed that the degree spectrum of automorphically trivial structures is a singleton and that otherwise it is upwards closed. The same holds for elementary b.e. spectra of structures. The key observation here is due to Fokina et al. [6] on the bi-embeddability type of automorphically trivial structures.

Proposition 3 ([6]). *Let \mathcal{A} be automorphically trivial and bi-embeddable with \mathcal{B} . Then $\mathcal{B} \cong \mathcal{A}$.*

Corollary 4. *The elementary bi-embeddability spectrum of a structure is either a singleton or upwards closed.*

Proof. It is a simple observation that the elementary b.e. spectrum of a structure \mathcal{A} is the union of isomorphism spectra of the structures elementary bi-embeddable with it, i.e.,

$$DgSp_{\cong}(\mathcal{A}) = \bigcup_{\mathcal{B} \cong \mathcal{A}} DgSp_{\cong}(\mathcal{B}).$$

Thus, if \mathcal{A} is automorphically trivial we get by Proposition 3 that its elementary b.e. spectrum is a singleton and that otherwise it is upwards closed. \square

Fokina et al. [7] showed the following.

Theorem 5 ([7]).

1. *There is a class \mathcal{F} of degrees such that for all $n \in \omega$ and all structures \mathcal{A} , $DgSp_{\equiv_n}(\mathcal{A}) \neq \mathcal{F}$.*
2. *There is a structure \mathcal{A} such that $DgSp_{\equiv}(\mathcal{A}) = DgSp_{\cong}(\mathcal{A}) = \mathcal{F}$.*

By the same argument as in Corollary 4 we get that there is a structure \mathcal{A} with $DgSp_{\cong}(\mathcal{A}) = \mathcal{F}$ and thus obtain an examples of an elementary bi-embeddability spectrum that is not a Σ_n spectrum for any $n \in \omega$.

Knight [12] showed that a set X is c.e. in all isomorphic copies of a structure \mathcal{A} if and only if it is enumeration reducible to the existential type of a tuple in \mathcal{A} . In fact this holds for elementary bi-embeddable copies.

Lemma 6. *Let $X \subseteq \omega$ and \mathcal{A} be a structure then the following are equivalent.*

1. *X is c.e. in every isomorphic copy of \mathcal{A} ,*
2. *X is e-reducible to $\exists\text{-tp}_{\mathcal{A}}(\bar{a})$ for some $\bar{a} \in A^{<\omega}$,*
3. *X is c.e. in every elementary bi-embeddable copy of \mathcal{A} .*

Proof. The equivalence of Items 1 and 2 was proven in [12]. To see the equivalence with Item 3 let \mathcal{B} be an elementary bi-embeddable copy of \mathcal{A} and $f : \mathcal{A} \rightarrow \mathcal{B}$ be elementary. Say X is e-reducible to $\exists\text{-tp}_{\mathcal{A}}(\bar{a})$, then by elementarity of f $\exists\text{-tp}_{\mathcal{A}}(\bar{a}) = \exists\text{-tp}_{\mathcal{B}}(f(\bar{a}))$ and thus X is e-reducible to $\exists\text{-tp}_{\mathcal{B}}(f(\bar{a}))$. Item 3 now follows from the equivalence between Items 1 and 2. \square

Theorem 7. *For $n > 1$ let $\mathbf{a}_1, \dots, \mathbf{a}_n$ be incomparable enumeration degrees. Then for any structure \mathcal{A} , $DgSp_{\cong}(\mathcal{A}) \neq \bigcup_{i < n} \{\mathbf{d} : \mathbf{d} \geq \mathbf{a}_i\}$.*

Corollary 8. *No elementary bi-embeddability spectrum is the union of finitely or countably many non degenerate cones of Turing degrees.*

Using Lemma 6 the proofs of the above Theorem and Corollary are similar to those for isomorphism spectra as presented by Montalbán [14, Theorem V.3.1]. We therefore omit them here.

Corollary 8 also holds for isomorphism spectra [18], spectra of atomic theories [1], and Σ_1 spectra [7]. On the other hand spectra of non-atomic theories and Σ_n spectra for $n > 1$ can be the union of two cones as was shown in [1], respectively, [7].

Given a set X , the *strong flower graph*¹ \mathcal{G}_X^s is the graph containing one vertex to which for every $x \in \omega$ a cycle of length $2x + 1$ is attached if $x \in X$ and a cycle of length $2x + 2$ if $x \notin X$. It is not hard to see that any copy of \mathcal{G}_X^s computes X and thus the isomorphism spectrum of \mathcal{G}_X^s is $\{deg(Y) : Y \geq_T X\}$. Furthermore any bi-embeddable copy \mathcal{A} of \mathcal{G}_X^s must be isomorphic to it as any cycle in \mathcal{G}_X^s must be mapped to a cycle of the same length in \mathcal{A} , all cycles must intersect in a single point and \mathcal{G}_X^s contains at most one cycle of length n for every $n \in \omega$. We thus obtain a similar result for elementary bi-embeddability spectra.

Proposition 9. *For every Turing degree \mathbf{d} , $\{\mathbf{e} : \mathbf{e} \geq \mathbf{d}\}$ is an elementary bi-embeddability spectrum.*

Definition 10. *Two sets A, B form a Σ_1 minimal pair if every set that is both A -c.e. and B -c.e. is c.e.*

Definition 11. *A structure \mathcal{A} has the c.e. extension property (ceep) if every existential type of a finite tuple of A is c.e.*

Proposition 12 ([1, Proposition 3.6]). *Let Y be any set and P be a non-empty Π_1^0 class. Then there is $X \in P$ such that X and Y form a Σ_1 -minimal pair.*

Proposition 13. *$DgSp_{\cong}(\mathcal{A})$ contains a Σ_1 minimal pair if and only if every \mathcal{B} elementary bi-embeddable with \mathcal{A} has the ceep.*

Proof. (\Rightarrow). Let $\mathbf{a}, \mathbf{b} \in DgSp_{\cong}(\mathcal{A})$ be a Σ_1 minimal pair and \mathcal{A} be \mathbf{a} -computable, \mathcal{B} be \mathbf{b} -computable. As the existential types realized in elementary b.e. copies coincide and existential types realized in a structure are computably enumerable from it, the existential types of \mathcal{A} and \mathcal{B} are c.e.

(\Leftarrow). This follows from the same Proposition for isomorphism spectra, i.e., it holds that given \mathcal{A} with the ceep then there exists a $\mathcal{B} \cong \mathcal{A}$ such that $deg(\mathcal{A})$ and $deg(\mathcal{B})$ form a Σ_1 minimal pair [1, Proposition 3.5]. □

¹ In the literature strong flower graphs are sometimes called flower graphs or daisy graphs. We use the term strong flower graph as we will use a similar construction only coding positive membership information of X below.

Proposition 14 ([1, Proposition 3.8]). *Let \mathcal{A} be a structure with the ceep, then there is an isomorphic copy of \mathcal{A} that does not compute a member of any special Π_1^0 class.*

A Π_1^0 class is *special* if it does not have a computable member. Putting Propositions 12 to 14 together we get that no elementary bi-embeddability spectrum is the upward closure of a special Π_1^0 class.

Theorem 15. *For all structures \mathcal{A} and special Π_1^0 classes P , $DgSp_{\cong}(\mathcal{A}) \neq \{deg(X) : \exists p \in P X \geq_T p\}$.*

The class of *diagonally noncomputable functions*, short *DNC*, form a special Π_1^0 class. Thus, their upward closure can not be an elementary bi-embeddability spectrum, and, furthermore, any elementary bi-embeddability spectrum contains a degree that does not compute a *DNC*. Combining this with the result by Jockusch and Soare [9], and Solovay [unpublished] that a degree computes a complete extension of Peano arithmetic (short, is a *PA degree*) if and only if it computes a two valued *DNC* function we obtain the following corollary.

Corollary 16. *The class of PA degrees is not the elementary bi-embeddability spectrum of a structure.*

In contrast to elementary bi-embeddability spectra, it was shown in [1] that there is a theory spectrum that consists of the PA degrees.

Slaman [16] and, independently, Wehner [19] showed that there is a structure whose isomorphism spectrum is all the non-computable degrees. Wehner used the following result.

Theorem 17 ([19]). *There is a family \mathfrak{F} of finite sets $(F_i)_{i \in \omega}$ such that (F_i) is uniformly *X-c.e.* for any non computable set *X* but not *c.e.**

To code this family in a structure we need a slightly weaker definition of a flower graph coding only positive information about membership in a set. Given a set *X* the *weak flower graph* \mathcal{G}_X^w is defined as the strong flower graph apart from the fact that if $x \notin X$ we do not attach a cycle to the central vertex. Then, for a family \mathfrak{F} the *bouquet graph* $\mathcal{G}_{\mathfrak{F}}^\infty$ is the disjoint union of infinitely many copies of the graphs \mathcal{G}_X^w for any $X \in \mathfrak{F}$. It is not hard to see that $\mathcal{G}_{\mathfrak{F}}^\infty$ has an *X*-computable copy if and only if \mathfrak{F} is *X-c.e.* Taking \mathfrak{F} as the family given by Wehner, one gets a graph whose degree spectrum is the set of noncomputable degrees.

The same strategy was later used by Kalimullin [10, 11] to show that for any *low* or *c.e.* degree **a** there is a structure having degree spectrum $\{\mathbf{x} : \mathbf{x} \not\leq \mathbf{a}\}$. This was later improved by Andrews et al. [1] who showed that for any degree **a** that is low over a *c.e.* degree **g** (a degree **a** is low over a degree **g** if $\mathbf{g} \leq \mathbf{a}$ and $\mathbf{a}' \leq \mathbf{g}'$) there is a structure having degree spectrum $\{\mathbf{x} : \mathbf{x} \not\leq \mathbf{a}\}$. As for flower graphs we have that if a structure is elementary bi-embeddable with a bouquet graph, then it is isomorphic to it. Thus, the above examples provide examples of elementary bi-embeddability spectra.

Proposition 18. *For every family \mathfrak{F} of finite sets and every graph $\mathcal{A} \cong \mathcal{G}_{\mathfrak{F}}^{\infty}$, $\mathcal{A} \cong \mathcal{G}_{\mathfrak{F}}^{\infty}$.*

Corollary 19. *Suppose that for \mathbf{a} there is c.e. $\mathbf{g} \leq \mathbf{a}$ such that $\mathbf{a}' \leq \mathbf{g}'$, then there is \mathcal{A} such that $DgSp_{\cong}(\mathcal{A}) = \{\mathbf{d} : \mathbf{d} \not\leq \mathbf{a}\}$.*

For Σ_1 spectra the above corollary does not hold. Fokina et al. [7] showed that the collection of non-computable Turing degrees is not a Σ_1 spectrum.

Proposition 20. *For every family \mathfrak{F} of finite sets there is a structure \mathcal{A} such that X computes an elementary bi-embeddable copy of \mathcal{A} if and only if X uniformly computes \mathfrak{F} .*

Csima and Kalimullin showed that there is a family of finite sets \mathfrak{F} such that \mathfrak{F} is X -computable if and only if $deg(X)$ is hyperimmune. Diamondstone et al. [3] constructed a family \mathfrak{F} of sets such that \mathfrak{F} is X -computable if and only if the degree of X is array non-computable, and they showed that the degrees who uniformly enumerate this family are exactly the non-jump traceable degrees. Using Propositions 20 and 18 we get that these collections are elementary bi-embeddability spectra of structures.

Corollary 21. *The hyperimmune degrees, the array non-computable degrees and the non-jump traceable degrees are all elementary bi-embeddability spectra of structures.*

3 Towards Jump Inversion for Elementary Bi-embeddability Spectra

Goncharov et al. [8] showed that if \mathcal{F} is the isomorphism spectrum of a structure, then so is $\{\mathbf{d} : \mathbf{d}^{(\alpha)} \in \mathcal{F}\}$ for successor ordinals $\alpha < \omega_1^{CK}$. This result can be seen as an analogue of the classical jump inversion results in computability theory. Andrews and Miller [1] showed that $\{\mathbf{d} : \mathbf{d}^{(\omega+1)} \geq_T \mathbf{0}^{(\omega \cdot 2 + 2)}\}$ is not the spectrum of a theory but by the above it is the isomorphism spectrum of a structure. Thus, in general we can not do transfinite jump inversion for theory spectra. In this section we obtain some positive examples for jump inversion of elementary bi-embeddability spectra, and, among other things give an example of an elementary bi-embeddability spectrum that is not the spectrum of a theory.

To obtain the results in this section we “invert” graphs. Given a graph \mathcal{G} we create a structure $\mathcal{G}^{-\alpha}$ by replacing every edge in \mathcal{G} by a copy of a structure $\mathcal{S}_{\alpha,0}$ and associating a structure $\mathcal{S}_{\alpha,1}$ with every pair of non-adjacent vertices. These two structures have the property that it is Δ_{α}^0 -complete to check whether a structure is a copy of $\mathcal{S}_{\alpha,0}$ or $\mathcal{S}_{\alpha,1}$. We then get that

$$DgSp_{\cong}(\mathcal{G}^{-\alpha}) = \{\mathbf{d} : \mathbf{d}^{(\alpha)} \in DgSp_{\cong}(\mathcal{G})\}.$$

Formally, $\mathcal{G}^{-\alpha}$ is in the language consisting of relation symbols $V/1, R/3$ union the language of $\mathcal{S}_{\alpha,0}, \mathcal{S}_{\alpha,1}$. The relation V is true of elements representing

the vertices of \mathcal{G} and R partitions the remaining elements into infinitely many infinite sets where for $a, b \in V$, $R(a, b, -) \cong \mathcal{S}_{\alpha,0}$ if a and b are adjacent in \mathcal{G} , and $R(a, b, -) \cong \mathcal{S}_{\alpha,1}$ otherwise.

In the following proofs we use pairs of linear orderings for $\mathcal{S}_{\alpha,0}$ and $\mathcal{S}_{\alpha,1}$. Formally a pair of linear orderings (L_1, L_2) is in the language $(T/1, \leq/2)$ where \leq restricted to T is isomorphic to L_1 and \leq restricted to $\neg T$ is isomorphic to L_2 . If we do jump inversion for even ordinals, i.e., ordinals of the form $2\alpha + 2$ we let

$$S_{2\alpha+2,0} \cong (\omega^{\alpha+1} + \omega^\alpha, \omega^{\alpha+1}) \text{ and } S_{2\alpha+2,1} \cong (\omega^{\alpha+1}, \omega^{\alpha+1} + \omega^\alpha).$$

And for jump inversion of odd ordinals, i.e., of the form $2\alpha + 1$ we use

$$S_{2\alpha+1,0} \cong (\omega^\alpha \cdot 2, \omega^\alpha) \text{ and } S_{2\alpha+1,1} \cong (\omega^\alpha, \omega^\alpha \cdot 2).$$

To get that for every copy \mathcal{H} of $\mathcal{G}^{-\alpha}$, $\mathcal{H}^{(\alpha)}$ computes a copy of \mathcal{G} we need the following Lemma.

- Lemma 22.** 1. It is $\Delta_{2\alpha+2}^0$ -complete to check whether $L \cong (\omega^{\alpha+1} + \omega^\alpha, \omega^{\alpha+1})$ or $L \cong (\omega^{\alpha+1}, \omega^{\alpha+1} + \omega^\alpha)$.
 2. It is $\Delta_{2\alpha+1}^0$ -complete to check whether $L \cong (\omega^\alpha \cdot 2, \omega^\alpha)$ or $L \cong (\omega^\alpha, \omega^\alpha \cdot 2)$.

To prove the Lemma we define an equivalence relation \sim_α on linear orderings by transfinite recursion on α :

Definition 23. Let \mathcal{L} be a linear ordering and $x, y \in L$. Then let

1. $x \sim_0 y$ if $x = y$,
2. $x \sim_1 y$ if $[x, y]$ or $[y, x]$ is finite,
3. for $\alpha = \beta + 1$, $x \sim_\alpha y$ if in \mathcal{L}/\sim_β , $[x]_{\sim_\beta} \sim_1 [y]_{\sim_\beta}$,
4. for $\alpha = \lim \beta$, $x \sim_\alpha y$ if for some $\beta < \alpha$, $x \sim_\beta y$.

The relation \sim_1 is commonly known as the block relation.

Proof (Lemma 22). The α block relation \sim_α for $\alpha = \beta + 1$ is definable by the following computable $\Sigma_{2\alpha}$ formula.

$$x \sim_\alpha y \Leftrightarrow \bigvee_{n \in \omega} \forall y_1, \dots, y_n \left(x < y_1 < \dots < y_n < y \rightarrow \bigvee_{1 \leq i < j \leq n} y_i \sim_\beta y_j \right).$$

For λ a limit ordinal the defining formula is the disjunction of all formulas defining \sim_β for $\beta < \lambda$. This is then clearly Σ_λ and $2\lambda = \lambda$. Goncharov et al. [8] proved the following.

Claim ([8, Lemma 5.1]). Let α be a computable successor ordinal and $\mathcal{A}_1, \mathcal{A}_2$ such that

1. $\{\mathcal{A}_1, \mathcal{A}_2\}$ is α -friendly,
2. $\mathcal{A}_1, \mathcal{A}_2$ satisfy the same infinitary $\Pi_{2\beta}$ formulas for $\beta < \alpha$,

3. and each $\mathcal{A}_i, i \in \{1, 2\}$, satisfies a computable $\Pi_{2\alpha}$ sentence not satisfied by the other,

Then for any Δ_α^0 set S there is a uniformly computable sequence $(C_i)_{i \in \omega}$ such that

$$C_i \cong \begin{cases} \mathcal{A}_1 & \text{if } i \in S, \\ \mathcal{A}_2 & \text{otherwise.} \end{cases}$$

Fix α . That the pairs $\mathcal{S}_{2\alpha+1,0}$ and $\mathcal{S}_{2\alpha+1,1}$, respectively, $\mathcal{S}_{2\alpha+2,0}$ and $\mathcal{S}_{2\alpha+2,1}$ satisfy (1) and (2) follows from [2, Lemma 15.10]. To see that (3) is satisfied consider the following facts.

1. $(\omega^\alpha 2, \omega^\alpha) \models \forall x, y ((\neg T(x) \wedge \neg T(y)) \rightarrow x \sim_\alpha y)$ which is $\Pi_{2\alpha+1}$,
2. $(\omega^\alpha, \omega^\alpha 2) \models \forall x, y ((T(x) \wedge T(y)) \rightarrow x \sim_\alpha y)$ which is $\Pi_{2\alpha+1}$,
3. $(\omega^{\alpha+1} + \omega^\alpha, \omega^{\alpha+1}) \models \forall x (\neg T(x) \rightarrow (\exists y (y > x \wedge \neg T(y) \wedge y \not\sim_\alpha x)))$ which is $\Pi_{2\alpha+2}$,
4. $(\omega^{\alpha+1}, \omega^{\alpha+1} + \omega^\alpha) \models \forall x (T(x) \rightarrow (\exists y (y > x \wedge T(y) \wedge y \not\sim_\alpha x)))$ which is $\Pi_{2\alpha+2}$.

Neither of the above sentences satisfied by one of the structures is satisfied by its partner. Hence, the conditions in the Claim are satisfied and the Lemma follows. □

Definition 24. A degree \mathbf{d} is non-low $_\alpha$ if $\mathbf{d}^{(\alpha)} > \mathbf{0}^{(\alpha)}$.

Theorem 25. For every $n < \omega$ the non-low $_n$ degrees, $\{\mathbf{d} : \mathbf{d}^{(n)} > \mathbf{0}^{(n)}\}$, are the elementary bi-embeddability spectrum of a structure.

Proof. Let \mathcal{G} be the bouquet graph of the Wehner family relativized to $\emptyset^{(n)}$, then $DgSp_{\cong}(G) = \{\mathbf{d} : \mathbf{d} > \mathbf{0}^{(n)}\}$. To obtain the inverted graph \mathcal{G}^{-n} we use the construction described above with $\mathcal{S}_{n,0}$ and $\mathcal{S}_{n,1}$ as our structures. We get from Lemma 22 that $DgSp_{\cong}(\mathcal{G}^{-n}) = \{\mathbf{d} : \mathbf{d}^{(n)} > \mathbf{0}^{(n)}\}$.

By Proposition 18, $DgSp_{\cong}(\mathcal{G}) = DgSp_{\cong}(\mathcal{G}^{-n})$. We show that $DgSp_{\cong}(\mathcal{G}^{-n}) = DgSp_{\cong}(\mathcal{G}^{-n})$. The proof relies on the fact that for ordinals $\alpha, \beta < \omega^\omega, \alpha \preceq \beta$ if and only if $\alpha \cong \beta$ [4]. Let $\mathcal{H} \cong \mathcal{G}^{-n}$ and $\mu : \mathcal{H} \rightarrow \mathcal{G}^{-n}$ be an elementary embedding. Clearly R holds only on triples with elements in the first and second column satisfying V and by the above mentioned fact for all $a, b \in H$ such that $V(a)$ and $V(b)$, $R(a, b, -) \cong R(\mu(a), \mu(b), -)$. Thus we can construct a graph \mathcal{H}^{+n} from \mathcal{H} such that $\mathcal{H}^{+n} \cong \mathcal{G}$ and hence, $\mathcal{H}^{+n} \cong \mathcal{G}$. But this implies that $\mathcal{H} \cong \mathcal{G}^{-n}$ and thus $deg(\mathcal{H}) \in DgSp_{\cong}(\mathcal{G}^{-n})$. □

Theorem 26. For all successor ordinals $\alpha, \beta < \omega_1^{CK}, \{\mathbf{d} : \mathbf{d}^{(\alpha)} \geq_T \mathbf{0}^{(\beta)}\}$ is the elementary bi-embeddability spectrum of a structure.

Proof. We start with the strong flower graph \mathcal{G} coding $\emptyset^{(\beta)}$ and produce a structure \mathcal{G}_e in the language $E/2, P/2, S/1$. We interpret E as the edge relation in \mathcal{G}, P as the successor relation on ω and let S hold of the single vertex which is the first element in P . It is easy to see that \mathcal{G}_e computes $\emptyset^{(\beta)}$ and that from $\emptyset^{(\beta)}$ we can compute a copy of \mathcal{G}_e .

Now we obtain a structure $\mathcal{G}^{-\alpha}$ by inverting \mathcal{G} . We get the structure $\mathcal{G}_e^{-\alpha}$ by adding relations $P/2, S/1$ and interpreting them so that the canonical bijection between \mathcal{G}_e and the elements for which V holds in $\mathcal{G}_e^{-\alpha}$ is structure preserving on P and S . Let this structure be $\mathcal{G}_e^{-\alpha}$. By Lemma 22 $DgSp_{\cong}(\mathcal{G}_e^{-\alpha})$ is the desired spectrum and clearly for $\mathcal{G}_e, \mathcal{H} \cong \mathcal{G}_e$ if and only if $\mathcal{H} \cong \mathcal{G}_e$.

We show that $\mathcal{H} \cong \mathcal{G}_e^{-\alpha}$ if and only if $\mathcal{H} \cong \mathcal{G}_e^{-\alpha}$. Let $\mathcal{H} \cong \mathcal{G}_e^{-\alpha}$ and $\mu : \mathcal{H} \rightarrow \mathcal{G}^{-n}, \nu : \mathcal{G}^{-n} \rightarrow \mathcal{H}$ be elementary embeddings. Then, as in the proof of Theorem 25, R holds only on triples with elements in the first and second column satisfying V . Let a be the single element in H such that $\mathcal{H} \models S(a)$. Then $\nu(\mu(a)) = a$ and by induction we get the same for any $u \in H$ such that $V(u)$. Thus we get that for all $a, b \in H$ satisfying V that $R(a, b, -) \cong R(\mu(a), \mu(b), -)$. Hence, we can construct a structure $\mathcal{H}^{+\alpha}$ from \mathcal{H} such that $\mathcal{H}^{+\alpha} \cong \mathcal{G}_e$, and therefore $\mathcal{H}^{+\alpha} \cong \mathcal{G}_e$. This implies that $\mathcal{H} \cong \mathcal{G}_e^{-\alpha}$. \square

Notice that in the proof of Theorem 26 we never used the fact that our embeddings are elementary, therefore the analogue of this theorem also holds for bi-embeddability spectra.

Corollary 27. *For all successor ordinals $\alpha, \beta < \omega_1^{\text{CK}}, \{\mathbf{d} : \mathbf{d}^{(\alpha)} \geq_T \mathbf{0}^{(\beta)}\}$ is the bi-embeddability spectrum of a structure.*

Andrews and Miller [1] showed that $\{\mathbf{d} : \mathbf{d}^{(\omega+1)} \geq \mathbf{0}^{(\omega \cdot 2 + 2)}\}$ is not the spectrum of a theory but by the above it is both a bi-embeddability spectrum and an elementary bi-embeddability spectrum. We have thus found an example of an elementary bi-embeddability spectrum that is not the spectrum of a theory.

Corollary 8 and Theorem 26 show that there are theory spectra that are not elementary bi-embeddability spectra and vice versa. From Theorem 5 we get an elementary bi-embeddability spectrum that is not a Σ_n spectrum for any n and Corollary 8 again shows that there are Σ_n spectra for $n > 1$ that are not elementary bi-embeddability spectra. Whether the same holds for Σ_1 spectra and the relationship between elementary bi-embeddability spectra and isomorphism spectra is still unknown.

Question 28. Is every isomorphism spectrum the elementary bi-embeddability spectrum of a structure and vice versa?

References

1. Andrews, U., et al.: The complements of lower cones of degrees and the degree spectra of structures. *J. Symb. Log.* **81**(3), 997–1006 (2016). <https://doi.org/10.1017/jsl.2015.59>
2. Ash, C., Knight, J.: *Computable Structures and the Hyperarithmetical Hierarchy*, vol. 144. Newnes, Amsterdam (2000). [https://doi.org/10.1016/s0049-237x\(00\)80006-3](https://doi.org/10.1016/s0049-237x(00)80006-3)
3. Diamondstone, D., Greenberg, N., Turetsky, D.: Natural large degree spectra. *Computability* **2**(1), 1–8 (2013). <https://doi.org/10.3233/COM-13008>

4. Doner, J.E. Mostowski, A., Tarski, A.: The elementary theory of well-ordering—a metamathematical study. *Studies in Logic and the Foundations of Mathematics*, vol. 96, pp. 1–54. Elsevier, Amsterdam (1978). [https://doi.org/10.1016/s0049-237x\(08\)71988-8](https://doi.org/10.1016/s0049-237x(08)71988-8)
5. Fokina, E., Harizanov, V., Melnikov, A.: Computable model theory. In: *Turing’s Legacy: Developments from Turing’s Ideas in Logic*, pp. 124–191. Cambridge University Press, Cambridge (2014). <https://doi.org/10.1017/cbo9781107338579.001>
6. Fokina, E., Rossegger, D. Mauro, L.S.: Bi-embeddability spectra and bases of spectra (2018, preprint)
7. Fokina, E., Semukhin, P. Turetsky, D.: Degree spectra of structures under equivalence relations. *Algebra Log.* (2018, to appear)
8. Goncharov, S., et al.: Enumerations in computable structure theory. *Ann. Pure Appl. Log.* **136**(3), 219–246 (2005). <https://doi.org/10.1016/j.apal.2005.02.001>
9. Jockush, C., Soare, R.: Degrees of members of Π_1^0 classes. *Pac. J. Math.* **40**(3), 605–616 (1972). <https://doi.org/10.2140/pjm.1972.40.605>
10. Kalimullin, I.: Almost computably enumerable families of sets. *Sb.: Math.* **199**(10), 1451 (2008). <https://doi.org/10.1070/sm2008v199n10abeh003967>
11. Kalimullin, I.: Spectra of degrees of some structures. *Algebra Log.* **46**(6), 729–744 (2007). <https://doi.org/10.1007/s10469-007-0039-6>
12. Knight, J.F.: Degrees coded in jumps of orderings. *J. Symb. Log.* **51**(04), 1034–1042 (1986). <https://doi.org/10.2307/2273915>
13. Marker, D.: *Model Theory: An Introduction*. Springer, New York (2002). <https://doi.org/10.1007/b98860>
14. Montalbán, A.: Computable structure theory, Draft, 20 January 2018
15. Richter, L.J.: Degrees of structures. *J. Symb. Log.* **46**(04), 723–731 (1981). <https://doi.org/10.2307/2273222>
16. Slaman, T.: Relative to any nonrecursive set. *Proc. Am. Math. Soc.* **126**(7), 2117–2122 (1998). <https://doi.org/10.1090/S0002-9939-98-04307-X>
17. Soare, R.I.: *Turing Computability. Theory and Applications of Computability*. Springer, Berlin (2016). <https://doi.org/10.1007/978-3-642-31933-4>
18. Soskov, I.N.: Degree spectra and co-spectra of structures. *Ann. Univ. Sofia* **96**, 45–68 (2004)
19. Wehner, S.: Enumerations, countable structures and turing degrees. *Proc. Am. Math. Soc.* **126**(7), 2131–2139 (1998). <https://doi.org/10.1090/S0002-9939-98-04314-7>



A Generic m -Reducibility

Alexander Rybalov^(✉)

Sobolev Institute of Mathematics, Pevtsova 13, Omsk 644099, Russia
alexander.rybalov@gmail.com

Abstract. Kapovich, Myasnikov, Schupp and Shpilrain in 2003 developed generic approach to algorithmic problems, which considers an algorithmic problem on “most” of the inputs (i.e., on a generic set) instead of the entire domain and ignores it on the rest of inputs (a negligible set). Jockusch and Schupp in 2012 began the study of generic computability in the context of classical computability theory. In particular, they defined a generic analog of Turing reducibility. In this paper we introduce a generic analog of m -reducibility as m -reducibility by computable functions, which preserve the non-negligibility of sets. We study generic m -reducibility of computable and c.e. sets. We prove the existence of generically m -complete c.e. sets, incomparable c.e. sets and m -degrees, which contain more than one generic m -degree.

1 Introduction

Kapovich et al. in [6] developed generic approach to algorithmic problems, which considers an algorithmic problem on “most” of the inputs (i.e., on a generic set) instead of the entire domain and ignores it on the rest of inputs (a negligible set). It turned out, that many famous undecidable problems are easily decidable on most inputs – on the so-called “generic” sets of inputs. For example, it was proved for the halting problem for Turing machines with a one-way infinite tape [5], for the word problem of many finitely presented groups with a word problem that is undecidable in the classical sense [6]. But there are problems, which remain undecidable even in the generic case [7].

Jockusch and Schupp in [4] began the study of generic computability in the context of classical computability theory. In particular, they defined a generic analog of Turing reducibility and found a deep connection between classical and generic Turing degrees. However, there are some open questions about the structure of generic Turing degrees. For example, it is not yet known whether or not there are minimal degrees or minimal pairs in the generic degrees (see [1–3]).

In this paper we introduce a generic analog of m -reducibility as m -reducibility by computable functions, which preserve the non-negligibility of sets. We study generic m -reducibility of computable and c.e. sets. We prove the existence of

A. Rybalov—Supported by the program of basic scientific researches SB RAS II.1, project 0314-2016-0004.

generically m -complete c.e. sets, incomparable c.e. sets and m -degrees, which contain more than one generic m -degree.

Now we give the basic definitions of generic-case computability from [4]. For a set $S \subseteq \omega$ define the following sequence

$$\rho_{n(S)} = \frac{|\{x : x \leq n, x \in S\}|}{n}, n = 1, 2, 3, \dots$$

The *asymptotic density* of $S \subseteq \omega$ is the following limit (if it exists)

$$\rho(S) = \lim_{n \rightarrow \infty} \rho_{n(S)}.$$

A set $S \subseteq \omega$ is called *generic* if $\rho(S) = 1$ and *negligible* if $\rho(S) = 0$. Clearly, S is generic if and only if its complement \bar{S} is negligible.

Let S be a subset of ω with characteristic function χ_S . A partial function $\varphi : \omega \rightarrow \{0, 1\}$ is called a *generic description* of S if $\varphi(x) = \chi_{S(x)}$ whenever $\varphi(x)$ is defined and the domain of φ is generic. A set $S \subseteq \omega$ is called *generically computable* if there exists a partial computable function φ , which is a generic description of S . Otherwise S is called *generically undecidable*.

2 Generic m-Reducibility

Definition 1. A set $A \subseteq \omega$ is *generically m -reducible* to $B \subseteq \omega$ (written $A \leq_{gm} B$) if there is a computable function $f : \omega \rightarrow \omega$ such that

1. $\forall x \in \omega \ x \in A \Leftrightarrow f(x) \in B$,
2. $\forall S \subseteq \omega \ S \text{ is not negligible} \Rightarrow f(S) \text{ is not negligible}$.

As usual, $A \equiv_{gm} B$ means that $A \leq_{gm} B$ and $B \leq_{gm} A$. Also we write $A <_{gm} B$, if $A \leq_{gm} B$ and $B \not\leq_{gm} A$. Define the gm -degree of a set A as $d_{gm}(A) = \{B \subseteq \omega : B \equiv_{gm} A\}$.

Theorem 1. For any sets $A, B, C \subseteq \omega$ it holds

1. $A \leq_{gm} A$.
2. $A \leq_{gm} B \Rightarrow A \leq_m B$.
3. $A \leq_{gm} B, B \leq_{gm} C \Rightarrow A \leq_{gm} C$.
4. If $A \leq_{gm} B$ and B is generically computable, then A is generically computable.

Proof. (1) and (2) are obvious.

To prove (3) suppose that $A \leq_{gm} B$ by a computable function f and $B \leq_{gm} C$ by a computable function g . Then $A \leq_{gm} C$ by $g(f)$, since for any $S \subseteq \omega$ the set S is not negligible $\Rightarrow f(S)$ is not negligible $\Rightarrow g(f(S))$ is not negligible.

To prove (4) suppose that $A \leq_{gm} B$ by a computable function f . Since B is generically computable, then there is a generic description φ of B . Now function $\varphi(f)$ is a generic description of set A . Indeed, assume for a contradiction, that domain of $\varphi(f)$ is not generic. That means that $C = \{x \in \omega : f(x) \notin \text{dom}(\varphi)\}$ is not negligible. But $f(C) \subseteq \omega \setminus \text{dom}(\varphi)$ is negligible, a contradiction. \square

The following Lemma describes a particular case when 1-reducibility is *gm*-reducibility.

Lemma 1. *Let $A \leq_1 B$ by a computable monotonically increasing function $f(x)$ such that there exists a constant $C > 0$ such that $f(x) < Cx$ for all $x > 0$. Then $A \leq_{gm} B$.*

Proof. Let $S \subseteq \omega$ be a non-negligible set. That means that there is a constant $\varepsilon > 0$ such that for any integer N there exists an $n > N$ such that

$$\rho_{n(S)} = \frac{|\{k : k \leq n, k \in S\}|}{n} > \varepsilon.$$

We need to prove that $f(S)$ is not negligible. Since f is monotonically increasing, we have

$$\begin{aligned} \rho_{f(n)}(f(S)) &= \frac{|\{f(k) : f(k) \leq f(n), k \in S\}|}{f(n)} = \\ &= \frac{|\{k : k \leq n, k \in S\}|}{f(n)} = \rho_{n(S)} \frac{n}{f(n)}. \end{aligned}$$

But $\frac{n}{f(n)} > \frac{1}{C}$, so $\rho_{f(n)}(f(S)) > \frac{\varepsilon}{C}$. It implies that $f(S)$ is not negligible. □

3 Generic m-Reducibility of Computable Sets

Theorem 2. *Let $A, B \subseteq \omega$ be computable sets, $A, B \neq \emptyset, \omega$, and there exist $\rho(A), \rho(B)$. Then it holds.*

1. *If $\rho(A) = \rho(B) = 1$, then $A \equiv_{gm} B$.*
2. *If $\rho(A) = \rho(B) = 0$, then $A \equiv_{gm} B$.*
3. *If $\rho(A), \rho(B) \neq 0$ and $\rho(A), \rho(B) \neq 1$, then $A \equiv_{gm} B$.*

Proof. (1) It is easy to see that $A \leq_{gm} B$ by the following function

$$f(x) = \begin{cases} x, & \text{if } x \in A \text{ and } x \in B, \\ x, & \text{if } x \notin A \text{ and } x \notin B, \\ b, & \text{if } x \in A \text{ and } x \notin B, \\ a, & \text{if } x \notin A \text{ and } x \in B, \end{cases}$$

where $a \in A, a \notin B$ and $b \in B$. Similarly $B \leq_{gm} A$.

Case (2) follows from (1).

Now consider case (3). Let $A = \{a_{1(i)} : i \in \omega\}, \bar{A} = \{a_{2(i)} : i \in \omega\}, B = \{b_{1(i)} : i \in \omega\}, \bar{B} = \{b_{2(i)} : i \in \omega\}$ be effective enumerations of sets in increasing order. Since A and B are computable, not finite and not co-finite, so $A \leq_1 B$ by the computable function f , which maps $a_{1(i)}$ to $b_{1(i)}$ and $a_{2(i)}$ to

$b_{2(i)}$. Since $\rho(A) > 0$, there exist constants $C_1 > C_2 > 0$ such that for large enough n

$$C_2 < \rho_{a_1(n)}(A) = \frac{|\{a_1(k) : a_1(k) \leq a_1(n)\}|}{a_1(n)} = \frac{n}{a_1(n)} < C_1.$$

Whence $n < C_1 a_1(n)$. Similarly, since $\rho(B) > 0$, we can bound $b_{1(n)} < C_3 n$ for some constant $C_3 > 0$. It follows from these two bounds that $b_{1(n)} < C_4 a_1(n)$ for some constant $C_4 > 0$. In the same manner we can bound $b_{2(n)} < C_5 a_2(n)$ with a constant $C_5 > 0$. Now Lemma 1 implies that $A \leq_{gm} B$. Analogously, $B \leq_{gm} A$. □

The following theorem states that there are incomparable computable sets with respect to the order \leq_{gm} , contained in a 1-degree.

Theorem 3. *There are computable sets A and B such that $A \equiv_1 B$, but $A \not\leq_{gm} B$ and $B \not\leq_{gm} A$.*

Proof. Let A be any computable negligible infinite set, and B – any computable generic set, which is not co-finite. Clearly, $A \equiv_1 B$. But if $A \leq_{gm} B$ by some computable function f , then f maps the generic set \bar{A} to the negligible set \bar{B} – a contradiction with the definition of gm -reducibility. If $B \leq_{gm} A$ by some computable function g , then g maps generic set B to the negligible set A , a contradiction. □

4 Generic m-Reducibility of C.E. Sets

As for any classical reducibility notion, a c.e. set S is called gm -complete, if $A \leq_{gm} S$ for every c.e. set A .

Theorem 4. *There is a gm -complete c.e. set.*

Proof. Let $W_0, W_1, \dots, W_k, \dots$ be an effective enumeration of all c.e. sets. Consider the following c.e. set

$$C = \{2^k(2m + 1) : m \in W_k, k = 0, 1, \dots\}.$$

Now Lemma 1 implies that $W_k \leq_{gm} C$ by the function $f_{k(x)} = 2^k(2x + 1)$. □

To construct incomparable generically undecidable c.e. sets we need more complicated arguments than in the proof of Theorem 3, because any generic c.e. set is generically computable ([4], Corollary 2.3). Recall that a subset $I \subseteq \omega$ is called *immune* if I is infinite and does not contain any infinite c.e. subset. A set $S \subseteq \omega$ is called *simple* if S c.e. and \bar{S} is immune.

Lemma 2. *Any simple non-generic set is generically undecidable.*

Proof. Let S be a simple non-generic set. Suppose S is generically computable and φ is its generic description. Then the set $\{x : \varphi(x) = 0\}$ is a not negligible (and therefore infinite) subset of immune set \bar{S} . A contradiction. □

Theorem 5. *There are generically undecidable c.e. sets $A, B \subseteq \omega$ such that $A \not\leq_{gm} B$ and $B \not\leq_{gm} A$.*

Proof. Let S be a negligible simple set. The existence of such sets was established in [4] (see the proof of Proposition 2.15). By Lemma 2 S is not generically computable. Consider the following sets:

$$A = S \cup \{2n : n \in \omega\}$$

and

$$B = 2S = \{2x : x \in S\}.$$

It is clear that the set A is simple, not negligible and not generic. By Lemma 2 A is not generically computable. Lemma 1 implies that $S \leq_{gm} B$ by the function $f(x) = 2x$. Therefore the set B cannot be generically computable. Now we will prove that the sets A and B are incomparable with respect to the order \leq_{gm} .

Suppose $A \leq_{gm} B$ by some computable function f . Then $f(A) \subseteq B$. But B is negligible, since S is negligible. So $f(A)$ is negligible. But A is not negligible — a contradiction with the definition of generic m -reducibility.

Conversely, suppose $B \leq_{gm} A$ by some computable function g . Note, that non-negligible c.e. set $C = \{2k + 1 : k \in \omega\}$ is a subset of \overline{B} . Therefore c.e. set $f(C)$ is a subset of \overline{A} . But \overline{A} is immune, so $f(C)$ is a finite and, certainly, negligible set. Again we have a contradiction. \square

The following theorem states that there are c.e. m -degrees, which contain more than one c.e. gm -degrees.

Theorem 6. *There are generically undecidable c.e. sets A and B such that $A \equiv_m B$, but $A <_{gm} B$.*

Proof. Let S be a negligible simple set. Consider the sets $A = S$ and $B = 2S = \{2x : x \in S\}$. By Lemma 2 the set A is generically undecidable. $A \leq_{gm} B$ by the function $f(x) = 2x$. Therefore the set B is generically undecidable too. Moreover $B \leq_m A$ by the following function

$$f(x) = \begin{cases} x/2, & \text{if } x \text{ is even,} \\ c, & \text{if } x \text{ is odd,} \end{cases}$$

where $c \notin S$. But $B \not\leq_{gm} A$ because non-negligible c.e. set $\{2k + 1 : k \in \omega\}$ is contained in \overline{B} and immune set \overline{A} does not contain any infinite c.e. subsets (see the proof of Theorem 5). Therefore $A <_{gm} B$. \square

The author would like to thank the anonymous reviewers for their remarks and suggestions to improve the quality of the paper.

References

1. Cholak, P., Igusa, G.: Density-1-bounding and quasiminimality in the generic degrees. *J. Symb. Log.* **82**(3), 931–957 (2017)
2. Igusa, G.: Nonexistence of minimal pairs for generic computability. *J. Symbol. Log.* **78**(2), 511–522 (2013)
3. Igusa, G.: The generic degrees of density-1 sets and a characterization of the hyperarithmetic reals. *J. Symbol. Log.* **80**, 1290–1314 (2015)
4. Jockusch, C., Schupp, P.: Generic computability, turing degrees, and asymptotic density. *J. Lond. Math. Soc.* **85**(2), 472–490 (2012)
5. Hamkins, J.D., Miasnikov, A.: The halting problem is decidable on a set of asymptotic probability one. *Notre Dame J. Form. Log.* **47**(4), 515–524 (2006)
6. Kapovich, I., Myasnikov, A., Schupp, P., Shpilrain, V.: Generic-case complexity, decision problems in group theory and random walks. *J. Algebra* **264**(2), 665–694 (2003)
7. Myasnikov, A., Rybalov, A.: Generic complexity of undecidable problems. *J. Symb. Log.* **73**(2), 656–673 (2008)



Some Nonstandard Equivalences in Reverse Mathematics

Sam Sanders^(✉)

Center for Advanced Studies and Munich Center for Mathematical Philosophy,
LMU Munich, Munich, Germany
sasander@me.com

Abstract. Reverse Mathematics (RM) is a program in the foundations of mathematics founded by Friedman and developed extensively by Simpson. The aim of RM is finding the minimal axioms needed to prove a theorem of ordinary (i.e. non-set theoretical) mathematics. In the majority of cases, one also obtains an *equivalence* between the theorem and its minimal axioms. This equivalence is established in a weak logical system called *the base theory*; four prominent axioms which boast lots of such equivalences are dubbed *mathematically natural* by Simpson. In this paper, we show that a number of axioms from Nonstandard Analysis are equivalent to theorems of ordinary mathematics *not involving Nonstandard Analysis*. These equivalences are proved in a base theory recently introduced by van den Berg and the author. Our results combined with Simpson's criterion for naturalness suggest the controversial point that Nonstandard Analysis is actually mathematically natural.

1 Introduction

Reverse Mathematics (RM) is a program in the foundations of mathematics founded by Friedman ([5]) and developed extensively by Simpson ([22]) and others. We refer to the latter for an overview of RM and will assume basic familiarity, in particular with the *Big Five* systems of RM. The latter are (still) claimed to capture the majority of theorems of ordinary (i.e. non-set theoretical) mathematics ([12, p. 495]). Our starting point is the following quote by Simpson on the ‘mathematical naturalness’ of logical systems from [22, I.12]:

From the above it is clear that the [Big Five] five basic systems RCA_0 , WKL_0 , ACA_0 , ATR_0 , $II_1^1\text{-}CA_0$ arise naturally from investigations of the Main Question. The proof that these systems are mathematically natural is provided by Reverse Mathematics.

In a nutshell, according to Simpson, the many equivalences in RM, proved over RCA_0 and involving the other four Big Five, imply that the Big Five systems are mathematically natural. In this paper, we show that a number of axioms

S. Sanders—This research was supported by the Alexander von Humboldt Foundation and LMU Munich (via the Excellence Initiative).

from *Nonstandard Analysis* (NSA) are equivalent to theorems of ordinary mathematics *not involving NSA*. These results combined with Simpson’s criterion for naturalness suggest the controversial point that NSA is actually mathematically natural. Indeed, both Alain Connes and Errett Bishop have expressed extremely negative (but unfounded; see [19]) opinions of NSA, in particular its naturalness.

Finally, the aforementioned equivalences are proved in a (weak) base theory recently introduced by van den Berg and the author in [3]. We sketch the main properties of this base theory in Sect. 2 and prove our main results in Sect. 3. The latter include an equivalence between the *Heine-Borel compactness* (for **any** open cover) of the unit interval and the *nonstandard compactness* of Cantor space. We obtain similar results based on WWKL, a weakening of WKL.

2 A Base Theory from Nonstandard Analysis

We introduce the system B_0 from [3]. This system is a Π_2^0 -conservative extension of EFA (aka $I\Delta_0 + \text{EXP}$) enriched with all finite types and fragments of Nelson’s axioms of *internal set theory* ([13]), a well-known axiomatic approach to NSA.

Let $E\text{-EFA}^\omega$ be EFA enriched with all finite types, i.e. Kohlenbach’s system $E\text{-G}_3A^\omega$ ([9, p. 55]). The language of B_0 is obtained from that of $E\text{-EFA}^\omega$ by adding unary predicates ‘ st^σ ’ for any finite type σ . Formulas in the old language of $E\text{-EFA}^\omega$, i.e. those not containing these new symbols, are *internal*; By contrast, general formulas of B_0 are *external*. The new ‘ st ’ predicates give rise to two new quantifiers as in (2.1), and we omit type superscripts whenever possible.

$$(\forall^{\text{st}}x)\Phi(x) \equiv (\forall x)(\text{st}(x) \rightarrow \Phi(x)) \text{ and } (\exists^{\text{st}}x)\Phi(x) \equiv (\exists x)(\text{st}(x) \wedge \Phi(x)). \quad (2.1)$$

The system B_0 is $E\text{-EFA}^\omega + \text{QF-AC}^{1,0}$, plus the basic axioms as in Definition 2.2, and fragments of Nelson’s axioms of internal set theory IST, namely *Idealisation* 1, *Standardisation* HAC_{int} , and *Transfer* PF-TP_\forall , defined as follows.

Definition 2.1 [QF-AC] For all finite types σ, τ and quantifier-free A :

$$(\forall x^\sigma)(\exists y^\tau)A(x, y) \rightarrow (\exists Y^{\sigma \rightarrow \tau})(\forall x^\sigma)A(x, Y(x)) \quad (\text{QF-AC}^{\sigma, \tau})$$

Definition 2.2

1. The axioms $\text{st}(x) \wedge x = y \rightarrow \text{st}(y)$ and $\text{st}(f) \wedge \text{st}(x) \rightarrow \text{st}(f(x))$.
2. The axiom $\text{st}(t)$ for each term t in the language of B_0 .
3. The axiom $\text{st}_0(x) \wedge y \leq_0 x \rightarrow \text{st}_0(y)$.

For the next definition, we note that x in (2.3) and $F(x)$ in (2.2) are both a *finite sequence* of objects of type σ , as discussed in Notation 2.4.

Definition 2.3

[Fragments of IST]

1. HAC_{int} : For any internal formula φ , we have

$$(\forall^{\text{st}}x^\rho)(\exists^{\text{st}}y^\sigma)\varphi(x, y) \rightarrow (\exists^{\text{st}}F^{\rho \rightarrow \sigma})(\forall^{\text{st}}x^\rho)(\exists y^\sigma \in F(x))\varphi(x, y). \quad (2.2)$$

2. I: For any internal formula φ , we have

$$(\forall^{\text{st}}x^\sigma)(\exists y^\tau)(\forall z^\sigma \in x)\varphi(z, y) \rightarrow (\exists y^\tau)(\forall^{\text{st}}z^\sigma)\varphi(z, y). \tag{2.3}$$

3. PF-TP $_{\forall}$: For any internal φ with all parameters shown, we have $\forall^{\text{st}}x \varphi(x) \rightarrow \forall x \varphi(x)$, i.e. x is the **only** free variable in $\varphi(x)$.

Notation 2.4 (Finite sequences). There are at least two ways of approaching ‘finite sequences of objects of type σ ’ in E-EFA $^\omega$: First of all, as in [2], we could extend E-EFA $^\omega$ with types σ^* for finite sequences of objects of type σ , add constants for the empty sequence and the operation of prepending an element to a sequence, as well as a list recursor satisfying the expected equations.

Secondly, as in [3], we could exploit the fact that one can code finite sequences of objects of type σ as a single object of type σ in such a way that every object of type σ codes a sequence. Moreover, the operations on sequences, such as extracting their length or concatenating them, are given by terms in Gödel’s T .

We choose the second option here and will often use set-theoretic notation as follows: ‘ \emptyset ’ is (the code of) the empty sequence, ‘ \cup ’ stands for concatenation, and ‘ $\{x\}$ ’ for the finite sequence of length 1 with sole component x . For x and y of the same type we will write $x \in y$ if x is equal to one of the components of the sequence coded by y . Furthermore, for $\alpha^{0 \rightarrow \rho}$ and k^0 , the finite sequence $\bar{\alpha}k$ is exactly $\langle \alpha(0), \alpha(1), \dots, \alpha(k-1) \rangle$. Finally, if Y is of type $\sigma \rightarrow \tau$ and x is of type σ we define $Y[x]$ of type τ as $Y[x] := \cup_{f \in Y} f(x)$.

With this notation in place, we can now formulate a crucial theorem from [3, Sect. 3].

Theorem 2.5. *For φ internal and Δ_{int} a collection of internal formulas, if the system $B_0 + \Delta_{\text{int}}$ proves $(\forall^{\text{st}}x)(\exists^{\text{st}}y)\varphi(x, y)$, then*

$$\text{E-EFA}^\omega + \text{QF-AC}^{1,0} + \Delta_{\text{int}} \vdash (\exists \Phi)(\forall x)(\exists y \in \Phi(x))\varphi(x, y) \tag{2.4}$$

By the results in Sect. 3 and [3], PF-TP $_{\forall}$ is useful for obtaining equivalences as in RM. However, this ‘usefulness’ comes at a price, as B_0^- (i.e. $B_0 \setminus \text{PF-TP}_{\forall}$) satisfies the following, where a *term of Gödel’s system T* is obtained, to be compared to the *existence* of a functional in (2.4). Hence, PF-TP $_{\forall}$ seems suitable for proof mining, as the latter deals with extracted *terms*

Theorem 2.6 (Term extraction). *If Δ_{int} is a collection of internal formulas and ψ is internal, and $B_0^- + \Delta_{\text{int}} \vdash (\forall^{\text{st}}x)(\exists^{\text{st}}y)\psi(x, y)$, then one can extract from the proof a term t from Gödel’s T such that*

$$\text{E-EFA}^\omega + \text{QF-AC}^{1,0} + \Delta_{\text{int}} \vdash (\forall x)(\exists y \in t(x))\psi(x, y). \tag{2.5}$$

We finish this section with some notations.

Notation 2.7 (Equality). The system E-EFA $^\omega$ includes equality ‘ $=_0$ ’ for numbers as a primitive. Equality ‘ $=_\tau$ ’ for x^τ, y^τ is:

$$[x =_\tau y] \equiv (\forall z_1^{\tau_1} \dots z_k^{\tau_k}) [xz_1 \dots z_k =_0 yz_1 \dots z_k], \tag{2.6}$$

if the type τ is composed as $\tau \equiv (\tau_1 \rightarrow \dots \rightarrow \tau_k \rightarrow 0)$. Inequality ' \leq_τ ' is (2.6) with $=_0$ replaced by \leq_0 . Similarly, we define 'approximate equality \approx_τ ' as:

$$[x \approx_\tau y] \equiv (\forall^{\text{st}} z_1^{\tau_1} \dots z_k^{\tau_k}) [xz_1 \dots z_k =_0 yz_1 \dots z_k] \tag{2.7}$$

Notation 2.8 (Real numbers and related notions in \mathbf{B}_0)

1. Natural numbers correspond to type zero objects. Rational numbers are defined as quotients of natural numbers, and ' $q \in \mathbb{Q}$ ' has its usual meaning.
2. A (standard) real number x is a (standard) fast-converging Cauchy sequence $q_{(\cdot)}^1$, i.e. $(\forall n^0, i^0)(|q_n - q_{n+i}| <_0 \frac{1}{2^n})$.
3. We write ' $x \in \mathbb{R}$ ' to express that $x^1 = (q_{(\cdot)}^1)$ is a real as in the previous item and $[x](k) := q_k$ for the k -th approximation of x .
4. Two reals x, y represented by $q_{(\cdot)}$ and $r_{(\cdot)}$ are *equal*, denoted $x =_{\mathbb{R}} y$, if $(\forall n^0)(|q_n - r_n| \leq \frac{1}{2^{n-1}})$. Inequality $<_{\mathbb{R}}$ is defined similarly.
5. We write $x \approx y$ if $(\forall^{\text{st}} n^0)(|q_n - r_n| \leq \frac{1}{2^n})$ and $x \gg y$ if $x > y \wedge x \not\approx y$.
6. Functions $F : \mathbb{R} \rightarrow \mathbb{R}$ are represented by $\Phi^{1 \rightarrow 1}$ such that

$$(\forall x, y)(x =_{\mathbb{R}} y \rightarrow \Phi(x) =_{\mathbb{R}} \Phi(y)). \tag{RE}$$

7. Sets of natural numbers X^1, Y^1, Z^1, \dots are represented by binary sequences.

Notation 2.9 (Using $\mathbf{HAC}_{\text{int}}$). As noted in Notation 2.4, finite sequences play an important role in \mathbf{B}_0 . In particular, $\mathbf{HAC}_{\text{int}}$ produces a functional which outputs a *finite sequence* of witnesses. However, $\mathbf{HAC}_{\text{int}}$ provides an actual *witnessing functional* assuming (i) $\tau = 0$ in $\mathbf{HAC}_{\text{int}}$ and (ii) the formula φ from $\mathbf{HAC}_{\text{int}}$ is 'sufficiently monotone' as in: $(\forall^{\text{st}} x^\sigma, n^0, m^0)([n \leq_0 m \wedge \varphi(x, n)] \rightarrow \varphi(x, m))$. Indeed, in this case one simply defines $G^{\sigma+1}$ by $G(x^\sigma) := \max_{i < |F(x)|} F(x)(i)$ which satisfies $(\forall^{\text{st}} x^\sigma)\varphi(x, G(x))$. To save space in proofs, we will sometimes skip the (obvious) step involving the maximum of finite sequences, when applying $\mathbf{HAC}_{\text{int}}$. We assume the same convention for other finite sequences e.g. obtained from Theorem 2.6, or the contraposition of idealisation I.

3 Reverse Mathematics and Nonstandard Analysis

In Sects. 3.1 and 3.2, we establish the equivalence between the *nonstandard compactness* of Cantor space and the *Heine-Borel compactness* (for **any** open cover) of the unit interval. The latter essentially predates¹ set theory, and is hence definitely part of 'ordinary mathematics' in the sense of RM. We establish similar results for theorems based on WWKL in Sect. 3.3. We shall use 'computable' in the sense of Kleene's schemes S1–S9 inside ZFC ([11, Sect. 5.1.1]).

¹ Heine-Borel compactness was studied before 1895 by Cousin ([4, p. 22]). The collected works of Pincherle contain a footnote by the editors ([17, p. 67]) stating that the associated *Teorema* (from 1882) corresponds to the Heine-Borel theorem.

3.1 Nonstandard Compactness and the Special Fan Functional

The main result of this section is Theorem 3.4, which establishes an equivalence involving the nonstandard compactness of Cantor space and the *special fan functional*, introduced in [18] and studied in detail in [14]. The variable ‘ T ’ is reserved for trees, and ‘ $T \leq_1 1$ ’ means that T is a binary tree.

Definition 3.1 [Special fan functional]. We define $\text{SCF}(\Theta)$ as follows for $\Theta^{(2 \rightarrow (0 \times 1))}$:

$$(\forall g^2, T^1 \leq_1 1) [(\forall \alpha \in \Theta(g)(2))(\bar{\alpha}g(\alpha) \notin T) \rightarrow (\forall \beta \leq_1 1)(\exists i \leq \Theta(g)(1))(\bar{\beta}i \notin T)].$$

Any functional Θ satisfying $\text{SCF}(\Theta)$ is referred to as a *special fan functional*.

From a computability theoretic perspective, the main property of Θ is the selection of $\Theta(g)(2)$ as a finite sequence of binary sequences $\langle f_0, \dots, f_n \rangle$ such that the neighbourhoods defined from $\bar{f}_i g(f_i)$ for $i \leq n$ form a cover of Cantor space; almost as a by-product, $\Theta(g)(1)$ can then be chosen to be the maximal value of $g(f_i) + 1$ for $i \leq n$. No type two functional computes Θ such that $\text{SCF}(\Theta)$ ([14]), while the following functional can compute Θ via a term of Gödel’s T ([20]).

$$(\exists \xi^3)(\forall Y^2) [(\exists f^1)(Y(f) = 0) \leftrightarrow \xi(Y) = 0]. \tag{E3}$$

We stress that g^2 in $\text{SCF}(\Theta)$ may be *discontinuous* and that Kohlenbach has argued for the study of discontinuous functionals in higher-order RM ([10]). Furthermore, $\text{RCA}_0^\omega + (\exists \Theta)\text{SCF}(\Theta)$ is conservative over WKL_0 ([14, 18]), and Θ naturally emerges from Tao’s notion of *metastability*, as discussed in [15, 20, 21].

The special fan functional arose from STP, the *nonstandard compactness* of Cantor space as in *Robinson’s theorem* ([7]). This fragment of *Standard Part* is also known as the ‘nonstandard counterpart of weak König’s lemma’ ([8]).

$$(\forall \alpha^1 \leq_1 1)(\exists^{\text{st}} \beta^1 \leq_1 1)(\alpha \approx_1 \beta), \tag{STP}$$

as explained by the equivalence between STP and (3.2), as follows.

Theorem 3.2. *In B_0^- , STP is equivalent to the following:*

$$(\forall^{\text{st}} g^2)(\exists^{\text{st}} w^1 \leq_1 1, k^0) [(\forall T^1 \leq_1 1)((\forall \alpha^1 \in w)(\bar{\alpha}g(\alpha) \notin T) \rightarrow (\forall \beta \leq_1 1)(\exists i \leq k)(\bar{\beta}i \notin T))], \tag{3.1}$$

as well as to the following:

$$(\forall T^1 \leq_1 1) [(\forall^{\text{st}} n)(\exists \beta)(|\beta| = n \wedge \beta \in T) \rightarrow (\exists^{\text{st}} \alpha^1 \leq_1 1)(\forall^{\text{st}} n)(\bar{\alpha}n \in T)]. \tag{3.2}$$

Furthermore, B_0^- proves $(\exists^{\text{st}} \Theta)\text{SCF}(\Theta) \rightarrow \text{STP}$.

Proof. A detailed proof may be found in any of the following: [14, 18, 21]. In a nutshell, the implication (3.1) ← (3.2) follows by taking the contraposition of the latter and introducing standard g^2 in the antecedent of the resulting formula. One then uses *Idealisation I* to pull the standard quantifiers to the front

and obtains (3.1). The other implication follows by pushing the standard quantifiers in the latter back inside. For the remaining implication $STP \rightarrow (3.2)$ (the other one and the final part then being trivial), one uses *overspill* (See [2, Sect. 3]) to obtain a sequence of nonstandard length for a tree $T \leq_1 1$ satisfying the antecedent of (3.2), and STP converts this sequence into a standard path in T . \square

For the below results, we need the following corollary which expresses the (trivial but important) fact that the type of the universal quantifier in STP (and equivalent formulations) may be lowered. We view $\alpha^0 \leq_0 1$ as a finite binary sequence; we define $\hat{\alpha}$ to be $\alpha * 00\dots$, i.e. the type one object obtained by concatenating α with 0^1 . Similarly, $T^0 \leq_0 1$ is a binary tree of type zero, and $SCF_0(\Theta)$ is the specification of Θ restricted to trees $T^0 \leq_0 1$.

Corollary 3.3. *In B_0^- , STP is equivalent to $(\forall \alpha^0 \leq_0 1)(\exists^{st} \beta^1 \leq 1)(\hat{\alpha} \approx_1 \beta)$, and also to the following:*

$$(\forall T^0 \leq_0 1)[(\forall^{st} n)(\exists \beta)(|\beta| = n \wedge \beta \in T) \rightarrow (\exists^{st} \alpha^1 \leq_1 1)(\forall^{st} n)(\bar{\alpha}n \in T)], \tag{3.3}$$

and also the following:

$$\begin{aligned} (\forall^{st} g^2)(\exists^{st} w^1 \leq_1 1, k^0)[(\forall T^0 \leq_0 1)((\forall \alpha^1 \in w)(\bar{\alpha}g(\alpha) \notin T) \\ \rightarrow (\forall \beta \leq_1 1)(\exists i \leq k)(\bar{\beta}i \notin T))]. \end{aligned} \tag{3.4}$$

The system $E-EFA^\omega + QF-AC^{1,0}$ proves $(\exists \Theta)SCF(\Theta) \leftrightarrow (\exists \Theta_0)SCF_0(\Theta_0)$.

Proof. Now, (3.3) \leftrightarrow (3.4) follows in the same way as for (3.2) \leftrightarrow (3.1). The first forward implication is trivial while the first reverse implication follows by considering $\bar{\alpha}N$ for nonstandard N^0 and $\alpha^1 \leq_1 1$. The implication $STP \rightarrow (3.3)$ follows in the same way as in the proof of the theorem. Note that $T^0 \leq_0 1$ as in the antecedent of (3.3) must be nonstandard by the basic axioms in Definition 2.2. The implication (3.3) \rightarrow (3.2) follows by restricting T^1 to sequences of some fixed nonstandard length, which yields a type zero object. The final equivalence follows by applying Theorem 2.6 to ‘ $B_0^- \vdash (3.1) \leftrightarrow (3.4)$ ’. \square

The following theorem was proved in [3] using the *Suslin functional*, rather than the *much weaker* Turing jump functional (\exists^2) as follows:

$$(\exists \varphi^2)(\forall f^1)[(\exists n)(f(n) = 0) \leftrightarrow \varphi(f) = 0]. \tag{\exists^2}$$

Theorem 3.4. *The system $B_0 + (\exists^2) + QF-AC^{2,1}$ proves $STP \leftrightarrow (\exists \Theta)SCF(\Theta)$, while the system $B_0^- + (\exists^3) + QF-AC$ does not.*

Proof. The reverse implication is immediate using $PF-TP_\forall$ and Theorem 3.2. For the forward implication, STP implies (3.4) by Corollary 3.3. Drop the second ‘ st ’ in (3.4), and apply $PF-TP_\forall$ to the resulting formula to obtain

$$\begin{aligned} (\forall g^2)(\exists w^1 \leq_1 1, k^0)[(\forall T^0 \leq_0 1)[(\forall \alpha^1 \in w)(\bar{\alpha}g(\alpha) \notin T) \\ \rightarrow (\forall \beta \leq_1 1)(\exists i \leq k)(\bar{\beta}i \notin T)]], \end{aligned} \tag{3.5}$$

where the formula in big square brackets is equivalent to a quantifier-free one, thanks to (\exists^2) . Apply $\text{QF-AC}^{2,1}$ to (3.5) to obtain Θ_0 producing w^1, k^0 from g^2 as in (3.5). Corollary 3.3 yield $(\exists\Theta)\text{SCF}(\Theta)$.

The non-implication follows from [14, Theorem 4.2] as the latter expresses that the special fan functional is not computable in any type two functional. Indeed, STP is equivalent to (3.1) by Theorem 3.2 and applying Theorem 2.6 to $\text{B}_0^- + (\exists^3) + \text{QF-AC} + (\exists\Theta)\text{SCF}(\Theta) \vdash (3.1)$, one obtains a term t of Gödel's T such that $\text{SCF}(t)$, which is impossible. \square

3.2 Nonstandard Compactness and Heine-Borel Compactness

We prove an equivalence between STP and the Heine-Borel theorem *in the general² case*, i.e. the statement that any (possibly uncountable) open cover of the unit interval has a finite sub-cover. In particular, any $\Psi : \mathbb{R} \rightarrow \mathbb{R}^+$ gives rise to a ‘canonical’ open cover $\cup_{x \in [0,1]} I_x$ of $[0, 1]$ where $I_x^\Psi \equiv (x - \Psi(x), x + \Psi(x))$. Hence, the Heine-Borel theorem trivially implies the following statement:

$$(\forall \Psi : \mathbb{R} \rightarrow \mathbb{R}^+)(\exists w^1)(\forall x \in [0, 1])(\exists y \in w)(x \in I_y^\Psi). \tag{HBU}$$

By Footnote 1, HBU is part of ordinary mathematics as it predates set theory. Furthermore, HBU is equivalent to many basic properties of the *gauge integral* ([16]). The latter is an extension of Lebesgue’s integral and provides a (direct) formalisation of the Feynman path integral.

Theorem 3.5. *The system $\text{B}_0 + (\exists^2) + \text{QF-AC}^{2,1}$ proves that $\text{STP} \leftrightarrow \text{HBU}$, while the system $\text{B}_0^- + (\exists^3) + \text{QF-AC}$ does not.*

Proof. Note that (\exists^2) allows us to (uniformly) convert reals into their binary representation (choosing the one with trailing zeros in case of non-uniqueness). Hence, any type two functional can be modified to satisfy (RE) from Notation 2.8 if necessary. Hence, HBU immediately generalises to *any* Ψ^2 . Now, for the reverse implication, note that HBU trivially implies

$$\underline{(\forall \Psi^2)(\exists w^1)(\forall q^0 \in [0, 1])(\exists y \in w)(|q - y| < \frac{1}{\Psi(y) + 1})}, \tag{3.6}$$

where the underlined formula in (3.6) may be treated as quantifier-free, due to the presence of (\exists^2) in the base theory. Applying QF-AC to (3.6), we obtain:

$$(\exists \Phi^{2 \rightarrow 1})(\forall \Psi^2)(\forall q^0 \in [0, 1])(\exists y \in \Phi(\Psi))(|q - y| < \frac{1}{\Psi(y) + 1}), \tag{3.7}$$

and applying PF-TP_\forall to (3.7) implies that

$$(\exists^{\text{st}} \Phi^{2 \rightarrow 1})(\forall \Psi^2)(\forall q^0 \in [0, 1])(\exists y \in \Phi(\Psi))(|q - y| < \frac{1}{\Psi(y) + 1}), \tag{3.8}$$

² The Heine-Borel theorem in RM is restricted to *countable* covers ([22, IV.1]).

and we now show that (3.8) implies STP. Since standard functionals yield standard outputs for standard inputs by Definition 2.2, (3.8) immediately implies

$$(\forall^{\text{st}}\Psi^2)(\forall q^0 \in [0, 1])(\exists^{\text{st}}y^1 \in [0, 1])(|q - y| < \frac{1}{\Psi(y) + 1}).$$

Now, $(\forall^{\text{st}}\Psi^2)(\exists^{\text{st}}y \in [0, 1])(|q - y| < \frac{1}{\Psi(y)+1})$ implies $(\exists^{\text{st}}y \in [0, 1])(q \approx y)$; indeed, $(\forall^{\text{st}}y \in [0, 1])(q \not\approx y)$ implies $(\forall^{\text{st}}y \in [0, 1])(\exists^{\text{st}}k^0)(|q - y| \geq \frac{1}{k})$, and applying HAC_{int} yields standard Ξ^2 such that $(\forall^{\text{st}}y \in [0, 1])(\exists k^0 \in \Xi(y))(|q - y| \geq \frac{1}{k})$. Defining standard Ψ_0^2 as $\Psi_0(y) := \max_{i < |\Xi(y)|} \Xi(y)(i)$, we obtain $(\forall^{\text{st}}y \in [0, 1])(|q - y| \geq \frac{1}{\Psi_0(y)+1})$, a contradiction. Hence, we have proved $(\forall q^0 \in [0, 1])(\exists^{\text{st}}y \in [0, 1])(q \approx y)$, which immediately yields $(\forall x^1 \in [0, 1])(\exists^{\text{st}}y^1 \in [0, 1])(x \approx y)$, as we have $x \approx [x](N)$ for any $x \in [0, 1]$ and nonstandard N^0 . However, every real has a binary expansion in RCA_0 (See [6]), and B_0 similarly proves that every (standard) real has a (standard) binary expansion. A real with non-unique binary expansion can be summed with an infinitesimal to yield a real with a unique binary expansion. Hence, the previous yields that $(\forall \alpha^1 \leq_1 1)(\exists^{\text{st}}\beta^1 \leq_1 1)(\alpha \approx_1 \beta)$, which is just STP.

For the forward direction, STP implies $(\forall x^1 \in [0, 1])(\exists^{\text{st}}y^1 \in [0, 1])(x \approx y)$ as in the previous paragraph, and we thus have:

$$(\forall^{\text{st}}\Psi^2)(\forall x^1 \in [0, 1])(\exists^{\text{st}}y^1 \in [0, 1])(|x - y| < \frac{1}{\Psi(y) + 1}), \tag{3.9}$$

Applying *Idealisation* to (3.9), we obtain

$$(\forall^{\text{st}}\Psi^2)(\exists^{\text{st}}w^1)(\forall x^1 \in [0, 1])(\exists y \in w)(|x - y| < \frac{1}{\Psi(y) + 1}). \tag{3.10}$$

Dropping the second ‘st’ in (3.10) and applying PF-TP_\forall , we obtain HBU.

The non-implication follows from [14, Theorem 4.2] as the latter expresses that the special fan functional is not computable in any type two functional. Indeed, STP is equivalent to (3.1) by Theorem 3.2, and apply Theorem 2.6 to $\text{B}_0^- + (\exists^3) + \text{HBU} + \text{QF-AC} \vdash (3.1)$, to obtain a term t of Gödel’s T such that $\text{SCF}(t)$, which is impossible. \square

3.3 Weak Compactness and the Weak Fan Functional

Clearly, HBU is a generalisation of WKL from RM. In this section, we list results similar to Theorems 3.4 and 3.5 for generalisations of WWKL. The *weak fan functional* Λ from [14] arises from the axiom WWKL, as follows:

$$(\forall T \leq_1 1)[\mu(T) >_{\mathbb{R}} 0 \rightarrow (\exists \beta \leq_1 1)(\forall m)(\bar{\beta}m \in T)], \tag{WWKL}$$

where ‘ $\mu(T) >_{\mathbb{R}} 0$ ’ is $(\exists k^0)(\forall n^0)(\frac{|\{\sigma \in T : |\sigma| = n\}|}{2^n} \geq \frac{1}{k})$. Although WWKL is not part of the Big Five, it sports *some* equivalences ([22, X.1]). The following fragment of *Standard Part* is the nonstandard counterpart of WWKL, as studied in [23]:

$$(\forall T^1 \leq_1 1)[\mu(T) \gg 0 \rightarrow (\exists^{\text{st}}\beta^1 \leq_1 1)(\forall^{\text{st}}m^0)(\bar{\beta}m \in T)], \tag{LMP}$$

where ‘ $\mu(T) \gg 0$ ’ is just the formula $[\mu(T) >_{\mathbb{R}} 0]^{\text{st}}$. Clearly, WWKL and LMP are weakened versions of WKL and STP; the following weaker version of the special fan functional arises from LMP. As for the special one, there is *no unique* weak fan functional, i.e. it is in principle incorrect to refer to ‘the’ weak fan functional.

Definition 3.6 [Weak fan functional]. We define $\text{WCF}(\Lambda)$ for $\Lambda^{(2 \rightarrow (1 \times 1))}$:

$$(\forall k^0, g^2, T^1 \leq_1 1)[(\forall \alpha \in \Lambda(g, k)(2))(\overline{\alpha}g(\alpha) \notin T) \rightarrow (\exists n \leq \Lambda(g, k)(1))(L_n(T) \leq \frac{1}{k})].$$

Any Λ satisfying $\text{WCF}(\Lambda)$ is referred to as a *weak fan functional*.

Now, WWKL is equivalent to the following statement: *for every X^1 , there is Y^1 which is Martin-Löf random relative to X* , as proved in [1, Theorem 3.1]. This equivalence is proved in RCA_0 , and the latter also suffices to e.g. define a *universal* Martin-Löf test $(U_i^X)_{i \in \mathbb{N}}$ (relative to any X^1). The latter has type $0 \rightarrow 1$ and represents a universal and effective (relative to X) null set, i.e. a rare event. Intuitively, Y is (Martin-Löf) *random* relative to X , if Y is not in such a rare event. To make this more precise, define ‘ $f^1 \in [\sigma^0]$ ’ as $\overline{f}|\sigma| =_0 \sigma$ for any finite binary sequence and define $\text{MLR}(Y, X)$ as $(\exists i^0)(\forall w^0 \in U_i^X)(Y \notin [w])$.

We can now define restrictions of STP and HBU to Martin-Löf random reals.

$$(\forall^{\text{st}} X^1)(\forall Y^1)(\exists^{\text{st}} Z^1)([\text{MLR}(Y, X)]^{\text{st}} \rightarrow Z \approx_1 Y). \tag{MLR}_{\text{ns}}$$

Let $\text{MLR}(X, Y, i)$ be $\text{MLR}(X, Y)$ without the leading quantifier. Now consider

$$(\forall \Psi^2, k^0, X^1)(\exists w^1)(\forall Y)(\exists Z \in w)(\text{MLR}(Y, X, k) \rightarrow Y \in [\overline{Z}\Psi(Z)]). \tag{HBU}_{\text{ml}}$$

Note that HBU_{ml} expresses that the canonical cover $\cup_{f \in 2^{\mathbb{N}}} [\overline{f}\Psi(f)]$ has a finite sub-cover which covers all reals which are random *and already outside the universal test at level U_k^X* of the universal test. Since $\mu(U_k) \leq \frac{1}{2^k}$, the finite sub-cover need not cover a measure one set in Cantor space. The following theorem is proved in the same way as Theorems 3.4 and 3.5.

Theorem 3.7. *The system $\text{B}_0 + (\exists^2) + \text{QF-AC}^{2,1}$ proves $\text{LMP} \leftrightarrow \text{MLR}_{\text{ns}} \leftrightarrow (\exists \Lambda)\text{WCF}(\Lambda) \leftrightarrow \text{HBU}_{\text{ml}}$.*

Finally, the ‘st’ in the antecedent of LMP (and MLR_{ns}) is essential: in particular, we show that STP (and hence Θ) is *robust* in the sense of RM ([12, p. 495]), but LMP is not. Consider the following variations of LMP and STP.

$$(\forall T \leq_1 1)[\mu(T) >_{\mathbb{R}} 0 \rightarrow (\exists^{\text{st}} \beta \leq_1 1)(\forall^{\text{st}} m)(\overline{\beta}m \in T)], \tag{LMP}^+$$

$$(\forall T \leq_1 1)[(\forall n^0)(\exists \beta^0)(\beta \in T \wedge |\beta| = n) \rightarrow (\exists^{\text{st}} \beta \leq_1 1)(\forall^{\text{st}} m)(\overline{\beta}m \in T)],$$

where the second one is called ‘ STP^- ’. We have the following theorem.

Theorem 3.8. *In $\text{B}_0^- + \text{WWKL}$, we have $\text{STP} \leftrightarrow \text{LMP}^+ \leftrightarrow \text{STP}^-$.*

Proof. For the first equivalence, we only need to prove $\text{STP} \leftarrow \text{STP}^-$, which follows by taking a tree $T \leq_1 1$ as in the antecedent STP , noting that by overspill it has a sequence of nonstandard length, and extending this sequence with $00\dots$ to obtain a tree as in the antecedent of STP^- . Then STP^- yields a standard path in the standard part of the modified tree, which is thus also in the standard part of the original tree. For $\text{STP} \rightarrow \text{LMP}^+$, apply STP to the path claimed to exist by WWKL and note that we obtain LMP^+ . For $\text{LMP}^+ \rightarrow \text{STP}$, fix $f^1 \leq_1 1$ and nonstandard N . Define the tree $T \leq_1 1$ which is f until height N , followed by the full binary tree. Then $\mu(T) >_{\mathbb{R}} 0$ and let standard $g^1 \leq_1 1$ be such that $(\forall^{\text{st}} n)(\bar{g}n \in T)$. By definition, $f \approx_1 g$ follows, and we are done. \square

References

1. Avigad, J., Dean, E.T., Rute, J.: Algorithmic randomness, reverse mathematics, and the dominated convergence theorem. *Ann. Pure Appl. Log.* **163**(12), 1854–1864 (2012)
2. van den Berg, B., Briseid, E., Safarik, P.: A functional interpretation for nonstandard arithmetic. *Ann. Pure Appl. Log.* **163**, 1962–1994 (2012)
3. van den Berg, B., Sanders, S.: Reverse Mathematics and parameter-free Transfer (Submitted, 2015). arXiv: <http://arxiv.org/abs/1409.6881>
4. Cousin, P.: Sur les fonctions de n variables complexes. *Acta Math.* **19**, 1–61 (1895)
5. Friedman, H.: Some systems of second order arithmetic and their use. In: *Proceedings of the ICM (Vancouver, B. C., 1974)*, vol. 1, pp. 235–242 (1975)
6. Hirst, J.L.: Representations of reals in reverse mathematics. *Bull. Pol. Acad. Sci. Math.* **55**(4), 303–316 (2007)
7. Hurd, A.E., Loeb, P.A.: *An Introduction to Nonstandard Real Analysis*, Pure and Applied Mathematics, vol. 118. Academic Press Inc., Orlando (1985)
8. Keisler, H.J.: Nonstandard arithmetic and reverse mathematics. *Bull. Symb. Log.* **12**, 100–125 (2006)
9. Kohlenbach, U.: *Applied Proof Theory: Proof Interpretations and Their Use in Mathematics*. Springer Monographs in Mathematics. Springer, Berlin (2008)
10. Kohlenbach, U.: Higher order reverse mathematics. In: *Reverse Mathematics 2001. Lecture Notes in Logic*, vol. 21. ASL, pp. 281–295 (2005)
11. Longley, J., Normann, D.: *Higher-order Computability. Theory and Applications of Computability*. Springer, Berlin (2015)
12. Montalbán, A.: Open questions in reverse mathematics. *Bull. Symb. Log.* **17**(3), 431–454 (2011)
13. Nelson, E.: Internal set theory: a new approach to nonstandard analysis. *Bull. Amer. Math. Soc.* **83**(6), 1165–1198 (1977)
14. Normann, D., Sanders, S.: *Nonstandard Analysis, Computability Theory, and Their Connections* (2017). arXiv: <https://arxiv.org/abs/1702.06556>
15. Normann, D., Sanders, S.: *Nonstandard Analysis, Computability Theory, and Metastability*. In: *Preparation* (2018)
16. Normann, D., Sanders, S.: *On the mathematical and foundational significance of the uncountable* (Submitted, 2017). <https://arxiv.org/abs/1711.08939>
17. Pincherle, S.: *Sopra alcuni sviluppi in serie per funzioni analitiche*. *Opere Scelte*, I, Roma **1954**, 64–91 (1882)

18. Sanders, S.: The Gandy-Hyland functional and a computational aspect of Non-standard Analysis, To appear in *Computability* (2015). arXiv: <http://arxiv.org/abs/1502.03622>
19. Sanders, S.: Formalism 16, *Synthese*, S.I.: Foundations of Mathematics, pp. 1–42 (2017)
20. Sanders, S.: Metastability and higher-order computability. In: Artemov, S., Nerode, A. (eds.) *LFCS 2018*. LNCS, vol. 10703, pp. 309–330. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72056-2_19
21. Sanders, S.: To be or not to be constructive, Special issue of *Indagationes Mathematicae*, p. 68 (2017). <https://arxiv.org/abs/1704.00462>
22. Simpson, S.G.: *Subsystems of second order arithmetic*. Perspectives in Logic, CUP, 2nd edn. Cambridge University Press, Newyork (2009)
23. Simpson, S.G., Yokoyama, K.: A nonstandard counterpart of WWKL. *Notre Dame J. Form. Log.* **52**(3), 229–243 (2011)



Bit Complexity of Computing Solutions for Symmetric Hyperbolic Systems of PDEs (Extended Abstract)

Svetlana V. Selivanova^{1,2} and Victor L. Selivanov^{3,4}(✉)

¹ S.L. Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia
sweseliv@gmail.com

² KAIST, Daejeon, South Korea

³ A.P. Ershov Institute of Informatics Systems SB RAS, Novosibirsk, Russia
vseliv@iis.nsk.su

⁴ Kazan Federal University, Novosibirsk, Russia

Abstract. We establish upper bounds of bit complexity of computing solution operators for symmetric hyperbolic systems of PDEs. Here we continue the research started in our papers of 2009 and 2017, where computability, in the rigorous sense of computable analysis, has been established for solution operators of Cauchy and dissipative boundary-value problems for such systems.

Keywords: Symmetric hyperbolic system · Solution operator
Bit complexity · Guaranteed precision · Symbolic computations
Algebraic real · Symmetric matrix · Eigenvalue · Eigenvector
Difference scheme

1 Introduction

The algorithms used in mathematics-oriented software can be divided into two big classes: symbolic algorithms which aim to find precise solutions, and approximate algorithms which aim to find “good enough” approximations to precise solutions. The symbolic algorithms are implemented e.g. in computer algebra systems while the approximate algorithms are included into numerical mathematics packages. Both classes of algorithms are widely used in applications and in mathematical research. The symbolic algorithms correspond well to computations on discrete structures (with mathematical foundations in the classical

S. V. Selivanova—The work of first author was supported by the National Research Foundation of Korea (grant NRF-2017R1E1A1A03071032), International Research & Development Program of the Korean Ministry of Science and ICT grant NRF-2016K1A3A7A03950702 and the Russian Foundation of Fundamental Research, project No 17-01-00801.

V. L. Selivanov—The work of second author was funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities, project No 1.12878.2018/12.1.

computability and complexity theory) while the approximate algorithms help to compute on continuous structures (with mathematical foundations in the field of computability and complexity in analysis evolving under the slogan “Exact real computation”).

An important idea relating the both classes of algorithms is to approximate the solution with “guaranteed precision”. This is of crucial importance for safety-critical applications but often requires much additional work. Accordingly, even the existence of such a guaranteed-precision algorithm is often not obvious, and when it exists it could require an additional work to establish reasonable complexity bounds. In many cases the statement of a guaranteed-precision version of some problem on a continuous structure (which requires to apply numerical mathematics and/or computable analysis) reduces it to a problem on a discrete structure which enables to apply the classical computability and complexity theory (sometimes called bit complexity).

In this paper, we investigate the bit complexity of finding guaranteed precision solutions for Cauchy and boundary-value problems for symmetric hyperbolic systems of PDEs $A \frac{\partial \mathbf{u}}{\partial t} + \sum_{i=1}^m B_i \frac{\partial \mathbf{u}}{\partial x_i} = f(t, x_1, \dots, x_m)$ where $A = A^* > 0$ and $B_i = B_i^*$ are symmetric $n \times n$ -matrices, $t \geq 0$, $x = (x_1, \dots, x_m) \in Q = [0, 1]^m$, $f : [0, +\infty) \times Q \rightarrow \mathbb{R}^n$ is a partial function. Such systems can be used to describe a wide variety of physical processes like those considered in the theories of elasticity, acoustics, electromagnetism etc., see e.g. [5, 7, 8]. Accordingly, many people from theoretical and numerical mathematics worked on the existence and uniqueness theorems as well as on numerical methods of computing solution operators for problems related to such systems (the explicit solution formulas exist only in some simplest particular cases).

In [13–15] we developed an approach to the study of computability of solutions for Cauchy and dissipative boundary-value problems for such systems based on finite-dimensional approximations (the so called difference schemes widely used in numerical analysis) and established the computability of solution operators in the rigorous sense of the TTE approach to computable analysis [4, 17]. The main obstacle in proving the computable dependence of solutions on the input matrices A, B_i is the fact that all known stable difference schemes for finding the approximate solutions use eigenvectors of some matrices and matrix pencils related to A, B_i but these eigenvectors are known to be non-computable [18]. To overcome the obstacle, we considered in [14, 15] restrictions of the solution operators to computably presentable real closed number fields and have shown that such restricted solution operators are computable. This fact together with close relationships of such fields to the field of computable reals (also established in [14, 15]) imply that the solution operators are computable for any fixed computable input matrices.

In this paper we develop the approach from [14, 15] to establish some reasonable upper bounds for some guaranteed-precision problems related to symmetric hyperbolic systems. To our knowledge, these are the first such bounds in the literature (though the bit complexity of some guaranteed-precision problems for differential equations was considered before, see e.g. [11]). Our approach makes

a heavy use of some known algorithms of computer algebra (exact computations with integers, rationals, algebraic reals and polynomials, see e.g. [1, 2, 10]), together with some algorithms from numerical mathematics and computable analysis used in [13–15]. Altogether, our proofs demonstrate a fruitful mix of methods from symbolic and numerical computation.

2 Preliminaries

2.1 Cauchy and Boundary-Value Problems

The Cauchy problem for a symmetric hyperbolic system is stated as follows:

$$\begin{cases} A \frac{\partial \mathbf{u}}{\partial t} + \sum_{i=1}^m B_i \frac{\partial \mathbf{u}}{\partial x_i} = f(t, x_1, \dots, x_m), \quad t \geq 0, \\ \mathbf{u}|_{t=0} = \varphi(x_1, \dots, x_m), \end{cases} \tag{1}$$

where $A = A^* > 0$ and $B_i = B_i^*$ are constant symmetric $n \times n$ -matrices, $t \geq 0$, $x = (x_1, \dots, x_m) \in Q = [0, 1]^m$, $\varphi : Q \rightarrow \mathbb{R}^n$, $f : [0, +\infty) \times Q \rightarrow \mathbb{R}^n$ and $\mathbf{u} : [0, +\infty) \times Q \rightarrow \mathbb{R}^n$ is a partial function acting on the domain H of existence and uniqueness of the Cauchy problem (1). The set H is known to be (see e.g. [7]) the intersection of semi-spaces

$$t > 0, \quad x_i - \mu_{\max}^{(i)} t > 0, \quad x_i - 1 - \mu_{\min}^{(i)} t < 0, \quad (i = 1, \dots, m)$$

of \mathbb{R}^{m+1} where $\mu_{\min}^{(i)}, \mu_{\max}^{(i)}$ are respectively the minimum and maximum of the eigenvalues of the matrix $A^{-1}B_i$.

The boundary-value problem is stated as follows:

$$\begin{cases} A \frac{\partial \mathbf{u}}{\partial t} + \sum_{i=1}^m B_i \frac{\partial \mathbf{u}}{\partial x_i} = f(t, x_1, \dots, x_m), \\ \mathbf{u}|_{t=0} = \varphi(x_1, \dots, x_m), \\ \Phi_i^{(1)} \mathbf{u}(t, x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_m) = 0, \\ \Phi_i^{(2)} \mathbf{u}(t, x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_m) = 0, \\ i = 1, 2, \dots, m, \end{cases} \tag{2}$$

where the boundary coefficients (which are constant rectangular matrices) $\Phi_i^{(1)}, \Phi_i^{(2)}$ meet the following conditions:

- (1) The number of rows of $\Phi_i^{(1)}$ (respectively, $\Phi_i^{(2)}$) is equal to the number of positive (respectively, negative) eigenvalues of the matrices $A^{-1}B_i$; coincidence conditions of the initial and boundary conditions hold (such conditions depend on the particular problem and on the smoothness which we want to obtain).
- (2) The boundary conditions are assumed to be dissipative which means that

$$(B_i \mathbf{u}, \mathbf{u}) \leq 0 \text{ for } x_i = 0, \quad (B_i \mathbf{u}, \mathbf{u}) \geq 0 \text{ for } x_i = 1, \quad i = 1, 2, \dots, m. \tag{3}$$

Condition (1) guarantees existence of solution of the boundary problem (2) in the cylinder $[0, \infty) \times Q$, while (2) implies its uniqueness.

For both problems (1) and (2), theorems on continuous dependence of the solution on the input data hold, i.e. the problems are correctly posed, for reasonable functional classes. Both are practically important, and there exist several numerical methods for solving them, from which we choose those developed in [7]. Their convergence relies on the following well-known theorem (see e.g. [7]): if the difference scheme approximates the given difference problem and is stable (which is an intrinsic property of a difference scheme), then the (discrete) solution of the corresponding difference equations converges to the exact solution of the differential problem in an appropriate grid norm; the speed of convergence corresponds to the order of approximation.

2.2 Discretization of the Problems

To investigate complexity of computing solutions of the problems (1) and (2) (and even to formulate the results), we need discrete approximations of the given and unknown functions, as well as their interpolations. Therefore we first describe some discretization details (following the approach of [7]).

Consider, for any positive integer N , the uniform rectangular grid G_N on $Q = [0, 1]^m$ defined by the points

$$\left(\frac{i_1 - \frac{1}{2}}{2^N}, \frac{i_2 - \frac{1}{2}}{2^N}, \dots, \frac{i_m - \frac{1}{2}}{2^N} \right)$$

where $1 \leq i_1, i_2, \dots, i_m \leq 2^N$. Let $h = 1/2^N$ be the spatial grid step, τ be the time step. Denote $G_N^\tau = G_N \times \{\tau\}_{l=1}^M$. The numbers of space and time steps are respectively 2^N and M , related to the steps h and τ , which guarantee good properties of the difference scheme, will be specified below.

Note that the number of points in the grid G_N is 2^{Nm} , so the set \mathbb{Q}^{G_N} of grid functions $g^{(h)} : G_N \rightarrow \mathbb{Q}^n$ may be identified with $\mathbb{Q}^{n \cdot 2^{Nm}}$. We will consider the following grid norms

$$\|g^{(h)}\|_s = \max_{x \in G_N} |g^{(h)}(x)|, \quad \|g^{(h)}\|_{L_2}^2 = h^m \sum_{x \in G_N} \langle g^{(h)}(x), g^{(h)}(x) \rangle.$$

We will consider the sL_2 -norm on the vector spaces $\mathbb{Q}^{G_N^\tau}$ of grid functions $v^{(h)}(t, x)$ on such grids:

$$\|v^{(h)}\|_{sL_2} = \max_{t \in \{\tau\}_{l=1}^M} h^m \sum_{x \in G_N} \langle v^{(h)}(t, x), v^{(h)}(t, x) \rangle.$$

From the the known formulas for multilinear interpolation, linearity of the interpolation operators $\mathbf{u} \mapsto \tilde{\mathbf{u}}$ and $\mathbf{u}^{(h)} \mapsto \widetilde{\mathbf{u}^{(h)}}$ is obvious. It is known, that

$$\|u - \tilde{u}\|_s \leq \max \|D^2 u\|_s h^2. \tag{4}$$

Further we will construct, by means of a stable difference scheme approximating the differential system (1), a grid function v on G_N^τ such that

$$\|\mathbf{u} - \widetilde{v|_H}\|_{sL_2} < \varepsilon, \tag{5}$$

where $\varepsilon > 0$ is a given precision, \mathbf{u} is the solution of (1) and $\widetilde{v}|_H$ is the multilinear interpolation of v restricted to H .

2.3 Algebraic Preliminaries and Encodings

Here we briefly recall some relevant algebraic notions and facts. For details see e.g. [16]. Coefficients of matrices and polynomials will usually be taken from a fixed (ordered) field $\mathbb{F} \subseteq \mathbb{R}$ of reals. For a field \mathbb{F} , let $\mathbb{F}[x_1, x_2, \dots]$ denote the ring polynomials with coefficients in \mathbb{F} and variables x_1, x_2, \dots . In this paper we most often work with ordered fields $\mathbb{F} \in \{\mathbb{Q}, \mathbb{Q}(\alpha), \mathbb{A} \mid 0 \neq \alpha \in \mathbb{A}\}$ where \mathbb{Q} is the ordered field of rationals, \mathbb{A} is the ordered field of algebraic reals (which consists of the reals roots of rational polynomials), and $\mathbb{Q}(\alpha_1, \dots, \alpha_n)$ is the smallest subfield of \mathbb{R} containing given $\alpha_1, \dots, \alpha_n \in \mathbb{A}$. For any latter field there is a “primitive element” $\alpha \in \mathbb{A}$ with $\mathbb{Q}(\alpha) = \mathbb{Q}(\alpha_1, \dots, \alpha_n)$. Note that \mathbb{A} is the smallest really closed ordered field. With any $\alpha \in \mathbb{A}$ we associate the unique pair (p_α, k) where $p_\alpha \in \mathbb{Q}[x]$ is the minimal (hence, irreducible) unitary polynomial of degree ≥ 1 with $p_\alpha(\alpha) = 0$, and k satisfies $\alpha = \alpha_k$ where $\alpha_1 < \dots < \alpha_m$ is the increasing sequence of all real roots of p_α . Next we encode pairs (p_α, k) by words in a finite alphabet.

Note (see e.g. [3]) that for any finite alphabet Σ there is a natural encoding (injective function) $c : \Sigma^* \rightarrow \{0, 1\}^*$ of words over Σ by binary words such that the function c , its range $rng(c)$, and the inverse function c^{-1} are computable in linear time, so in all interesting cases we can without loss of generality stick to binary encodings. The standard binary encoding b of \mathbb{Q} induces the encoding $e : \mathbb{Q}[x] \rightarrow \{0, 1, *\}^*$ which associates with a polynomial $p = a_0 + a_1x + \dots + a_nx^n$, $a_n \neq 0$, its code $b(a_0) * \dots * b(a_n)$. By the remark above, we may convert e to a natural binary coding $b : \mathbb{Q}[x] \rightarrow \{0, 1\}^*$. Now we associate with any $\alpha \in \mathbb{A}$ the word $b(p_\alpha) * b(k)$ where (p_α, k) is the pair from the previous paragraph which yields an injection f from \mathbb{A} into $\{0, 1, *\}^*$.

Let now $\mathbb{A}_1 = (A_1; <, +, \times, 0, 1)$, where $A_1 = rng(f)$, be the isomorphic copy of \mathbb{A} induced by f . As follows from well-known facts about symbolic computations (see e.g. [1, 9, 10]), this copy is polynomial-time computable (p -computable), i.e. the set A_1 , as well as all the signature functions and predicates are p -computable. This shows that the ordered field \mathbb{A} is p -presentable, i.e. isomorphic to a p -computable structure. The ordered fields \mathbb{Q} and $\mathbb{Q}(\alpha)$ (for each $\alpha \in \mathbb{A}$) are also p -presentable (this is much easier to show than for \mathbb{A}). Note however that some important computational properties of the presentation of \mathbb{A} differ from those for the presentations of \mathbb{Q} and $\mathbb{Q}(\alpha)$. In particular, the “long sum” $(\alpha_1 * \dots * \alpha_n) \mapsto \alpha_1 + \dots + \alpha_n$ is not p -computable (even not computable in PSPACE) uniformly on n w.r.t. the presentation of \mathbb{A} (this follows from the results in [19]) but it is p -computable in \mathbb{Q} and $\mathbb{Q}(\alpha)$.

The last fact extends to the evaluation of arbitrary rational polynomials. The mentioned encoding of univariate rational polynomials induces encodings of $\mathbb{Q}[x_1, \dots, x_n]$ for each $n \geq 1$ which provide p -presentations of the corresponding rings. Furthermore, these encodings induce an encoding of $\mathbb{Q}[x_1, \dots] = \bigcup_n \mathbb{Q}[x_1, \dots, x_n]$ in which any of $\mathbb{Q}[x_1, \dots, x_n]$ is p -computable; moreover, the

evaluation partial function $\mathbb{Q}[x_1, \dots] \times \mathbb{Q}^* \rightarrow \mathbb{Q}$, where \mathbb{Q}^* is the set of finite strings over \mathbb{Q} , is p -computable. It is not hard to see (using well known algorithms of polynomial arithmetics [1]) that the same fact holds for the field $\mathbb{Q}(\alpha)$ for each $\alpha \in \mathbb{A}$. According to the previous paragraph, similar fact does not hold for the field \mathbb{A} .

We will use some results from [2] about the complexity of root-finding in the field $\mathbb{C}_{\text{alg}} = (\mathbb{C}_{\text{alg}}; +, \times, 0, 1)$ of complex algebraic numbers, i.e. of finding all roots of an equation $\alpha_e x^e + \dots + \alpha_1 x + \alpha_0 = 0$ where $\alpha_i \in \mathbb{C}_{\text{alg}}$ for $i \leq e$. More precisely, the authors of [2] consider equations of the form

$$t_e(\alpha_1, \dots, \alpha_k) x^e + \dots + t_1(\alpha_1, \dots, \alpha_k) x + t_0(\alpha_1, \dots, \alpha_k) = 0, \tag{6}$$

where $\alpha_1, \dots, \alpha_k \in \mathbb{C}_{\text{alg}}$ and $t_j(\bar{x}) \in \mathbb{Q}[x_1, \dots, x_k]$. The problem is to find a list of (codes of) all roots of (6) from given $b(\alpha_1) * \dots * b(\alpha_k)$ and $b(t_0(\bar{x})) * \dots * b(t_e(\bar{x}))$ where b is a natural binary encoding of \mathbb{C}_{alg} induced by the presentation of \mathbb{A} described above and by Gauss representation of complex numbers as pairs of reals. As shown in [2], the problem is solvable in polynomial time for any fixed k . Moreover, the same estimate holds for the version of this problem when one computes the list of all distinct real roots of (6) in increasing order.

Let $M_n(R)$ be the set of $n \times n$ -matrices over a (commutative associative with a unit element 1) ring R , and $M(R)$ be the union of all $M_n(R)$, $n \geq 1$. We use without reminding some terminology and notation from linear algebra. In particular, $\det(A)$ is the determinant of $A = (a_{ij}) \in M_n(R)$, $\text{diag}(a_1, \dots, a_n)$ is the diagonal matrix with the diagonal elements $a_1, \dots, a_n \in R$, so $I = I_n = \text{diag}(1, \dots, 1)$ is the unit matrix. The roots of the polynomial $ch_A = \det(\lambda I - A)$ are called *eigenvalues of $A \in M_n(\mathbb{C})$* . In general, the eigenvalues of a real matrix are complex numbers. The eigenvalues of a symmetric real matrix are always real.

Associate with any matrix $A = (a_{ij}) \in M_n(\mathbb{F})$ its code $c(A) \in \{0, 1, *\}^*$ by $c(A) = b(a_{11}) * \dots * b(a_{1n}) * b(a_{21}) * \dots * b(a_{2n}) * \dots * b(a_{n1}) * \dots * b(a_{nn})$ where b is a binary encoding of \mathbb{F} . These encodings induce the binary encoding of the set $M(\mathbb{F}) = \bigcup_n M_n(\mathbb{F})$ of all square matrices over \mathbb{F} in which any set $M_n(\mathbb{F})$ is p -computable. Furthermore, many matrix properties like symmetricity are also p -computable. It is known (see e.g. [12]) that these encodings give p -presentations of the rings $M_n(\mathbb{Q})$ uniformly on n (uniformity means that there is a polynomial bound working for all n). Moreover, it is not hard to check that evaluation of some “long” terms in these rings are also p -computable w.r.t. these presentations (in particular the function $A_1 * \dots * A_n \mapsto A_1 \times \dots \times A_n$ is p -computable). Even more involved matrix algorithms like computing of the determinant, computing of the inverse of a non-degenerate matrix, and Gauss method also work in polynomial time (see Chap. 3 [12] for additional details).

From results in [10,12] it follows that the polynomial time estimates of the previous paragraph remain true for the rings $M_n(\mathbb{Q}(\alpha))$ for each non-zero $\alpha \in \mathbb{A}$. Moreover, our presentation of the ring $M_n(\mathbb{A})$ is a p -presentation for any fixed $n \geq 1$, but not uniformly on n . The “long terms” are of course not p -computable w.r.t. our presentation for $M_n(\mathbb{A})$, even for a fixed n .

3 Main Results

Now we have enough notions and terminology to state our guaranteed-precision problems in a rigorous form.

First we consider the task of computing the domain H of existence and uniqueness of the Cauchy problem. By the structure of H in Sect. 2.1, the computation of H reduces to computing of the eigenvalues of the matrices $A^{-1}B_i$. Our algorithms for solving the Cauchy problem will be for technical reasons presented only for the case when H satisfies the condition $\mu_{\min}^{(i)} < 0 < \mu_{\max}^{(i)}$ for all $i = 1, \dots, m$; which often holds for natural physical systems. Note that this condition implies that the closure of H is a compact subset of $[0, +\infty) \times Q$.

The next result establishes the complexity of computing the set H satisfying the mentioned condition.

Theorem 1. *For each $m, n \geq 2$ there is a polynomial time algorithm which for given $A, B_1 \dots, B_m \in M_n(\mathbb{A})$ finds the vector $(\mu_{\max}^{(1)}, \dots, \mu_{\max}^{(m)}, \mu_{\min}^{(1)}, \dots, \mu_{\min}^{(m)})$ and checks the condition $\mu_{\min}^{(i)} < 0 < \mu_{\max}^{(i)}$ for all $i = 1, \dots, m$. Thus, the algorithm finds the domain H satisfying the condition above, or reports on the absence of such a domain.*

Now we state a guaranteed-precision version of a restricted Cauchy problem (1). Let $m, n \geq 2$ and $a \geq 1$ be fixed integers. The version asks for an algorithm (and its complexity estimation) which, for any given matrices $A, B_1 \dots, B_m \in M_n(\mathbb{A})$ and polynomials $\varphi_1 \dots, \varphi_n \in \mathbb{Q}[x_1 \dots, x_m]$, $f_1 \dots, f_n \in \mathbb{Q}[t, x_1 \dots, x_m]$, computes a rational $T > 0$ with $H \subseteq [0, T] \times Q$, a spatial rational grid step h dividing 1, a time grid step τ dividing T and an algebraic h, τ -grid function $v : G \rightarrow \mathbb{A}$ such that $\|\mathbf{u} - \widetilde{v|_H}\|_{sL_2} < \varepsilon$ where $\varepsilon = \frac{1}{a}$. We abbreviate this problem as CP($m, n, a, \mathbb{A}, \mathbb{Q}$). The guaranteed-precision version of a restricted boundary-value problem (2) is stated in a similar way. Let $m, n \geq 2$ and $a \geq 1$ be fixed positive integers and let T be a fixed positive rational number. The version asks for an algorithm (and its complexity estimation) which, for any given matrices $A, B_1 \dots, B_m \in M_n(\mathbb{A})$, polynomials $f_1 \dots, f_n \in \mathbb{Q}[t, x_1 \dots, x_m]$ and matrices $\Phi_i^{(1)}, \Phi_i^{(2)} \in M'_n(\mathbb{A})$ meeting the conditions in (2), computes a spatial rational grid step h dividing 1, a time grid step τ dividing T and a rational h, τ -grid function $v : G \rightarrow \mathbb{Q}$ such that $\|\mathbf{u} - v|_H\|_{sL_2} < \varepsilon$ where $\varepsilon = \frac{1}{a}$ and $H = [0, T] \times [0, 1]^m$. We abbreviate this problem as BVP($m, n, a, T, \mathbb{A}, \mathbb{Q}$). In the cases when the precision $a \geq 1$ is included into the input, we will omit it in the list of parameters. Our basic result on the stated problems may be formulated as follows (see the definitions of complexity classes e.g. in [3]):

Theorem 2. *1. For any fixed integers $m, n \geq 2$, the problems CP($m, n, \mathbb{A}, \mathbb{Q}$) and BVP($m, n, T, \mathbb{A}, \mathbb{Q}$) are solvable in EXPTIME.*

2. For any fixed integers $m, n \geq 2, a \geq 1, M > 0$ and input data such that the following quantities are bounded by M : $\max_{i,j} \{ \|A^{-1}B_i A^{-1}B_j - A^{-1}B_j A^{-1}B_i\|_2, \sup_{t,x} \|\frac{\partial^2 f}{\partial x_i \partial x_j}(t, x)\|_2, \sup_x \|\frac{\partial^2 \varphi}{\partial x_i \partial x_j}(x)\|_2 \}$, $\max_i \{ \|B_i\|_2,$

$\{\|A^{-1}B_i\|_2, \|\Phi_i^{(1)}\|_2, \|\Phi_i^{(2)}\|_2\}, \|A\|_2, \frac{\lambda_{max}(A)}{\lambda_{min}(A)},$ problems $CP(m, n, a, \mathbb{A}, \mathbb{Q})$ and $BVP(m, n, a, T, \mathbb{A}, \mathbb{Q})$ are solvable in *P*TIME.

Here we provide only a short proof sketch for the problem $CP(m, n, \mathbb{A}, \mathbb{Q})$.

The first ingredients of the proof are the following upper bounds for the complexity of symbolic computations of eigenvalues and eigenvectors for some classes of matrices and matrix pencils. The computations are w.r.t. the encodings specified in Sect. 2.3. The upper bounds follow from the known bound for the root-finding in \mathbb{A} and results in Sect. 2.3.

By *spectral decomposition* of a symmetric real matrix $A \in M_n(\mathbb{R})$ we mean a pair $((\lambda_1, \dots, \lambda_n), (\mathbf{v}_1, \dots, \mathbf{v}_n))$ where $\lambda_1 \leq \dots \leq \lambda_n$ is the non-decreasing sequence of all eigenvalues of A (each eigenvalue occurs in the sequence several times, according to its multiplicity) and $\mathbf{v}_1, \dots, \mathbf{v}_n$ is a corresponding orthonormal basis of eigenvectors.

Proposition 1. (1) *For any fixed $n \geq 1$, there is a polynomial time algorithm which, given a symmetric matrix $A \in M_n(\mathbb{A})$, computes a spectral decomposition of A .*

(2) *There is a polynomial time algorithm which, given a symmetric matrix $A \in M_n(\mathbb{Q})$, computes a spectral decomposition of A uniformly on n . The same holds if we replace \mathbb{Q} by $\mathbb{Q}(\alpha)$ where α is any fixed algebraic real.*

Matrix pencil is a pair (A, B) (often written in the form $\lambda A - B$) of real non-degenerate symmetric matrices such that A is positive definite (i.e., all its eigenvalues are positive). *Spectral decomposition* of such a pencil is a tuple

$$((\lambda_1, \dots, \lambda_n), (\mathbf{v}_1, \dots, \mathbf{v}_n), (\mu_1, \dots, \mu_n), (\mathbf{w}_1, \dots, \mathbf{w}_n))$$

where $((\lambda_1, \dots, \lambda_n), (\mathbf{v}_1, \dots, \mathbf{v}_n))$ and $((\mu_1, \dots, \mu_n), (\mathbf{w}_1, \dots, \mathbf{w}_n))$ are spectral decompositions of the symmetric matrices A and D^*L^*BLD respectively, where L is the matrix formed by vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ written as columns and $D = \text{diag}\{\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, \dots, \frac{1}{\sqrt{\lambda_n}}\}$.

Proposition 2. *For any fixed $n \geq 1$, there is a polynomial time algorithm which, given a matrix pencil (A, B) with $A, B \in M_n(\mathbb{A})$, computes a spectral decomposition of (A, B) .*

The proof of Propositions 1 and 2 are based on the polynomial estimate for root-finding from Sect. 2.3 and some non-trivial facts of linear algebra [6]. These propositions imply Theorem 1 and provide important auxiliary data for the computations in the Godunov scheme used in the proof of Theorem 2.

The second ingredient of the proof of Theorem 2 is the following non-trivial fact (based on the known properties of Godunov scheme) stating that the spacial step h and the time step τ depend polynomially on the input data.

Proposition 3. *For any fixed $m, n \geq 2$ there exists a polynomial time algorithm that computes, from given $A, B_i \in M_n(\mathbb{A})$, $T \in \mathbb{Q}$, $\varphi_1, \dots, \varphi_n \in \mathbb{Q}[x_1, \dots, x_m]$, and precision $\varepsilon = \frac{1}{a} \in \mathbb{Q}$, a rational $h = \frac{1}{2^N}$ and a rational $\tau > 0$ dividing T such that the Godunov's difference scheme is stable and $\|\mathbf{u} - \mathbf{v}\|_{sL_2} < \varepsilon$.*

With these propositions at hand, it is not hard to count the computation steps in the Godunov scheme (all computations are w.r.t. the p -presentation of \mathbb{A} in Sect. 2.3). By Proposition 3, the number of grid points in the scheme (see Sect. 2.2) is bounded by a polynomial. The computations in Godunov scheme proceed bottom-up by layers, along the time axis. At the bottom level, we just evaluate the initial functions in the grid points which requires polynomial time according to remarks in Sect. 2.3. To go one level up requires, for each grid point on the next level, the values at the previous level and a fixed number of matrix multiplications by matrices, computed in advance using Propositions 1 and 2. Therefore, climbing one level up also requires polynomial time. Let p_i , $i = 1, \dots, s$ (where $s = \frac{T}{\tau}$), be a polynomial bounding the computation time for level i . Since the computation at level i uses only the values of v at grid points of level $i - 1$ (and some matrices computed in advance), the whole computation time is (essentially) bounded by the composition $p_s \circ \dots \circ p_1$ of polynomials which lays down to EXPTIME. Obtaining the PTIME complexity bounds in item 2 relies on a careful study of approximation and stability conditions of the considered difference scheme, as well as on multiple application of estimates from the uniqueness theorem proofs for (1), (2).

Our results provide apparently first upper bounds (which are exponential) for computing solutions of the Cauchy and dissipative boundary-value problems for symmetric hyperbolic systems of PDEs with guaranteed precision. An interesting and appealing problem is to modify the algorithm (or to find a new one) in order to improve the upper bound. Improving our estimate to PSPACE would be a natural step, the above algorithm does not work in PSPACE, by the remarks in Sect. 2.3 on the complexity of a “long sum”.

Although our methods do not seem to directly yield practically feasible algorithms for guaranteed precision problems for PDEs, we hope that investigations in this direction are fruitful for both theoretical research and applications. In particular, on the implementation level it seems useful and rewarding to enhance the existing systems of “exact real computations” (like iRRAM) by packages based on highly developed algorithms of computer algebra. We are not aware of the existence of such “hybrid” systems built under the slogan of “guaranteed precision numerical computations”.

References

1. Akritas, A.G.: Elements of Computer Algebra with Applications. Wiley, New York (1989)
2. Alaev, P.E., Selivanov, V.L.: Polynomial-time presentations of algebraic number fields. In: Manea, F., et al. (eds.) CiE 2018. LNCS, vol. 10936, pp. 20–29. Springer, Heidelberg (2018)
3. Balcázar, J.L., Díaz, J., Gabarró, J.: Structural Complexity I. EATCS Monographs on Theoretical Computer Science, vol. 11. Springer, Heidelberg (1988). <https://doi.org/10.1007/978-3-642-97062-7>
4. Brattka, V., Hertling, P., Weihrauch, K.: A tutorial on computable analysis. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) New Computational Paradigms, pp. 425–491. Springer, New York (2008). https://doi.org/10.1007/978-0-387-68546-5_18

5. Friedrichs, K.O.: Symmetric hyperbolic linear differential equations. *Commun. Pure Appl. Math.* **7**, 345–392 (1954)
6. Gantmacher, F.R.: *Matrix Theory*. Nauka, Moscow (1967). (in Russian)
7. Godunov, S.K. (ed.): *Numerical Solution of Higher-dimensional Problems of Gas Dynamics*. Nauka, Moscow (1976). (in Russian)
8. Kulikovskii, A.G., Pogorelov, N.V., Semenov, A.Y.: *Mathematical Aspects of Numerical Solution of Hyperbolic Systems*. Chapman & Hall/CRC Press, Boca Raton (2001)
9. Lenstra, A.K., Lenstra, H.W., Lovasz, L.: Factoring polynomials with rational coefficients. *Math. Ann.* **261**, 515–534 (1982)
10. Loos, R.: Computing in algebraic extensions. In: Buchberger, B., Collins, G.E., Loos, R. (eds.) *Computer Algebra: Symbolic and Algebraic Computations*, pp. 115–138. Springer, Vienna (1982). https://doi.org/10.1007/978-3-7091-3406-1_12
11. Pan, V., Reif, J.: The bit complexity of discrete solutions of partial differential equations: compact multigrid. *Comput. Math. Appl.* **20**(2), 9–16 (1990)
12. Schrijver, A.: *Theory of Linear and Integer Programming*. Wiley, New York (1986)
13. Selivanova, S.V., Selivanov, V.L.: Computing solution operators of symmetric hyperbolic systems of PDEs. *J. Univers. Comput. Sci.* **15**(6), 1337–1364 (2009)
14. Selivanova, S., Selivanov, V.: Computing Solution Operators of Boundary-value Problems for Some Linear Hyperbolic Systems of PDEs. *Log. Methods Comput. Sci.* **13**(4:13), 1–31 (2017). Earlier version on [arXiv:1305.2494](https://arxiv.org/abs/1305.2494) (2013)
15. Selivanova, S.V., Selivanov, V.L.: On constructive number fields and computability of solutions of PDEs. *Dokl. Math.* **477**(3), 282–285 (2017)
16. van der Waerden, B.L.: *Algebra*. Springer, Berlin (1967). <https://doi.org/10.1007/978-3-662-22183-9>
17. Weihrauch, K.: *Computable Analysis*. Springer, Berlin (2000). <https://doi.org/10.1007/978-3-642-56999-9>
18. Ziegler, M., Brattka, V.: A computable spectral theorem. In: Blanck, J., Brattka, V., Hertling, P. (eds.) *CCA 2000*. LNCS, vol. 2064, pp. 378–388. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45335-0_23
19. Jian-Ping, Z.: On the degree of extensions generated by finitely many algebraic numbers. *J. Number Theor.* **34**, 133–141 (1990)



What Is the *Concept* of Computation?

Wilfried Sieg^(✉)

Carnegie Mellon University, Pittsburgh, USA
sieg@cmu.edu

Abstract. This essay examines the two central and equivalent ways of introducing rigorous notions of *computation* as starting-points. The first way explicates *effective calculability* of number theoretic functions as the (uniform) calculability of their values in formal calculi; Gödel, Church and Kleene initially pursued this way. The other way views *mechanical procedures* as transforming finite configurations via production rules, a path Turing and Post took. For both explications, one can prove important mathematical theorems, namely, absoluteness and reducibility results. These results are of great interest, but problematic when viewed as *justifying* the adequacy of the rigorous notions. However, the theorems and their proofs reveal features that motivate the abstract concept of a *computable dynamical system*. That is, so I argue, the appropriate *structural axiomatization* of computation.

1 Introduction

In 1936, a remarkable confluence of ideas started to take shape: a confluence of ideas concerning the effective calculation of number theoretic functions and the mechanical operation on finite configurations. These ideas had emerged in mathematical and logical practice, were shaped into properly mathematical concepts and proved to be equivalent in their rigorous forms. The equivalence of the rigorous concepts has been taken as strong support for their adequacy to capture the associated informal ideas. There are two clusters of notions surrounding, on the one hand, Gödel's concept of general recursive functions whose values can be determined in (formal) logics and, on the other hand, Turing's concept of machines that manipulate finite configurations. The pioneers established important mathematical theorems concerning these two clusters: absoluteness and reducibility results were taken by them, in particular by Gödel, as "justificatory". These theorems are of great interest, but fall short of supporting the adequacy of the core concepts. And yet, they highlight crucial abstract features that motivate the formulation of *principles* for a structural axiomatic concept, namely that of a *computable dynamical system*.

This essay is dedicated to Martin Davis—mentor, friend, collaborator—on the occasion of his 90th birthday.

2 Two Gödel Mysteries: Looking Back

In 1934, Gödel gave beautiful lectures on his incompleteness theorems at Princeton. Exactly thirty years later, in a *Postscriptum* to those lectures, he drew consequences from advances that had been made in the intervening years. He formulated matters as follows:

In consequence of later advances, in particular of the fact that, due to A.M. Turing’s work, a precise and unquestionably adequate definition of the general concept of formal system can now be given, the existence of undecidable arithmetical propositions and the non-demonstrability of the consistency can now be proved for every consistent formal system containing a certain amount of finitary number theory.

So, to rephrase, the incompleteness theorems hold now, in 1964, for *all* formal theories that are consistent and contain a modicum of number theory, not just for the specific theories Gödel had considered in his 1931 paper and the 1934 lectures. What is the reason for this assertion?

According to Gödel, we have “a precise and unquestionably adequate definition of the general concept of formal system”, and it is Turing’s work that provided it:

Turing’s work gives an analysis of the concept of “mechanical procedure” (alias “algorithm” or “computation procedure” or “finite combinatorial procedure”). This concept is shown to be equivalent with that of a “Turing machine”.

Gödel also mentions Post production systems since formal systems can be defined as mechanical procedures for generating formulas.¹ Gödel’s dramatic remark on Turing is *mysterious*, as neither “analysis” nor “shown to be equivalent” is elucidated in his *Collected Works* or, as far as I know, in his *Nachlass*.

Equally mysterious is a remark Gödel made in the spring of 1934 that was reported by Church in a letter to Kleene. Having judged Church’s identification of the informal notion of *effective calculability* with λ -definability as “thoroughly unsatisfactory”, Gödel proposed to Church a way of proceeding: formulate axiomatically the generally recognized aspects of calculability and “do something on that basis”; but he hinted neither at suitable axioms nor at what might be done on their basis. The remark was later interpreted as calling for a formal theory in which Church’s Thesis could be proved as a theorem.²

The axiomatic approach I have taken is quite different; it is modeled after the way in which modern mathematics introduces *abstract notions* via structural

¹ Gödel states then that an appeal to *mechanical* procedures “is required by the concept of formal system, whose essence it is that reasoning is completely replaced by mechanical operations on formulas.”

² The most ambitious undertaking along these lines is the work of Dershowitz and Gurevich in their (2008) paper. My perspective on their mathematical work is detailed in (Sieg 2013a, 116–123).

definitions, like those for groups or fields or topological spaces. Such concepts do not express essential properties of privileged objects like numbers or geometric magnitudes; rather, they apply to classes of objects with appropriate relational connections. This transformation of 19th century mathematics is most strikingly reflected in Dedekind's *Was sind und was sollen die Zahlen?* as well as in Hilbert's *Grundlagen der Geometrie*. I will follow Dedekind's approach to the question *What are natural numbers?* For him, it does not matter what they are as individual objects as long as their system is *simply infinite*. Analogously, what computations are as individual processes does not matter as long as ...

The ellipsis will be filled in later. Now, I just recall the definition of Dedekind's concept of a *simply infinite system* and its central properties: N is a *simply infinite system* just in case there is a mapping f on N and an element 1 in N , such that (1) $f[N] \subseteq N$, (2) $N = \{1\}_0$, (3) $1 \notin f[N]$, and (4) f is injective.³ The conditions (1)–(4) are the defining *characteristics* of an *abstract, structural* concept and serve as the sole starting-points for proofs in arithmetic. Through his theorem #66, Dedekind “*established*” the existence of a particular \mathcal{N} satisfying these conditions and proved a *representation theorem*, his #132, stating that every simply infinite system is isomorphic to \mathcal{N} . The very concept is thus *categorical*, as the representation theorem immediately implies that any two simply infinite systems are isomorphic. Dedekind also proved the following fact: *any* system that stands in a 1-1 relation with \mathcal{N} can be turned into a simply infinite system. Dedekind took an enormous methodological leap when treating the concept *natural numbers* as a simply infinite system *from the same abstract perspective* as the concepts of a group or field. How can a similar leap be motivated for the concept of computation?

As a prelude, I consider a particular problem involving that very concept in a central way. The issue of *deciding by a computation*, whether a formula is provable (in *Principia Mathematica*), was in the 1920s one of the most important problems in mathematical logic. von Neumann was convinced, in 1927, that it is unsolvable. He saw its unsolvability indeed as “a *conditio sine qua non* for the contemporary practice of mathematics ... to make any sense”. However, he had to admit, “We have no clue how a proof of unsolvability would go”. The reason for his assessment was the lack of a concept of computation. The *confluence of ideas* in 1936 provided rigorous concepts. With their respective concepts in hand, Church and Turing proved the unsolvability of the decision problem for first-order logic; they also argued that their concepts were adequate to represent the informal notions. The question “*Are they adequate?*” has ever since provoked philosophical and other debates under the heading of Church's Thesis, Turing's Thesis, or the Church-Turing Thesis.

³ $\{1\}_0$, denotes the *chain of the system* $\{1\}$ with respect to f , i.e., the intersection of all systems that contain 1 as an element and are closed under f .

3 Formal Calculi: Absoluteness

The classical approach to the *effective calculability* of number theoretic functions led, mainly through Gödel and Church, to a notion of *computability in formal calculi* and to *absoluteness theorems*. Why did Gödel, Church and others work with number theoretic functions in the first place? Well, mathematicians had experience with numerical procedures defined by primitive recursion and, ever since Dedekind, knew how to solve the resulting equations. In 1931, Herbrand defined a broader class of functions that included the non-primitive recursive, but effectively calculable Ackermann function. Herbrand's ways motivated Gödel's definition of (*general*) *recursive functions* via two characteristics: (I) they are given as unique solutions of recursion equations; (II) their values are calculable from the equations by formal rules. In a manuscript from the late 1930s, Gödel analyzed the way in which calculations proceed and isolated two simple rules: they allow substitutions of numerals for variables and of equals for equals.

Gödel made also the *conceptual claim* that (I) and (II) are “exactly those [characteristics] that give the correct definition of a computable function”, i.e., the *correct definition* of computable functions *as* recursive ones. Ultimately, a *metamathematical fact* made it plausible to Gödel that recursiveness captured effective calculability. The fact is formulated in (Gödel 1936) for systems S_i of i -th order logic:

It can, . . . , be shown that a function computable in one of the systems S_i is computable already in S_1 . Thus, the notion “computable” is in a certain sense “absolute”, while almost all metamathematical notions otherwise known (for example, provable, definable, and so on) quite essentially depend upon the system adopted.

Ten years later, Gödel took absoluteness as *the* reason for the importance of recursiveness. Here we have, he claimed in his (1946), the first epistemological notion whose definition is not dependent on the chosen formalism.

Gödel's 1936 remark concerned clearly specified, particular formal systems of higher-order arithmetic. His more general 1946 absoluteness claim pertained to arbitrary formal theories that extend number theory. The latter assertion reveals a difficulty: one has to exploit for its proof (by a version of Kleene's normal form theorem) that the extending theories are *formal* ones. This difficulty is also apparent in two analogous arguments that are due to Church and Bernays, respectively. They introduced in (Church 1936) and (Hilbert and Bernays 1939, Supplement II) two closely related concepts for number-theoretic functions. Church considered the concept *computable in a logic*, whereas Bernays investigated that of *reckonable in a deductive system*. They proved the characterized function classes to be co-extensional with the class of general recursive functions. However, for these arguments to be mathematical proofs some conditions had to be imposed on Church's “logics” and Bernays' “deductive systems”. The crucial condition in either approach requires the deductive steps to be (primitive) recursive. The arguments of Gödel, Church and Bernays indeed establish *absoluteness*. But they do so only if one takes for granted or explicitly demands,

as Bernays did, that inference steps in the extending logical frameworks are (primitive) recursive.

The explication of effectively calculable functions as functions whose values can be algorithmically determined in “logical calculi” is definitely very appealing, as is their provable “absoluteness”. However, a fundamental question remains: Why should *recursiveness* conditions be imposed on calculation steps? A different approach avoids the question altogether by (1) viewing computations in a more general setting and (2) considering computation steps as *mechanical (substitution) operations* on finite, mainly syntactic, configurations. This approach will be spelled out next.

4 Production Rules: Reducibility

The classical approach to *mechanical operations* on syntactic configurations led, mainly through Turing and Post, to a notion of *computability via production systems* and to *reduction theorems*. This assertion may surprise you: the 1936 models of Turing and Post don’t look like production systems at all; however, in our paper, Davis and I argue for an underlying *conceptual confluence* that indeed involves production systems.

In the early 1920s, Post presented operations on *Principia Mathematica*’s symbolic configurations as operations on strings. He proved that operations on strings can be reduced to those of *normal (production) systems*, which have one initial word and finitely many rules of the form: $gP \rightarrow Ph$, where g and h are fixed strings, whereas P ranges over strings of the appropriate alphabet.⁴ Turing reduced in his (1936) mechanical operations on two-dimensional configurations to those of *string machines*, i.e., machines that function like Turing machines, but operate on finite strings of symbols and not just on letters. He then observed that computations of string machines are provably reducible to those of his letter machines.

These parallel reductive arguments are striking, and even more so, if one realizes — as Post did in his (1947) — that computations of Turing’s letter machines can be described via production rules. Turing machine computations are thus seen as *manipulating* sequences of symbols according to production rules. Post used this formulation to prove the unsolvability of the word-problem for semigroups. Turing extended that result in 1951 to semigroups with cancellation by refining Post’s techniques. Given the character of their reductions with quite open-ended starting-points, it is natural that Turing and Post wanted to generalize them. They were concerned with mechanical operations on broader classes of configurations, i.e., with the question, “What configurations and

⁴ In his letter to Gödel written on 30 October 1938, Post pointed out that “the whole force” of his argument for his analysis of “all finite processes of the human mind” depended on “identifying my ‘normal systems’ with any symbolic logic and my sole basis for that were the reductions I mentioned.” This letter was published in volume V of Gödel’s *Collected Works*.

substitution operations are to be considered?” The question was important to both, but neither Post nor Turing investigated it systematically.⁵

The central methodological problem for this way of ensuring the adequacy of the mathematical notion can be formulated as follows: Is there an end to finding *generalized production systems* or *machines* that operate on broader classes of configurations going beyond letters, strings, and K-graphs, yet are still reducible to Turing machines? To obtain reductions for different classes of machines, one exploits *boundedness* and *locality* conditions: (B) There is a bound on the number of symbolic configurations a machine can directly work on, *and* (L) The operations of a machine locally modify one of the configurations. The question is now: Why should *these* conditions be imposed?

The conditions actually originate from Turing’s 1936 paper. As hinted at above, Turing attempted to show that his machines can perform all operations carried out by human computers who proceed in a “machine-like” way. For that purpose, Turing strove to isolate operations that are “*so elementary that it is not easy to imagine them further divided*” and required, in particular, that human computers *immediately recognize* symbolic configurations on which they operate. His thought experiment concerning the immediate recognition of finite strings exposes sensory limitations of human computers and grounds the boundedness and locality conditions in these psychological limitations.⁶

5 Structural Definitions: Representation

In his last published paper, (Turing 1954), Turing describes and investigates a variety of *puzzles*, i.e., discrete finite configurations with particular transformation rules. He asserts that every puzzle can be easily reduced to a *substitution puzzle*, where the latter is for Turing a particular kind of Post production system.⁷ He then uses substitution puzzles to prove that there are “unsolvable problems”. Before beginning the mathematical exposition, he makes a remark that is of great methodological interest in the context of my reflections here:

This statement [on the reducibility of puzzles to substitution puzzles] is still somewhat lacking in definiteness, and will remain so. . . . The statement is moreover one which one does not attempt to prove. . . . its status is something between a theorem and a definition. In so far as we know *a priori* what is a puzzle and what is not, the statement is a theorem. In

⁵ Kolmogorov introduced operations on special kinds of graphs in his paper with Uspensky, (1963). Byrnes and I took up their work in our (1996) and generalized it to *K-graphs*. The production rules for those graphs are viewed as formalizations of generalized Turing machines operating on K-graphs.

⁶ For Post, there is a genuine philosophical difficulty, as he programmatically appeals to (subtle) properties of the human mind; see (Sieg et al. 2016, Sects. 7.6 and 7.7).

⁷ (Turing 1954, 15). I have analyzed this extremely interesting article, which was intended for a general audience, in (Sieg 2012). One straightforward observation should be mentioned. Turing discusses all the methodological issues surrounding “Turing’s Thesis”, but does not at all appeal to computing machines.

so far as we do not know what puzzles are, the statement is a definition which tells us something about what they are.

The structural axiomatic approach locates the “definition” of computable dynamical systems in just that position, as it formulates central aspects of our understanding of mechanical procedures; but it also allows us to prove a theorem, namely, the *Representation Theorem* formulated below.

An abstractive move in the ways of Dedekind yields informal *requirements for human computers*: (1) they operate on finite, but unbounded configurations; (2) they recognize, in each configuration, a unique sub-configuration or pattern from a fixed, finite list; (3) they operate on the recognized pattern; (4) they assemble the next configuration from the original one and the result of the local operation. These informal requirements can be formulated rigorously as the defining *characteristics* or axioms for discrete dynamical systems $(\mathcal{S}, \mathcal{F})$, where \mathcal{S} is its state space and consists of hereditarily finite sets and \mathcal{F} is an operation from \mathcal{S} to \mathcal{S} . Such a system is called *computable* if and only if the operation \mathcal{F} is computable in the following sense: there are finitely many patterns and local operations such that the value of \mathcal{F} , for any argument, is obtained via a finite sequence of states effected via the local operations. For the precise formulations, I refer to earlier papers of mine; see footnote 8. However, I have to mention two important facts: (1) Turing machines are computable dynamical systems, and (2) the computations of any computable dynamical system are reducible to those of Turing machines (*Representation Theorem*).

Now you may ask: How does this address the central methodological issue or, for that matter, Gödel’s mysteries? Consider the left and right part of the next diagram; they reflect that particular systems and machines fall under the abstract concepts of “computable dynamical systems” and of “Turing machines”.

*Concept of computable
dynamical systems*

Concept of Turing machines

↓

↓

*Particular computable
dynamical system*

←→

Particular Turing machine

The facts (1) and (2) I formulated in the last paragraph allow us to draw the bottom part of the diagram; I will come back to it when “resolving” Gödel’s mysteries.

The guiding ideas of this analysis of human computations can be used to investigate *machine computations* as Gandy did in his 1980-paper *Church’s Thesis and Principles for Mechanisms*. He viewed machines informally as “discrete mechanical devices” that can operate in *parallel* and considered cellular automata as paradigmatic machines. In general, machines satisfy boundedness and locality conditions on account of two physical facts, (i) a lower bound on the size of distinguishable atomic components, and (ii) an upper bound on the propagation of signals. Extending Gandy’s work, I introduced *computable dynamical*

P-system to capture parallel computations of machines. The restrictive conditions together with a complex *re-assembly operation* allow us to reduce computations of any P-system to those of Turing machines, yielding again a *Representation Theorem*.⁸ As Turing machines are computable P-systems, we can complete the above “conceptual diagram” for P-systems as well. At the end of Sects. 3 and 4, I asked critical and pointed *specific* questions. Here I ask just *one* critical and pointed *general* question: *Why should we proceed in this structural axiomatic way?*

6 Connections: Looking Ahead

Let me answer the single general question in a number of different ways. There is, first of all, the *broad methodological issue* surrounding Church’s and Turing’s Theses. The above diagram reflects my answer to the questions I raised after quoting Gödel’s 1964-remarks about Turing: the extension of Turing’s and Post’s analysis of “mechanical procedure” has led to the abstract concept of a computable dynamical system, and that concept has been shown “to be equivalent with that of a Turing machine”. It clarifies not only this *first* Gödel mystery, but also the *second* one: after all, axioms were formulated and “something” concretely mathematical was done on their basis (proofs of representation theorems).

Beyond the importance for this very traditional question, the work can be used, secondly, to answer a significant methodological question concerning parallelism. The Gandy inspired analysis of parallel computations gives a mathematical form to the informal assumption(s) underlying almost all models of concurrent systems or distributed computing. In (Lampert and Lynch 1990, 1166) one finds this description of the issue:

Underlying almost all models of concurrent systems is the assumption that an execution consists of a set of discrete events, each affecting only part of the system’s state. Events are grouped into processes, each process being a more or less completely sequenced set of events sharing some common locality in terms of what part of the state they affect. For a collection of autonomous processes to act as a coherent system, the processes must be synchronized.

The analysis of parallel computation is conceptually convincing and gives a sharp mathematical form to the informal assumptions “underlying almost all models of concurrent systems”. Szabo explores in his (Szabo 2017, Chap. 4) standard models of parallel computing and shows that they are computable dynamical P-systems.

Dynamical P-systems had been applied earlier in a quite different context. As is well-known, production systems have been used to model mental processes,

⁸ The mathematical details are found in my (2002) and (2009), where the earlier paper is Sect. 5.

especially at Carnegie Mellon in the footsteps of Newell and Simon. Other cognitive scientists, for example Jay McClelland, have been using for that purpose *parallel distributed systems or artificial neural nets*. The commonly used neural nets were shown in (DePisapia 2000) to be computable dynamical P-systems.⁹ Their reducibility to Turing machine computations shows that the ideological controversy between the appropriateness of symbolic or parallel distributed processes for modeling aspects of cognition is resolved by focusing on levels of descriptions. This is the third way of answering the above general question.

Finally and fourthly, taking my structural axiomatic approach there is no question of *proving* Turing's Thesis. There is only one, albeit basic question: do the characteristic conditions of computable dynamical systems adequately capture the "generally recognized aspects of calculability"? I think they do, as they are *obtained by abstraction* from those of production systems in the tradition of Post and Turing. The representation theorem brings out the central role of Turing's machines and Post's systems for the theory of computability. Focusing on this mathematical core does not lead us to an unusual thesis. Rather, we find ourselves in harmony with the reflective practice of mathematics made possible by the axiomatic method. The real foundational question concerns the limitations of computing devices and their justification: (i) for human computers via psychological laws and (ii) for machines via physical laws.

Let me mention two deeply connected research directions that are programmatically related not only to the main topic of this essay and to concerns of the pioneers,¹⁰ but also to the computational modeling of cognitive processes I just pointed to. Is there a prospect of modeling *mathematical reasoning*? Given the remarkable technological tools at our disposal, we can take a modest step and face the practical issue of formally verifying mathematical theorems — in *humanly intelligible ways*. Such *natural formalization* requires that we structure proofs conceptually and uncover heuristic strategies used in mathematics for finding proofs. This *natural formalization program* leads directly to a quest for automated proof search. When focused on procedures and results that are humanly intelligible, it presents a deep challenge to, but also a remarkable opportunity for, cognitive science: *we do need a better understanding of the sophisticated mathematical mind*.¹¹ Such a deeper understanding can be obtained only by a radically interdisciplinary approach that involves logicians, mathematicians, computer scientists, and cognitive psychologists.

Remark: Versions of this essay have been presented to a number of audiences during the last three years: in November of 2014 at the Philosophy Colloquium of Carnegie Mellon, as the Williams Lecture at the University of Pennsylvania, the Philosophy Colloquium at Ohio State University, at Waseda University in Tokyo, at the Sorbonne in Paris, and most recently at the Institut für Informatik

⁹ E.g., artificial neural nets with a variety of learning algorithms, but also with back propagation.

¹⁰ These concerns, with respect to Gödel and Turing, are exposed in my (2013b).

¹¹ In quite different ways, this goal has been pursued by (Ganesalingam and Gowers 2017) and (Sieg and Walsh 2017).

at the University of Bern, the Polaris Colloquium of the University of Lille and the Mathematics Department at the University of Notre Dame in November of 2017.

References

- Church, A.: An unsolvable problem of elementary number theory. *Am. J. Math.* **58**, 345–363 (1936)
- Davis, M., Sieg, W.: Conceptual confluence in 1936: Post and Turing. In: Sommaruga, G., Strahm, T. (eds.) *Turing’s Revolution*, pp. 3–27. Birkhäuser, Basel (2015)
- Dedekind, R.: *Was sind und was sollen die Zahlen?*. Vieweg, Braunschweig (1888)
- DePisapia, N.: Gandy machines: an abstract model of parallel computation for Turing machines, the game of life, and artificial neural networks. Master’s thesis, Carnegie Mellon University (2000)
- Dershowitz, N., Gurevich, Y.: A natural axiomatization of computability and proof of Church’s thesis. *Bull. Symb. Log.* **14**, 299–350 (2008)
- Gandy, R.: Church’s thesis and principles for mechanisms. In: Barwise, J., Keisler, H., Kunen, K. (eds.) *The Kleene Symposium*, pp. 123–148. North-Holland Publishing Company, Amsterdam (1980)
- Gandy, R.: The confluence of ideas in 1936. In: Herken, R. (ed.) *The Universal Turing Machine - A Half-Century Survey*, pp. 55–111. Oxford University Press (1988)
- Ganesalingam, M., Gowers, W.T.: A fully automatic problem solver with human-style output (2013). [arXiv:1309.4501](https://arxiv.org/abs/1309.4501)
- Ganesalingam, M., Gowers, W.T.: A fully automatic problem solver with human-style output. *J. Autom. Reason.* **58**(2), 253–291 (2017)
- Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I (1931). *Trans. in Gödel*, pp. 144–195 (1986)
- Gödel, K.: On undecidable propositions of formal mathematical systems (1934). *Gödel*, pp. 346–371 (1986)
- Gödel, K.: Über die Länge von Beweisen (1936). *Gödel*, pp. 396–399 (1986)
- Gödel, K.: Undecidable Diophantine propositions (1939). *Gödel*, pp. 164–175 (1995)
- Gödel, K.: Remarks before the Princeton Bicentennial Conference on problems in mathematics (1990). *Gödel*, pp. 150–153 (1946)
- Gödel, K.: *Collected Works I*. Oxford University Press, Oxford (1986)
- Gödel, K.: *Collected Works II*. Oxford University Press, Oxford (1990)
- Gödel, K.: *Collected Works III*. Oxford University Press, Oxford (1995)
- Gödel, K.: *Collected Works V*. Oxford University Press, Oxford (2003)
- Herbrand, J.: On the consistency of arithmetic (1931). *Herbrand*, pp. 282–298 (1971)
- Herbrand, J.: *Logical Writings*. Harvard University Press, Cambridge (1971)
- Hilbert, D.: *Grundlagen der Geometrie*. Teubner, Leipzig (1899)
- Hilbert, D., Bernays, P.: *Grundlagen der Mathematik II*. Springer, Heidelberg (1939). <https://doi.org/10.1007/978-3-642-86896-2>
- Kolmogorov, A., Uspensky, V.: On the definition of an algorithm. *AMS Trans.* **21**(2), 217–245 (1963)
- Lampart, L., Lynch, N.: Distributed computing: models and methods. In: van Leeuwen, J. (ed.) *Handbook of Theoretical Computer Science*. Elsevier, Groningen (1990)
- McClelland, J.: A PDP approach to mathematical cognition (2015). <https://stanford.edu/~jlmcc/PDPMathCogLecture.html>
- Post, E.: Finite combinatory processes—formulation I. *J. Symb. Log.* **1**, 103–105 (1936)

- Post, E.: Recursive unsolvability of a problem of Thue. *J. Symb. Log.* **12**, 1–11 (1947)
- Sieg, W.: Calculations by man and machine: mathematical presentation. In: Gärdenfors, P., Wolenski, J., Kijania-Placek, K. (eds.) *In the Scope of Logic, Methodology and Philosophy of Science*. Synthese Library, vol. 315, pp. 247–262. Kluwer (2002)
- Sieg, W.: On computability. In: Irvine, A. (ed.) *Philosophy of Mathematics*. Elsevier, New York City (2009)
- Sieg, W.: Normal forms for puzzles: a variant of Turing’s thesis. In: Cooper, S., van Leeuwen, J. (eds.) *Alan Turing: His Work and Impact*, pp. 332–339. Elsevier (2012)
- Sieg, W.: Axioms for computability: do they allow a proof of Church’s Thesis?. In: Zenil, H. (ed.) *Computable Universe - Understanding and Exploring Nature as Computation*, pp. 99–123. World Scientific (2013a)
- Sieg, W.: Gödel’s challenge (to Turing). In: Copeland, B., Posy, C., Shagrir, O. (eds.) *Computability - Turing, Gödel, Church, and Beyond*, pp. 183–202. The MIT Press (2013b)
- Sieg, W., Byrnes, J.: K-graph machines: generalizing Turing’s machines and arguments. In: Hajek, P. (ed.) *Gödel 1996. Lecture Notes in Logic 6*, pp. 98–119. Springer, Heidelberg (1996)
- Sieg, W., Szabo, M., McLaughlin, D.: Why Post did [not] have Turing’s thesis. In: Omodeo, E., Policriti, A. (eds.) *Martin Davis on Computability, Computational Logic, and Mathematical Foundations*, pp. 175–208. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-41842-1_7
- Sieg, W., Walsh, P.: Natural formalization: deriving the Cantor-Bernstein theorem in ZF (2017). Manuscript
- Szabo, M.: Human and machine computation: an exploration. Ph.D. thesis, Carnegie Mellon University (2017)
- Turing, A.: On computable numbers with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **42**, 230–265 (1936)
- Turing, A.: Solvable and unsolvable problems. *Sci. News* **31**, 7–23 (1954)
- von Neumann, J.: Zur Hilbertschen Beweistheorie. *Math. Z.* **26**, 1–46 (1927)



Witness Hiding Without Extractors or Simulators

André Souto^{1,4,5(✉)}, Luís Antunes^{3,6}, Paulo Mateus^{2,4},
and Andreia Teixeira^{6,7,8}

¹ DI - FCUL, Lisboa, Portugal
ansouto@fc.ul.pt

² IST - UL, Lisboa, Portugal

³ DI - FCUP, Porto, Portugal

⁴ LASIGE, Lisboa, Portugal

⁵ Instituto de Telecomunicações, Lisboa, Portugal

⁶ INESC-TEC, Porto, Portugal

⁷ Cintesis, Porto, Portugal

⁸ ARC4DigiT, Viana do Castelo, Portugal

Abstract. In a witness hiding protocol the prover tries to convince the verifier that he knows a witness to an instance of an **NP** problem without revealing the witness. We propose a new look at witness hiding based on the information conveyed in each particular instance of the protocol.

We introduce the concept of *individual witness hiding* (IWH) and prove that zero-knowledge protocols for classical problems like **HAM** are not IWH. On the other hand, we show that all **FewP** problems have an IWH protocol. Finally, by introducing a *Kolmogorov string commitment* protocol we can show that all **FewP** problems have an IWH protocol that is zero-knowledge relative to an oracle.

1 Introduction

In [9], Goldwasser et al. introduced the concept of interactive proof systems and knowledge complexity giving particular importance to Zero Knowledge (ZK) proofs. The notions of witness indistinguishability and witness hiding were introduced in [4], aiming to relax the conditions of zero knowledge. Rather than not revealing any information, we just want to “hide” the witnesses used during the protocol.

The definitions of zero-knowledge and witness-hiding are based on the existence of simulators or witness extractors that an adversary can use [2]. This leads

A. Teixeira—Work was funded by PEst-OE/EEI/LA0008/2013 of Instituto de Telecomunicações and LASIGE, ref. UID/CEC/00408/2013 and Confident project PTDC/EEI-CTP/4503/2014. A.T thanks the scholarship 6585/BPD B3-A/2018 within project “NanoSTIMA” ref. NORTE-01-0145-FEDER-000016 under PORTUGAL 2020 and ERDF. L.A acknowledges Digi-NewB project funded from the European Unions Horizon 2020 research and innovation programme under grant agreement No 689260.

to problems in the context of tamper-proof devices [12] or in contexts where parallel repetitions are important (for which ZK protocols are not applicable). The issue in both cases is in the definition itself rather than in the amount of information leaked. In fact, the definition does not allow one to quantify the information leakage. Perhaps there is a more fundamental way to define zero-knowledge, and the existence of a simulator would be just a device to prove it (see Protocol 4). Another concern is that the existence of a simulator might be too weak to guarantee that no information whatsoever is leaked to the verifier in all instances of the protocol. In many known protocols, a small number of executions of the protocol leak a substantial amount of information about the witness. Can the prover identify these weak executions of the protocol beforehand?

These questions motivate us to propose a new approach based on Kolmogorov complexity [11], a rigorous measure of the amount of information contained in an individual object, usually a string, as the size of the smallest program that generates that object. The definition we propose of individual witness hiding is closer to the intuitive idea of zero-knowledge and does not depend on the existence of simulators. A protocol is individual witness hiding if the advantage to the verifier to retrieve the secret (for all possible exchanges of information during the interaction with the prover) is at most a logarithmic term when compared with the time-bounded Kolmogorov complexity of extracting the secret without interaction (Definition 7). We show that the well-known zero-knowledge (ZK) protocol for the Quadratic residues problem (QR) described in [9] is not individual witness hiding according to our proposed definition. By identifying the transcripts that leak a lot of information about the witnesses we are able to design an individual witness-hiding protocol for QR, for FHAM, a version of HAM the Hamiltonian path problem and, in fact for all problems in **FewP**. Furthermore, we address the problem of designing individual witness-hiding protocols that are also zero-knowledge in the random oracle model.

2 Preliminaries

We assume the reader is familiar with standard notation on Kolmogorov complexity and computational complexity (namely **NP**, **FewP**). We say that a function v is *negligible* if for any polynomial p , and sufficiently large n , $v(n) \leq 1/p(n)$. For any $L \in \mathbf{NP}$ there exists a polynomial time testable relationship R such that $R(x, w) = 1$ if and only if w is a witness for $x \in L$. We denote by W^L the set of witnesses for L and define $W_x^L = \{w : (x, w) \in W\}$ as the set of witnesses for $x \in L$. A reduction f between L_1 and L_2 is *parsimonious* if for all $x \in L_1$, $|W_x^{L_1}| = |W_{f(x)}^{L_2}|$, i.e., the reduction also preserves the number of witnesses.

2.1 Algorithmic Information

We present the basic definitions and results needed in this work. We use prefix-free Kolmogorov complexity.

Definition 1 (Kolmogorov complexity). Let U be a fixed prefix-free universal Turing machine. For any strings $x, y \in \Sigma^*$, the *Kolmogorov complexity* of x given y with oracle access to f is defined by $\mathbf{K}_f(x|y) = \min_p\{|p| : U_f(p, y) = x\}$.

For any time-constructible t , the *t -time-bounded Kolmogorov complexity* of x given y with oracle access to f is defined by $\mathbf{K}_f^t(x|y) = \min_p\{|p| : U_f(p, y) = x \text{ in at most } t(|x| + |y|) \text{ steps}\}$.

The common values for y and f are the empty string and the empty oracle and are omitted. The incompressibility theorem states that for every n , fixed constant r , and fixed finite string z , the set $\{x : x \in \Sigma^n \wedge \mathbf{K}(x|z) \leq n + \mathbf{K}(\langle n \rangle|z) - r\}$ has at most $2^{n-r+O(1)}$ elements, where $\langle n \rangle$ is the binary representation of n .

Theorem 1 (Symmetry of information). For all x, y and z we have, $\mathbf{K}(x, y|z) = \mathbf{K}(x|z) + \mathbf{K}(y|x, z) + O(\log \min\{\mathbf{K}(x|z), \mathbf{K}(y|z)\})$ and $\mathbf{K}(x, y) = \mathbf{K}(x) + \mathbf{K}(y|x, \mathbf{K}(x)) + O(1)$.

Theorem 2. Let m be some polynomial and $x, y \in \Sigma^{m(n)}$ be fixed strings. For any sufficiently large constant e , there is a constant d and at least $2^n - 2^{n-d}$ strings $z \in \Sigma^n$ such that $\mathbf{K}(x|y, z) \geq \mathbf{K}(x|y) - e$.

Clearly, the same theorem is true if we consider time-bounded Kolmogorov complexity. In fact, the same strings satisfy $\mathbf{K}^t(x|y, z) \geq \mathbf{K}^t(x) - O(\log n)$.

2.2 Interactive Proof Systems

We suggest [4, 5] for more details on this topic. We opt to use $\ll \mathcal{P}, \mathcal{V} \gg(x)$ to represent the random variable of \mathcal{V} 's view of the interaction between \mathcal{P} and \mathcal{V} on input x .

Definition 2 (Knowledge complexity [7]). Fix $k : \mathbb{N} \rightarrow \mathbb{N}$. An interactive proof system $(\mathcal{P}, \mathcal{V})$ for a language L can be simulated using a hint of length k if for every probabilistic polynomial time verifier \mathcal{V}' there is a probabilistic polynomial time Turing machine $\mathcal{S}_{\mathcal{V}'}$, called simulator, such that for every $x \in L$ there is a string $h(x)$ of length at most $k(|x|)$, the knowledge complexity, such that $\{\mathcal{S}_{\mathcal{V}'}(x, h(x))\}_{x \in L}$ and $\{\ll \mathcal{P}, \mathcal{V}' \gg(x)\}_{x \in L}$ are statistically close.

An interactive proof system is statistical (resp. computational) zero knowledge if k , in the above definition, can be the null function. Hence we say that a protocol $(\mathcal{P}, \mathcal{V})$ is statistical (resp. computational) zero knowledge if there is a probabilistic polynomial time Turing machine that can statistically (resp. computationally) simulate the interaction between \mathcal{P} and \mathcal{V} .

We are interested in analyzing the amount of information about a witness of the statement " $x \in L$ " that is leaked during the interaction. Two related notions have been introduced by Feige and Shamir [4]: witness indistinguishability and witness hiding. These concepts were developed as a tool for cryptographic identification schemes in which the requirement of zero knowledge is too strong. A protocol is witness hiding if, at the end of the protocol, \mathcal{V}' is not able to compute any new witness as a result of the interaction.

Definition 3 (Witness indistinguishability [4]). Let $L \in \mathbf{NP}$, W^L a witness relation for L and $(\mathcal{P}, \mathcal{V})$ be a proof of knowledge system for L . We say that $(\mathcal{P}, \mathcal{V})$ is a witness indistinguishable (WI) for the relation W^L if for any \mathcal{V}' , any large enough common input x , any two witnesses w_1, w_2 of W_x^L , and for any auxiliary input z for \mathcal{V}' , the ensembles, $\mathcal{V}'_{\mathcal{P}(x, w_1)}(x, z)$ and $\mathcal{V}'_{\mathcal{P}(x, w_2)}(x, z)$, corresponding to the views of \mathcal{V}' of the protocol with the two witnesses respectively, are computationally indistinguishable.

Notice that unlike the definition of zero knowledge, the definition for witness indistinguishability involves no simulator. Also, as shown in [4] any zero knowledge protocol is a witness indistinguishable protocol.

Definition 4 (Generator [4]). Let $L \in \mathbf{NP}$ and W^L a polynomial time testable relation for L . G is a generator of W^L if on input 1^n it produces instances $(z, w) \in W^L$ such that $|z| = n$ and $|w| \leq p(n)$ for some polynomial p .

Definition 5 (Witness hiding [4]). Let $L \in \mathbf{NP}$ and W^L a polynomial time testable relation for L . Let $(\mathcal{P}, \mathcal{V})$ be a proof of knowledge for L , and let G be a generator for W^L . $(\mathcal{P}, \mathcal{V})$ is witness hiding on (W^L, G) if there is a witness extractor M running in expected polynomial time, such that for any nonuniform polynomial time \mathcal{V}' , $\Pr[\mathcal{V}'_{\mathcal{P}(x, w)}(x) \in W_x^L] < \Pr[M(x; \mathcal{V}', G) \in W_x^L] + v(n)$ where $G(1^n) = (x, w')$ for some w' , v is some negligible function.

If each input has at least two “computationally independent” witnesses i.e., two witnesses that are not possible to relate in polynomial time, then any witness indistinguishability protocol induces a witness hiding protocol [4].

3 Individual Witness Hiding

We propose the notion of individual witness hiding protocol based on algorithmic information. A similar flavored definition to the one proposed in this paper has been considered in Information complexity (see for example [10, 13]).

Intuitively, an interactive proof system $(\mathcal{P}, \mathcal{V})$ is individual witness hiding if for every (malicious) verifier \mathcal{V}' , and every possible transcript c of an interaction between \mathcal{P} and \mathcal{V}' , the algorithmic information of a witness w , given the validity of “ $x \in L$ ” is essentially the same as the algorithmic information of w given both x and c , i.e., there is no communication c leading to a significant compression of w . In order to be able to measure the gain of information by \mathcal{V} concerning the witnesses, while interacting with \mathcal{P} , we only consider the part $c_{\mathcal{P}\mathcal{V}}$ of the communication coming from \mathcal{P} to \mathcal{V} . This way, we also avoid the problem of addressing the (very unlikely) case where \mathcal{V}' randomly generates a witness w .

To define rigorously the notion of individual witness hiding we need to introduce the concept of individual knowledge complexity.

Definition 6 (Individual knowledge complexity). Let L be a language, S_x be a family of strings (for example a set of witnesses for x) and let $(\mathcal{P}, \mathcal{V})$ be an

interactive proof system (IPS) for L . Consider $c \in C_x$, where C_x is the set of all possible communications between \mathcal{P} and \mathcal{V} on x and denote by $c_{\mathcal{P}\mathcal{V}}$ the part of the communication from \mathcal{P} to \mathcal{V} . Moreover, consider t to be any time bound.

1. If $x \in L$, the *knowledge complexity* gained about $w \in S_x$ on transcript c in time t is $\mathbf{KC}_{\mathcal{P},\mathcal{V}}^t(w; x, c) = \mathbf{K}^t(w|x) - \mathbf{K}^t(w|x, c_{\mathcal{P}\mathcal{V}})$.
2. The amount of *individual knowledge complexity* gained by interaction on x about $w \in S_x$ is $\mathbf{KC}_{\mathcal{P},\mathcal{V}}^t(w; x) = \max_{c \in C_x} \mathbf{KC}_{\mathcal{P},\mathcal{V}}^t(w; x, c)$.

We show that individual knowledge complexity has some relevant properties, which follow from the properties of the time-bounded Kolmogorov complexity.

Proposition 1. *Let $L \in \mathbf{NP}$ and $W_x^L = \{w : (x, w) \in W\}$ the set of witnesses for input x and $(\mathcal{P}, \mathcal{V})$ be an (χ, σ) -interactive proof system for L . Consider $c \in C_x$, where C_x is the set of all possible communications between \mathcal{P} and \mathcal{V} on x and denote by $c_{\mathcal{P}\mathcal{V}}$ the part of the communication from \mathcal{P} to \mathcal{V} . Finally, let t be the any time constructible function. Then, we have:*

1. $\mathbf{KC}_{\mathcal{P},\mathcal{V}}^t(w; x) \geq 0$;
2. $\mathbf{KC}_{\mathcal{P},\mathcal{V}}^t(w; x) \leq \max_{c \in C} K^t(c)$;
3. $\mathbf{KC}_{\mathcal{P},\mathcal{V}}^t(w; x) \leq \mathbf{K}^t(w|x) \leq \max_{w' \in W_x} \mathbf{K}^t(w'|x)$;

Definition 7 (Individual witness hiding proof). Let L be a language, $S = \{S_x\}$ be a family of sets of strings and $(\mathcal{P}, \mathcal{V})$ an interactive proof system for L . We say that $(\mathcal{P}, \mathcal{V})$ is an (χ, σ) -individual S hiding proof for L if for every possible probabilistic polynomial-time verifier \mathcal{V}' and for every polynomial t , for all sufficiently large $x \in L$, and every $w \in S_x$, $\mathbf{KC}_{\mathcal{P},\mathcal{V}'}^t(w; x) \leq O(\log |x|)$.

If $S = \{W_x : x \in L\}$ where W_x is the set of witnesses for $x \in L$ and $L \in \mathbf{NP}$ then we say that $(\mathcal{P}, \mathcal{V})$ is an *individual witness hiding proof* for L .

We illustrate our approach by considering, for example, the Quadratic Residues problem ($\mathbf{QR} \in \mathbf{FwP}$), i.e., the problem of determining, given $n = pq$ for unknown primes p and q and x , whether there is w such that $w^2 = x \pmod n$.

Protocol 1 (Zero-knowledge protocol for \mathbf{QR} (see [8]))

Input: An $n = pq$, where p and q are unknown primes and $x \in \mathbb{Z}_n^*$.

Secret: If $x \in \mathbf{QR}(n)$, the secret is u such that $x = u^2 \pmod n$ (unknown to \mathcal{V}).

The following steps are repeated $\ell = \log n$ times.

1. \mathcal{P} chooses a random $v \in \mathbb{Z}_n^*$, and computes $y = v^2 \pmod n$. \mathcal{P} sends y to \mathcal{V} .
2. \mathcal{V} chooses a random $i \in \{0, 1\}$ and challenges \mathcal{P} to see the square root of $x^i y$.
3. \mathcal{P} computes $z = u^i v \pmod n$ and sends it to \mathcal{V} .
4. \mathcal{V} checks if $z^2 = x^i y \pmod n$.

If all rounds are completed successfully, \mathcal{V} accepts $x \in \mathbf{QR}(n)$. Otherwise reject.

Theorem 3. *If $\mathbf{QR} \notin \mathbf{P}$ then Protocol 1 is not individual witness hiding for \mathbf{QR} .*

Proof. Let c be the unlikely, but possible, communication where $y = 1$. When \mathcal{V} challenges \mathcal{P} on $i = 1$, then \mathcal{P} must answer u to fulfill \mathcal{V} 's challenge, i.e., c contains u . Hence, $\mathbf{K}^t(u|(x, n), c) \leq d$ for some constant d and then we would have for all $x \in \mathbb{QR}(n)$ and for all $u \in W_x$, $\mathbf{KC}_{\mathcal{P}, \mathcal{V}}^t((x, n), u) = \max_{c \in C} (\mathbf{K}^t(u|x, n) - \mathbf{K}^t(u|x, n, c_{\mathcal{P}\mathcal{V}})) \geq \mathbf{K}^t(u|x, n) - d$. Under the individual witness hiding assumption, we would have $\forall x \in \mathbb{QR}(n)$, that there is a square root $u \in S_x$ of x such that, $\mathbf{K}^t(u|x, n) \leq O(\log |(x, n)|)$. By searching through all polynomially many candidates to u (given by programs of length at most $O(\log |(x, n)|)$), this would yield a polynomial-time algorithm for \mathbb{QR} .

In order to propose an individual witness hiding protocol for \mathbb{QR} , we improve Protocol 1 by imposing restrictions on the values v that the prover can send in the first step.

Protocol 2 (Individual witness hiding protocol for \mathbb{QR} (see [14]))

Input: An $n = pq$, where p and q are prime and $x \in \mathbb{QR}(n)$.

Secret: $S_x = \{u_1, u_2, u_3, u_4\}$ such that $x = u_i^2 \pmod n$.

Let m be a parameter of security. The following steps are repeated m times. The round j consists of:

1. (Commitment) At round j , \mathcal{P} chooses $v_j \in \mathbb{Z}_n^*$ such that:
 - (a) For $i = 1, \dots, 4$, $\mathbf{K}^t(u_i|z_1, \dots, z_{j-1}, v_j) \geq \mathbf{K}^t(u_i|z_1, \dots, z_{j-1}) - O(1)$;
 - (b) For $i = 1, \dots, 4$, $\mathbf{K}^t(u_i|z_1, \dots, z_{j-1}, uv_j) \geq \mathbf{K}^t(u_i|z_1, \dots, z_{j-1}) - O(1)$; \mathcal{P} computes $y = v_j^2 \pmod n$. \mathcal{P} sends y to \mathcal{V} .
2. \mathcal{V} chooses a random $i \in \{0, 1\}$ and challenges \mathcal{P} to see the square root of $x^i y_j$.
3. (Revealing) \mathcal{P} computes $z_j = u^i v_j \pmod n$ and sends it to \mathcal{V} .
4. \mathcal{V} checks if $z_j^2 = x^i y \pmod n$.

If all rounds are completed with success, \mathcal{V} accepts that $x \in \mathbb{QR}(n)$.

Theorem 4. If m , the number of rounds played in Protocol 2 is logarithmic in the size of (n, x) then the Protocol is a $(0, \text{poly})$ -individual witness hiding. (see the proof in the Appendix).

4 Individual Witness Hiding Protocol for FewP

In this section we discuss a possible protocol that is individual witness hiding for any language in **FewP**. Before continuing observe that:

1. Theorem 4 does not have a similar counterpart for general **NP** problems. In fact, the counting argument only works in the case of \mathbb{QR} since the number of witnesses is finite. In general, given a problem H in **NP** the size of W_H is unknown.
2. In contrast to classical zero-knowledge, if we are not interested in simulating the communication between \mathcal{P} and \mathcal{V} , we can design an individual witness hiding protocol based on the assumption that $\text{HAM} \notin \mathbf{P}$.

Herein we use the \mathbb{HAM} problem to setup individual witness hiding for **FewP**. Assume that one-way functions exist and, consequently, it is possible to construct a bit commitment scheme.

Protocol 3 (Zero-knowledge protocol for \mathbb{HAM} (see [5])

Input: A graph H with set of vertices $V = \{1, \dots, n\}$.

Secret of \mathcal{P} : A Hamiltonian path c in H .

The following steps are repeated n^2 times.

1. \mathcal{P} chooses a random permutation π and sets $H' = \pi(H)$. For each $(i, j) \in V \times V$, \mathcal{P} runs the commitment phase of a bit commitment scheme to commit to “ (i, j) is an edge/is not an edge in H' ” and sends them to \mathcal{V} .
2. \mathcal{V} chooses randomly one bit in $\{0, 1\}$ and sends it to \mathcal{P} challenging him to show the isomorphism between the graph he got and H in the first case, and to show the Hamiltonian path in H' on the latter one.
3. If \mathcal{P} receives 0, it reveals the permutation π and all the keys to the commitment. If it receives 1, it reveals the keys of the commitment containing the Hamiltonian path in H' .
4. \mathcal{V} checks the validity of the things sent by \mathcal{P} .

The idea is to consider a subset of problems of \mathbb{HAM} with the promise that if $H \in \mathbb{HAM}$ then the number of Hamiltonian paths are polynomially bounded. We consider, fixed a family of polynomials p , the language $\cup_p \mathbb{FHAM}_p$ that is **FewP** hard and any language L in **FewP** is parsimoniously reducible to some \mathbb{FHAM}_p . For each instance of \mathbb{FHAM}_p , we can prove that it is possible to hide all the witnesses, as there are at most polynomially many of them. Notice that in the individual witness hiding setting, we are not concerned with the construction of a simulator, we can design a protocol that does not use the assumption of the existence of one-way functions.

Definition 8. Given a polynomial p , promised problem *Few Hamiltonian Path Problem* (\mathbb{FHAM}_p) is defined as follows: Given a graph H is H Hamiltonian? The promise is that if there is one Hamiltonian path then $0 < |W_H| \leq p(|H|)$.

Clearly \mathbb{HAM} is **FewP** hard and for every p , $\mathbb{FHAM}_p \subset \mathbb{HAM} \in \mathbf{NP}$.

Protocol 4 (Individual witness hiding for \mathbb{FHAM}_p)

Input: Graph H with n vertices and with at most poly p many Hamiltonian paths.

Secret of \mathcal{P} : $S_H = \{c_1, \dots, c_{p(n)}\}$, the set of all Hamiltonian path in H . Assume that z_i is the information revealed at round i . Let m be a fixed parameter of security. The following steps are repeated m times.

1. At round j , \mathcal{P} picks a K -random permutation π_j and sets $H' = \pi_j(H)$ s. t.:
 - (a) $\forall i=1, \dots, p(n), \mathbf{K}^t(c_i|H, z_1, \dots, z_{j-1}, \pi_j(c_1)) \geq \mathbf{K}^t(c_i|H, z_1, \dots, z_{j-1}) - O(1)$;
 - (b) $\forall i=1, \dots, p(n), \mathbf{K}^t(c_i|H, z_1, \dots, z_{j-1}, \pi_j) \geq \mathbf{K}^t(c_i|H, z_1, \dots, z_{j-1}) - O(1)$; \mathcal{P} sends H' to \mathcal{V} .
2. \mathcal{V} chooses $b \in \{0, 1\}$, and if $b = 0$ he asks \mathcal{P} to see the isomorphism between the graph he received and H . Otherwise he asks for Hamiltonian path in H' .

3. \mathcal{P} provides π if he received 0 and $\pi(c)$ if he received a 1.
4. \mathcal{V} checks whether the messages sent by \mathcal{P} are sound.

It seems that the previous protocol is not classical zero-knowledge, since we are not able to build a simulator capable of mimicking the communication.

Theorem 5. *If m is a logarithmic function on the $|H|$, then the Protocol 4 is $(0, \text{poly})$ -individual witness hiding for FHAM_p for some polynomial p .*

We provide the proof of this theorem in the Appendix.

Note that using a parsimonious reduction to HAM (in fact FHAM_p for some poly p) we can design an IWH protocol for all $L \in \mathbf{FewP}$. Notice that, if some information about a witness $y \in W_x$ for some $x \in L$ was revealed during the protocol than the Protocol 4 would not be witness hiding for FHAM since we could apply the reduction to get information about a path in the reduced graph. From this discussion, we conclude the following result.

Theorem 6. *Any \mathbf{FewP} problem admits a witness hiding protocol.*

5 A Variant of IWH that Is Classical ZK

Classical zero-knowledge proofs can be defined relative to auxiliary inputs, where the verifier has access to some string used during the protocol.

Definition 9 (Zero-knowledge relative to oracles (see for instance [12])). Let $R(x, w)$ be a predicate relative to a statement x and a witness w , $\mathcal{O}_{\mathcal{P}}, \mathcal{O}_{\mathcal{V}}; \mathcal{O}'_{\mathcal{V}}$ be three lists of oracles with access to the same random coins r_I .

An *argument of knowledge for R relative to $(\mathcal{O}_{\mathcal{P}}, \mathcal{O}_{\mathcal{V}})$* is a pair $(\mathcal{P}_{\mathcal{O}_{\mathcal{P}}}, \mathcal{V}_{\mathcal{O}_{\mathcal{V}}})$ of polynomial time interactive machines $\mathcal{P}_{\mathcal{O}_{\mathcal{P}}}(x, w, r_{\mathcal{P}})$ and $\mathcal{V}_{\mathcal{O}_{\mathcal{V}}}(x, z, r_{\mathcal{V}})$ such that: x is a common input; \mathcal{P} has a secret input w ; \mathcal{V} has an auxiliary input z and produces a binary output (“accept” or “reject”). The system has to fulfil the following properties:

Completeness for any $r_I, r_{\mathcal{P}}, r_{\mathcal{V}}, x, w, z$ such that $R(x; w)$ holds, the outcome of interaction $\mathcal{P}_{\mathcal{O}_{\mathcal{P}}}(x, w; r_{\mathcal{P}}) \leftrightarrow \mathcal{V}_{\mathcal{O}_{\mathcal{V}}}(x, z; r_{\mathcal{V}})$ makes \mathcal{V} accept.

Soundness there exists a polynomial-time algorithm \mathcal{E} (called extractor) which is given full black-box access to the prover such that for any x and z polynomial-time algorithm \mathcal{P}' with access to $\mathcal{O}_{\mathcal{P}}$, if the probability (over all random coins) that $\mathcal{P}'_{\mathcal{O}_{\mathcal{P}}}(x, z; r_{\mathcal{P}}) \leftrightarrow \mathcal{V}_{\mathcal{O}_{\mathcal{V}}}(x, z; r_{\mathcal{V}})$ makes \mathcal{V} accept is non-negligible then $\mathcal{E}^{\mathcal{P}'}(x; r)$ produces w such that $R(x; w)$ with non-negligible probability (over r).

The argument system is called *zero-knowledge (ZK) relative to $\mathcal{O}'_{\mathcal{V}}$* if for any polynomial time algorithm $\mathcal{V}'_{\mathcal{O}'_{\mathcal{V}}}$ there exists a polynomial-time algorithm $\mathcal{S}_{\mathcal{O}'_{\mathcal{V}}}$ (called simulator), which could be run by the verifier, such that for any x, w , and z such that $R(x, w)$, the experiments of either computing $\text{View}_{\mathcal{V}}(\mathcal{P}_{\mathcal{O}_{\mathcal{P}}}(x, w; r_{\mathcal{P}}) \leftrightarrow \mathcal{V}_{\mathcal{O}_{\mathcal{V}}}(x, z; r_{\mathcal{V}})$ or running $\mathcal{S}_{\mathcal{O}'_{\mathcal{V}}}(x, z; r)$ produce two random outputs (over all random coins) with indistinguishable distributions.

5.1 Kolmogorov String Commitment

In order to construct a zero-knowledge protocol from an individual witness hiding protocol, we address the problem of string commitment.

Zero-knowledge protocols for **NP**-complete problems use bit commitment. Unfortunately, Kolmogorov complexity is not suitable to deal with bit commitment. We consider string commitment instead, which (like the notion of bit commitment scheme) is based on the existence of (Kolmogorov) one-way functions.

Protocol 5 (String commitment protocol). Let $f : \Sigma^* \rightarrow \Sigma^*$ be an honest injective one-way function.

Commitment phase: The sender, willing to commit to a string $x \in \Sigma^n$ selects $k \in \{0, 1\}^n$ uniformly at random and sends the pair $(f(k); k \oplus x)$ to the receiver. We call such a k the key for x .

Opening phase: The sender reveals the key k used in the commitment phase. The receiver accepts the value x if $f(k) = \alpha$ and $k \oplus x = \sigma$ where (α, σ) is the receiver’s view of the commitment phase.

Observe that since f is a one-way function, the receiver is not able to compute with non negligible probability (k, x) because it is a pre-image of the string commitment (α, σ) . On the other hand, since the one-way function is injective the sender is not able to change his commitment.

In [1, 3] a Kolmogorov approach to the existence of one-way functions is given. Considering classical one-way functions, the following results were obtained:

1. If E is the expected value of $\mathbf{K}_f^t(x|f(x), n)$ and $E > c$ for some constant c and some fixed polynomial time t , then f is a weak one-way function.
2. If f is a strong one-way function, then $E > c \log n$ for every constant c .

Definition 10 (Kolmogorov one-way function). Let $t(\cdot)$ be some computable time bound, c a constant, $f : \Sigma^n \rightarrow \Sigma^m$ an injective and polynomial time computable function and $\delta(\cdot)$ a positive function. We say that an instance x of length n is (t, δ) -secure relative to a random string $r \in \Sigma^{t(n)}$ and to the function f if $\mathbf{K}_f^t(x|r, n) \geq c \log n$ and $\mathbf{K}_f^t(x|r, n) - \mathbf{K}_f^t(x|f(x), r, n) \leq \delta(n)$. Let $\varepsilon(\cdot)$ be a function. f is a (t, ε, δ) -secure Kolmogorov one-way function if for sufficiently large n , $\Pr_{(x,r) \in \Sigma^n \times \Sigma^{t(n)}} [x \text{ is not } (t, \delta)\text{-secure for } r] \leq \varepsilon(n)$.

Theorem 7 ([1]). *If f is a $(t(n), \varepsilon(n), \delta(n))$ -secure Kolmogorov one-way function, then $E(\mathbf{K}_f^t(x|f(x), r, n)) \geq (1 - \varepsilon(n)) \cdot (n - \log n - \delta(n)) - 2$.*

We can ground the existence of Kolmogorov string commitment schemes on the assumption that there is a Kolmogorov one-way function (KOWF). Given such a KOWF f we note that there are many strings k such that $\mathbf{K}_f^t(k|f(k), r, n) > \mathbf{K}_f^t(k|r, n) - O(\log n)$ that can be used as keys in Protocol 5. Moreover, we can, for example, use Theorem 2 to derive Kolmogorov independent keys.

5.2 A Variant of the IWH Protocol that Is Classical ZK for FHAM

Capitalizing on the knowledge that one-way functions are essential for zero-knowledge of NP complete problems (see [6]), we derive a version of individual witness hiding protocols that is also zero-knowledge in the random oracle model.

Protocol 6 (Random oracle zero-knowledge for FHAM_p)

Input: Graph H with vertices $V = \{1, \dots, n\}$ and with at most $p(n)$ Hamiltonian paths.

Secret of \mathcal{P} : Hamiltonian path c in H . Let $c_1, \dots, c_{p(n)}$ be all possible Hamiltonian paths of H .

Oracle: $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ where the oracle \mathcal{O}_1 when receiving a query either from the prover or the verifier, produces a random permutation π such that:

- (i) For all $i \in \{1, \dots, p(n)\}$, $\mathbf{K}^t(\pi(c_i)|\pi, H) \geq \mathbf{K}^t(c_i) - O(\log n)$;
- (ii) For all $i \in \{1, \dots, p(n)\}$, $\mathbf{K}^t(c_i|\pi(c), H) \geq \mathbf{K}^t(c_i) - O(\log n)$;
- (iii) π is $O(\log n)$ -Kolmogorov independent from all other π 's produced in the previous rounds;

and the oracle \mathcal{O}_2 when queried for key production it generates k_i such that:

- (i) For all $0 \leq j < i$, $\mathbf{K}^t(k_i|k_j) > O(\log n)$;
- (ii) For all $0 \leq j < i$, $\mathbf{K}^t(z_i|k_j) \geq \mathbf{K}^t(z_i) - O(\log n)$ and $\mathbf{K}^t(z_j|k_i) \geq \mathbf{K}^t(z_j) - O(\log n)$;

The following steps are repeated n^2 times.

1. \mathcal{P} queries \mathcal{O}_1 to get a random permutation π and sets $H' = \pi(H)$.
2. \mathcal{P} uses \mathcal{O}_2 to get k_1, \dots, k_{n^2} keys and Protocol 5 to commit to “ $z_i = (u, v)$ is/is not an edge on H' ”, where u and v are any two vertices of H' .
3. \mathcal{V} chooses randomly a bit b . If $b = 0$ he asks \mathcal{P} for the isomorphism between the graph received and H , and to show a Hamiltonian path in H' otherwise.
4. If \mathcal{P} receives 0, he reveals π and all the keys k_1, \dots, k_{n^2} used. Otherwise, he reveals the keys k_1, \dots, k_n forming the path in H' .
5. \mathcal{V} checks whether the messages sent by \mathcal{P} are sound.

Theorem 8. *Protocol 6 is individual witness hiding.*

The proof of this theorem is similar to the proof of Theorem 5. Clearly this new protocol is zero-Knowledge in the Random Oracle model since the usual simulator that queries the Oracles is a valid one. Thus:

Theorem 9. *Protocol 6 is zero-knowledge relative to the given oracle \mathcal{O} .*

Acknowledgements. A very special thank is due to S. Laplante for many discussions. We also would like to thank P. Vitányi, D. Ziao and A. Matos.

6 Appendix

We now provide the proof of Theorem 4.

Proof. The proof of perfect completeness is similar to the proof of the original protocol. Notice that, if \mathcal{P} is able to choose the strings z_j as in the Protocol, (for the existence of such string see below and Theorem 2) then since $(x, n) \in \mathbb{QR}$ and \mathcal{P} knows u such that $u^2 \equiv x \pmod n$, it follows that the Prover is able to fulfill both challenges of \mathcal{V} and hence the probability of \mathcal{P} convincing \mathcal{V} , is 1.

On the other hand, if $(x, n) \notin \mathbb{QR}$ then no matter what \mathcal{P} does, there is no possibility of \mathcal{P} to fulfill both challenges of \mathcal{V} . In particular, the best we can do is to guess which challenge we will be given by \mathcal{V} . The probability of guessing correctly the challenge and prepare the proper commitment for that challenge is $1/2$. Hence, the probability of passing all the m rounds is 2^{-m} . Since m is logarithmic on $|(x, n)|$, then 2^{-m} is a polynomial on $|(x, n)|$.

Now we observe that, at each round there are exponential many strings v_j that can be chosen.

Consider $\ell = \log n$ and observe that $|(x, n)| \leq 2\ell$.

At round j , there were already revealed $j - 1$ strings. Then the tuple $\langle z_1, \dots, z_{j-1} \rangle$ has polynomial size on ℓ and hence by Theorem 2, for any constant d there are, at least, $2^\ell - 2^{\ell-e}$ strings $v \in \Sigma^\ell$ such that $\mathbf{K}^t(u_1, u_2, u_3, u_4 | x, n, z_1, \dots, z_{j-1}, v) \geq \mathbf{K}^t(u_1, u_2, u_3, u_4 | x, n, z_1, \dots, z_{j-1}) - d$ for some constant e sufficiently large. Let A be the set of such strings. Since $\gcd(x, n) = 1$ (otherwise the instance of \mathbb{QR} would be easy to solve), the function $f_u(v) = uv$ is injective. Hence $f_u(A)$ also has, at least, $2^\ell - 2^{\ell-e}$ elements. Therefore $A \cap f_u(A)$ has, at least, $2(2^\ell - 2^{\ell-e}) - 2^m = 2^m - 2^{m-\ell-1}$ elements.

Notice that every v in $A \cap f_u(A)$ satisfies the requirements for round j since, in particular, we have $\mathbf{K}^t(u_i | x, n, z_1, \dots, z_{j-1}, v) \geq \mathbf{K}^t(u_i | x, n, z_1, \dots, z_{j-1}) - d'$ and $\mathbf{K}^t(u_i | x, n, z_1, \dots, z_{j-1}, uv) \geq \mathbf{K}^t(u_i | x, n, z_1, \dots, z_{j-1}) - d'$ for $i = 1..4$ and for some constant d' depending on d .

To complete the proof we only need to show that, at the end of the protocol, the amount of information that is leaked about the witnesses is logarithmic. Notice that, by the choices of v 's, in between rounds, there is only a constant number of bits of information that are leaked.

So, for all $u_i = u_1, \dots, u_4$ we have

$$\begin{aligned} \mathbf{KC}_{\mathcal{P}, \mathcal{V}}^t(u_i; (x, n)) &= \max_{c \in \mathcal{C}} (\mathbf{K}^t(u_i | x, n) - \mathbf{K}^t(u | x, n, c_{\mathcal{P}\mathcal{V}})) \\ &= \mathbf{K}^t(u_i | x, n) - \mathbf{K}^t(u_i | x, n, z_1, \dots, z_m) \\ &\leq \mathbf{K}^t(u_i | x, n) - \mathbf{K}^t(u_i | x, n, z_1, \dots, z_{m-2}) + d_m + d_{m-1} \\ &\leq \mathbf{K}^t(u_i | x, n) - \mathbf{K}^t(u_i | x, n) + d_m + d_{m-1} + \dots + d_1 \\ &\leq d' \times m = d' \times c \log |(x, n)| \leq O(\log n) \end{aligned}$$

Next we provide the proof of Theorem 5.

Proof. Similarly to the proof presented for Protocol 2, the proof of perfect completeness follows from the fact that if \mathcal{P} is able to choose the permutations π

as in the Protocol, (for the existence of such permutations see below and Theorem 2) then since $H \in \mathbb{FHAM}$ and \mathcal{P} knows a Hamiltonian path, it follows that the Prover is able to fulfill both challenges of \mathcal{V} and hence the probability of \mathcal{P} convincing \mathcal{V} , is 1.

On the other hand, if H is not Hamiltonian then no matter what \mathcal{P} does, there is no possibility of \mathcal{P} to fulfill both challenges of \mathcal{V} . In particular, the best we can do is to guess which challenge we will be given by \mathcal{V} . The probability of guessing correctly the challenge and prepare the proper commitment for that challenge is $1/2$. Hence, the probability of passing all the m rounds is 2^{-m} . Since m is logarithmic on $|H|$, then 2^{-m} is a polynomial on $|H|$.

Now we observe that, at each round there are exponentially many strings π that can be chosen. Notice that any permutation π can be described with $n \log n$ bits where n is the number of vertices in H . Let $\ell = n \log n$.

At round j , there were already revealed $j - 1$ strings, that either a cycle of a permutation. Then the tuple $\langle z_1, \dots, z_{j-1} \rangle$ has polynomial size on ℓ and hence by Theorem 2, for any constant d there are, at least, $2^\ell - 2^{\ell-e}$ permutations $v \in \Sigma^\ell$ such that $\mathbf{K}^t(\langle c_1, \dots, c_j \rangle | H, z_1, \dots, z_{j-1}, \pi) \geq \mathbf{K}^t(\langle c_1, \dots, c_j \rangle | H, z_1, \dots, z_{j-1}) - d$ for some constant e sufficiently large. On the other hand, since $|\pi(c)| = \ell$, then there are also $2^\ell - 2^{\ell-e}$ permutations $v \in \Sigma^\ell$ such that $\mathbf{K}^t(\langle c_1, \dots, c_j \rangle | H, z_1, \dots, z_{j-1}, \pi(c)) \geq \mathbf{K}^t(\langle c_1, \dots, c_j \rangle | H, z_1, \dots, z_{j-1}) - d$ for some constant e sufficiently large. Then, again there are $2^\ell - 2^{\ell-e-1}$ possible permutations satisfying the conditions required in the protocol.

To complete the proof we have to show that the amount of information that leaked about the paths is only logarithmic. So, for every path c_i that is a cycle in G we have

$$\begin{aligned} \mathbf{KC}_{\mathcal{P}, \mathcal{V}}^t(H, c_i) &= \max_{c \in C} (\mathbf{K}^t(c_i | H) - \mathbf{K}^t(c_i | H, c_{\mathcal{P}\mathcal{V}})) \\ &\leq \mathbf{K}^t(c_i | H) - \mathbf{K}^t(c_i | H, z_1, \dots, z_{m-1}) + d_m \\ &\leq \mathbf{K}^t(c_i | H) - \mathbf{K}^t(c_i | H) + d_m + d_{m-1} + \dots + d_1 \\ &\leq d' \times m = d' \times c \log |H| \leq O(\log n) \end{aligned}$$

References

1. Antunes, L., Matos, A., Pinto, A., Souto, A., Teixeira, A.: One-way function using algorithmic and classical information theories. ToCS **52**, 162 (2013)
2. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC, pp. 235–244 (2000)
3. Casal, F., Rasga, J., Souto, A.: Kolmogorov one-way functions revisited. Cryptogr. - MDPI **2**, 9 (2018)
4. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proceedings of STOC. ACM (1990)
5. Goldreich, O.: Foundations of Cryptography. Cambridge University Press, Cambridge (2001)
6. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM **38**, 690–728 (1991)

7. Goldreich, O., Petrank, E.: Quantifying knowledge complexity. *Comput. Complex.* **8**(1), 50–98 (1999)
8. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: *Proceedings of STOC*. ACM (1985)
9. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **18**(1), 186–208 (1989)
10. Kaplan, M., Laplante, S.: Kolmogorov complexity and combinatorial methods in communication complexity. In: Chen, J., Cooper, S.B. (eds.) *TAMC 2009*. LNCS, vol. 5532, pp. 261–270. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02017-9_29
11. Li, M., Vitányi, P.: *An Introduction to Kolmogorov Complexity and Its Applications*. TCS. Springer, New York (2008). <https://doi.org/10.1007/978-0-387-49820-1>
12. Mateus, P., Vaudenay, S.: On tamper-resistance from a theoretical viewpoint. In: Clavier, C., Gaj, K. (eds.) *CHES 2009*. LNCS, vol. 5747, pp. 411–428. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04138-9_29
13. Yehuda, R., Chor, B., Kushilevitz, E., Orlitsky, A.: Privacy, additional information and communication. *IEEE Tran. Inf. Theo.* **39**(6), 1930–1943 (1993)
14. Stinson, D.: *Cryptography: Theory and Practice*. CRC Press, Boca Raton (1995)



Algorithms and Geometric Constructions

Vladimir Uspenskiy¹ and Alexander Shen²(✉)

¹ Moscow State Lomonosov University, Moscow, Russia

² LIRMM CNRS and University of Montpellier, on leave from IITP RAS,
Montpellier, France

`alexander.shen@lirmm.fr`

Abstract. It is well known that several classical geometry problems (e.g., angle trisection) are unsolvable by compass and straightedge constructions. But what kind of object is proven to be non-existing by usual arguments? These arguments refer to an intuitive idea of a geometric construction as a special kind of an “algorithm” using restricted means (straightedge and/or compass). However, the formalization is not obvious, and different descriptions existing in the literature are far from being complete and clear. We discuss the history of this notion and a possible definition in terms of a simple game.

1 Introduction

The notion of an algorithm as an intuitively clear notion that precedes any formalization, has a rather short history. The first examples of what we now call algorithms were given already by Euclid and al-Khwârizmî. But the general idea of an algorithm seems to appear only in 1912 when Borel considered “les calculs qui peuvent être réellement effectués”¹ and emphasized: “Je laisse intentionnellement de côté le plus ou moins grande longueur pratique des opérations; l’essentiel est que chacune de ces opérations soit exécutable en un temps fini, par une méthode sûre et sans ambiguïté”² [4, p. 162]. The formal definition of a representative class of algorithms was given in 1930s (in the classical works of Gödel, Church, Kleene, Turing, Post and others); the famous Church–Turing thesis claims that the class of algorithms provided by these formal definitions is representative.

In this paper we look at the history of another related notion: the notion of a *geometric construction*. One may consider geometric constructions as a special type of algorithms that deal with geometric objects. Euclid provided many examples of geometric constructions by compass and straightedge (ruler); later these constructions became a standard topic for high school geometry exercises. Several classical problems (angle trisection, doubling the square, squaring the

A. Shen—Supported by ANR-15-CE40-0016-01 RaCAF grant.

¹ The computations that can be really performed.

² I intentionally put aside the question of bigger or smaller practical length of the operation; it is important only that each of the operations can be performed in a finite time by a clear and unambiguous method”.

circle) were posed and remained unsolved since ancient times (though solutions that involve more advanced instruments than compass and straightedge were suggested). These problems were proved to be unsolvable in 19th century. One would expect that the proof of unsolvability assumes as a prerequisite a rigorously defined notion of a “solution” that does not exist. Recall that the first undecidability proofs could appear only after an exact definition of an algorithm was given.

However, historically this was not the case and the impossibility proofs appeared without an exact definition of a “geometric construction”. These proofs used the algebraic approach: For example, to show that the cube cannot be doubled, one proves that $\sqrt[3]{2}$ cannot be obtained from rationals by arithmetic operations and square roots. The reduction from a geometric question to an algebraic one looks quite obvious and was omitted by Wantzel who first proved the impossibility of angle trisection and cube doubling. As he wrote in [22], “pour reconnaître si la construction d’un problème de Géométrie peut s’effectuer avec la règle et le compas, il faut chercher s’il est possible de faire dépendre les racines de l’équation à laquelle il conduit de celles d’un système d’équations du second degré”.³ This is said in the first paragraph of the paper and then he considers only the algebraic question.

Several other interesting results were obtained in 19th century. It was shown that all constructions by compass and straightedge can be performed using the compass only (the *Mohr–Mascheroni theorem*) if we agree that a line is represented by a pair of points on this line. Another famous result from 19th century, the *Poncelet–Steiner theorem*, says that if a circle with its center is given, then the use of compass can be avoided, straightedge is enough. Other sets of tools were also considered, see, e.g., [3, 9, 14].

Later geometric construction became a popular topic of recreational mathematics (see, e.g., [6, 10, 13, 15]). In most of the expositions the general notion of a geometric construction is still taken as granted, without a formal definition, even in the nonexistence proofs (e.g., when explaining Hilbert’s proof that the center of a circle cannot be found using only a straightedge [6, 10, 15]; see below Sect. 6 about problems with this argument). Sometimes a definition for some restricted class of geometric construction is given (see, e.g., [18]). In [13] an attempt to provide a formal definition is made, still it remains ambiguous with respect to the use of “arbitrary points” (see Sect. 4). Baston and Bostock [2] observe that the intuitive idea of a “geometric construction” has no adequate formal definition and discuss several examples but do not attempt to give a formal definition that is close to the intuitive notion. It seems that even today people still consider the intuitive notion of a “geometric construction algorithm” as clear enough to be used without a formal definition (cf. [1], especially the first arXiv version).

In Sect. 2 we consider a naïve approach that identifies constructible points with the so-called “derivable” points. Then in Sects. 3 and 4 we explain why

³ To find out whether a geometric problem can be solved by straightedge and compass construction, one should find whether it is possible to reduce the task of finding the roots of the corresponding equation to a system of equations of second degree.

this approach contradicts our intuition. In Sect. 5 we suggest a more suitable definition, and finally in Sect. 6 we note that the absence of formal definitions has led to incorrect proofs.

2 Derivable Points and Straight-Line Programs

At first it seems that the definition of a geometric construction is straightforward. We have three classes of geometric objects: points, lines and circles. Then we consider some operations that can be performed on these objects. We need to obtain some object (the goal of our construction) applying the allowed operations to given objects. As Tao [18] puts it,

Formally, one can set up the problem as follows. Define a configuration to be a finite collection \mathcal{C} of points, lines, and circles in the Euclidean plane. Define a construction step to be one of the following operations to enlarge the collection \mathcal{C} :

- (Straightedge) Given two distinct points A, B in \mathcal{C} , form the line \overline{AB} that connects A and B , and add it to \mathcal{C} .
- (Compass) Given two distinct points A, B in \mathcal{C} , and given a third point O in \mathcal{C} (which may or may not equal A or B), form the circle with centre O and radius equal to the length $|AB|$ of the line segment joining A and B , and add it to \mathcal{C} .
- (Intersection) Given two distinct curves γ, γ' in \mathcal{C} (thus γ is either a line or a circle in \mathcal{C} , and similarly for γ'), select a point P that is common to both γ and γ' (there are at most two such points), and add it to \mathcal{C} .

We say that a point, line, or circle is constructible by straightedge and compass from a configuration \mathcal{C} if it can be obtained from \mathcal{C} after applying a finite number of construction steps.

We can even try to define the geometric construction algorithm as a straight-line program, a sequence of assignments whose left-hand side is a fresh variable and the right-hand side contains the name of the allowed operation and the names of objects to which this operation is applied.

Baston and Bostock [2] use the name “derivable” for objects that can be obtained in this way starting from given objects. In other words, starting with some set of given objects, they consider its closure, i.e., the minimal set of objects that contains the given ones and is closed under allowed operations. The objects that belong to this closure are called *derivable* from the given ones. In these terms, the impossibility of trisecting the angle with the compass and the straightedge can be stated as follows: *for some points A, B, C the trisectors of the angle BAC are not derivable from $\{A, B, C\}$.*

Baston and Bostock note that the intuitive notion of a “constructible” point (that they intentionally leave without any definition) may differ from the formal notion of a derivable point in both directions. We discuss the differences in the following sections.

3 Uniformity and Tests

There are some problems with this approach. First of all, this approach is “non-uniform”. Asking a high school student to construct, say, a center of an inscribed circle of a triangle ABC , we expect the solution to be some specific construction that works *for all triangles*, not just the proof that this center is always derivable from A , B , and C . The naïve approach would be to ask for a straight-line program that computes this center starting from A , B , and C . However, an obvious problem arises: the operation of choosing an intersection point of two curves is non-deterministic (we need to choose one of two intersection points). We may guarantee only that *some* run of the program produces the required object, or guarantee that the required object is among the objects computed by this program. This is a common situation for classical constructions. For example, the standard construction of the centre of the incircle of a triangle can also produce centres of excircles (the circles outside the triangle that touch one of its sides and the extensions of two other sides).

The non-deterministic nature of the operations was mentioned by different authors. Bieberbach [3] says that the constructions should be performed in the “oriented plane” (not giving any definitions). Tietze [19–21] notes that some objects can be constructed but only in a non-deterministic way, again without giving definition of these notions.

One could give up and consider the non-uniform setting only. As Manin [13, p. 209] puts it, “we ignore how to choose the required point from the set of points obtained by the construction”. Another approach is to replace straight-line programs by decision trees where tests appear as internal nodes. Still none of these two approaches (decision trees or non-deterministic choice) is enough to save some classical constructions in a uniform setting as observed by Baston and Bostock [2, p. 1020]. They noted that the construction from Mohr–Mascheroni theorem allows us to construct the intersection point of two intersecting lines AB and CD (given A, B, C, D) using only a compass. Each use of the compass increases the diameter of the current configuration at most by an $O(1)$ -factor, and the intersection point can be arbitrarily far even if A, B, C, D are close to each other, so there could be no *a priori* bound on the number of steps. The necessity of an iterative process in the Mohr–Mascheroni theorem was earlier mentioned in another form by Dono Kijne [11, ch. VIII, p. 99]; he noted that this result depends on Archimedes’ axiom.

To save the Mohr–Mascheroni construction, one may consider programs that allow loops. This was suggested, e.g., by Engeler [7]. Here we should specify what kind of data structures are allowed (e.g., whether we allow dynamic arrays of geometric objects or not). In this way we encounter another problem, at least if we consider straightedge-only constructions on the *rational* plane \mathbb{Q}^2 and allow using tests and do not bound the number of steps/objects. Baston and Bostock [2] observed that having four different points $A, B, C, D \in \mathbb{Q}^2$ in a general position (no three points lie on a line, no two connecting lines are parallel), we can enumerate all (rational) points and therefore all rational lines. Then we can wait until a line parallel to AB appears (we assume that we may

test whether two given lines intersect or are parallel) and then use this parallel line to find the midpoint of AB . This construction does not look like a intuitively valid geometric construction and contradicts the belief that one cannot construct the midpoint using only a straightedge, see [2] for details.

4 Arbitrary Points

Let us now consider the other (and probably more serious) reason why the notion of a derivable object differs from the intuitive notion of a constructible object. Recall the statement about angle trisection as stated by Tao [18]: for some triangle ABC the trisectors of angle BAC are not derivable from $\{A, B, C\}$. (Tao uses the word “constructible”, but we keep this name for the intuitive notion, following [2].) Tao interprets this statement as the impossibility of angle trisection with a compass and straightedge, and for a good reason.

On the other hand, the center of a circle is not derivable from the circle itself, for the obvious reason that no operation can be applied to enlarge the collection that consists only of the circle. Should we then say that the center of a given circle cannot be constructed by straightedge and compass? Probably not, since such a construction is well known from the high school. A similar situation happens with the construction of a bisector of a given angle (a configuration consisting of two lines and their intersection point).

Looking at the corresponding standard constructions, we notice that they involve another type of steps, “choosing an arbitrary point” (on the circle or elsewhere). But we cannot just add the operation “add an arbitrary point” to the list of allowed operations, since all points would become derivable. So what are the “arbitrary points” that we are allowed to add? Bieberbach [3, p. 21] speaks about “Punkte, über die keine Angaben affiner oder metrischer Art gemacht sind”⁴ and calls them “willkürliche Punkte”—but this hardly can be considered as a formal definition.

Tietze [21] notes only that “the role of arbitrary elements is not so simple as it is sometimes thought”. Baston and Bostock [2] explain the role of arbitrary elements, but say only that “the distinction between constructibility and derivability arising from the use of arbitrary points is not very complex” and “we will not pursue a more detailed analysis in this direction”; they refer to [12] for an “elementary approach”, but this book also does not give any clear definition. Probably the most detailed explanation of the role of arbitrary points is provided by Manin [13], but he still defines the construction as a “finite sequence of steps” (including the “arbitrary choices”) and says that a point is constructible if there exists a construction that includes this point “for all possible intermediate arbitrary choices”; this definition, if understood literally, makes no sense since different choices leads to different constructions. Schreiber [16] tries to define the use of arbitrary points in a logical framework, but his exposition is also far from being clear.

⁴ Points for which we do not have affine or metric information.

How can we modify the definitions to make them rigorous? One of the possibilities is to consider the construction as a strategy in some game with explicitly defined rules. We discuss this approach in the next section.

5 Game Definition

The natural interpretation of the “arbitrary choice” is that the choice is made by an adversary. In other words, we consider a game with two players, Alice and Bob. We start with the non-uniform version of this game.

Let E be some finite set of geometric objects (points, lines, and circles). To define which objects x are constructible starting from E , consider the following full information game. The *position* of the game is a finite set of geometric objects. The initial position is E . During the game, Alice and Bob alternate. Alice makes some requests, and Bob fulfills these requests by adding some elements to the current position. Alice wins the game when x becomes an element of the current position. The number of moves is unbounded, so it is possible that the game is infinite (if x never appears in the current position); in this case Alice does not win the game.

Here are possible request types.

- Alice may ask Bob to add to the current position some straight line that goes through two different points from the current position.
- Alice may ask Bob to add to the current position a circle with center A and radius BC , if A, B, C are points from the current position.
- Alice may ask Bob to add to the current position one or two points that form the intersection of two different objects (lines or circles) that already belong to the current position.

If we stop here, we get exactly the notion of derivable points, though in a strange form of a “game” where Bob has no choice. To take the “arbitrary” points into account, we add one more operation:

- Alice specifies an open subset of the plane (say, an open circle), and Bob adds some point of this subset to the current position.

The point x is *constructible* from E if Alice has a winning strategy in this game.

Let us comment on the last operation.

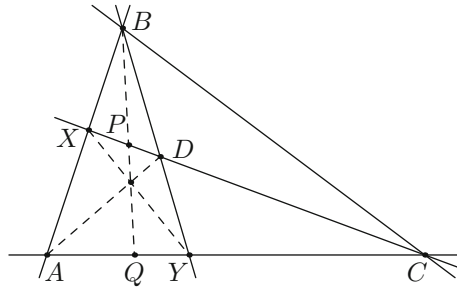
- (1) Note that Alice cannot (directly) force Bob to choose some point on a line or on a circle, and this is often needed in the standard geometric constructions. But this is inessential since Alice can achieve this goal in several steps. First she asks to add points on both sides of the line or circle (selecting two small open sets on both sides in such a way that every interval with endpoints in these open sets intersects the line or circle), then asks to connect these points by a line, and then asks to add the intersection point of this new line and the original one.

- (2) On the other hand, according to our rules, Alice can specify with arbitrarily high precision where the new point should be (by choosing a small open set). A weaker (for Alice) option would be to allow her to choose a connected component of the complement of the union of all objects in the current position. Then Bob should add some point of this component to the current position.

Proposition 1. *This restriction does not change the notion of a constructible point.*

Proof. Idea: Using the weaker option, Alice may force Bob to put enough points to make the set of derivable points dense, and then use the first three options to get a point in an arbitrary open set.

Let us explain the details. First, she asks for an arbitrary point A , then for a point B that differs from A , then for line AB , then for a point C outside line AB (thus having the triangle ABC), then for the sides of this triangle, and then for a point D inside the triangle. (All this is allowed in the restricted version.)



Now the points P and Q obtained as shown are derivable (after the projective transformation that moves B and C to infinity, the points P and Q become the midpoints of XD and AY). Repeating this construction, we get a dense set of derivable points on intervals XD and AY , then the dense set of derivable points in the quadrangle $AXDY$ and then in the entire plane.

Now, instead of asking Bob for a point in some open set U , Alice may force him to include one of the derivable points (from the dense set discussed above) that is in U .

This definition of constructibility turns out to be equivalent to the negative definition suggested by Akopyan and Fedorov [1]. They define non-constructibility as follows: an object x is *non-constructible* from a finite set E of objects if there exists a set $E' \supset E$ that is closed under the operations of adding points, lines, and circles (contains all objects derivable from E'), contains an everywhere dense set of points, but does not contain x .

Proposition 2 (Akopyan–Fedorov). *This negative definition is equivalent to the game-theoretic definition given above.*

Proof. The equivalence is essentially proven as [1, Proposition 15, p. 9], but Akopyan and Fedorov avoided stating explicitly the game-theoretic definition and spoke about “algorithms” instead (without an exact definition).

Assume that x is non-constructible from E according to the negative definition. Then Bob can prevent Alice from winning by always choosing points from E' when Alice asks for a point in an open set. Since E' is dense, these points are enough. If Bob follows this strategy, then the current position will always be a subset of E' and therefore will never contain x .

On the other hand, assume that x is not constructible from E in the sense of the positive definition. Consider the following strategy for Alice. She takes some triangle abc and point d inside it and ask Bob to add points A, B, C, D that belong to some small neighborhoods of a, b, c, d respectively. The size of these neighborhoods guarantees that ABC is a triangle and D is a point inside ABC . There are two cases:

- for every choice of Bob Alice has a winning strategy in the remaining game;
- there are some points A, B, C, D such that Alice does not have a winning strategy in the remaining game.

In the first case Alice has a winning strategy in the entire game and x is constructible. In the second case we consider the set E' of all objects derivable from $E \cup \{A, B, C, D\}$. As we have seen in the proof of the previous proposition, this set is dense. Therefore, x is non-constructible in the sense of the negative definition.

The advantage of the game definition is that it can be reasonably extended to the uniform case. For the uniform case the game is no more a full-information game. Alice sees only the names (and types) of geometric objects in E , and assigns names to new objects produced by Bob. One should agree also how Alice can get information about the configuration and how she can specify the connected component when asking Bob for a point in this component. For example, we may assume that Alice has access to the list of all connected components and the full topological information about the structure they form, as well as the places of objects from E in this structure. Then Alice may choose some component and request a point from it. To win, Alice needs to specify the name of the required object x . After we agree on the details of the game, we may define construction algorithms as computable strategies for such a game. (Note that in this version Alice deals only with finite objects).

6 Formal Definitions Are Important

In fact, the absence of formal definitions and exact statements is more dangerous than one could think. It turned out that some classical and well known arguments contain a serious gap that cannot be filled without changing the argument. This happened with a proof (attributed to Hilbert in [5]) that one cannot find the center of a given circle using only a straightedge. It is reproduced in many

popular books (see, e.g., [6, 10, 15]) and all the arguments (at least in the four sources mentioned above) have the same gap. They all go as follows [10, p. 18]:

Let the construction be performed in a plane P_1 and imagining a transformation or mapping T of the plane P_1 into another plane P_2 such that:

- (a) straight lines in P_1 transform into straight lines in P_2 $\langle \dots \rangle$
- (b) The circumference C of our circle is transformed into a circumference $T(C)$ for some circle in P_2 .

As the steps called for in the construction are being performed in P_1 , they are being faithfully copied in P_2 . Thus when the construction in P_1 terminates in the centre O of C , the “image” construction *must* terminate in the centre $T(O)$ of the circle $T(C)$.

Therefore if one can exhibit a transformation T satisfying (a) and (b), but such that $T(O)$ is *not* the centre of $T(C)$, then the impossibility of constructing the centre of a circle by ruler alone will be demonstrated.

Such a transformation indeed exists, but the argument in the last paragraph has a gap. If we understand the notion of construction in a non-uniform way and require that the point was among the points constructed, the argument does not work since the center of $T(C)$ could be the image of some other constructed point. If we use some kind of the uniform definition and allow tests, then these tests can give different results in P_1 and P_2 (the projective transformation used to map P_1 into P_2 does not preserve the ordering), so there is no reason to expect that the construction is “faithfully copied”. And a uniform definition that does not allow tests and still is reasonable, is hard to imagine (and not given in the book). Note also that some lines that intersect in P_1 , can become parallel in P_2 .

It is easy to correct the argument and make it work for the definition of constructibility given above (using the fact that there are many projective mappings that preserve the circle), but still one can say without much exaggeration that the first correct proof of this impossibility result appeared only in [1]. One can add also that the stronger result about two circles that was claimed by Cauer [5] and reproduced with a similar proof in [15], turned out to be plainly false as shown in [1], and the problems in the proof were noted already by Gram [8]. It is not clear why Gram did not question the validity of the classical proof for one circle, since the argument is the same. Gram did not try to give a rigorous definition of the notion of a geometric construction, speaking instead about constructions in the “ordered plane” and referring to Bieberbach’s book [3] that also has no formal definitions.

The weak version of Cauer’s result saying that for some pairs of circles one cannot construct their centers, can be saved and proven for the definition of constructibility discussed above (see [1] and the popular exposition in [17]).

It would be interesting to reconsider the other results claimed about geometric constructions (for example, in [9, 19–21]) to see whether the proofs work for some clearly defined notion of a geometric construction. Note that in some cases (e.g., for Tietze’s results) some definition of the geometric construction for the uniform case is needed (and the negative definition is not enough).

Acknowledgements. The authors thank Sergey Markelov, Arseny Akopyan, Roman Fedorov and their colleagues at Moscow State University and LIRMM (Montpellier) for interesting discussions.

References

1. Akopyan, A., Fedorov, R.: Two circles and only a straightedge (2017). <https://arxiv.org/abs/1709.02562>
2. Baston, V.J., Bostock, F.A.: On the impossibility of ruler-only constructions. *Proc. Am. Math. Soc.* **110**(4), 1017–1025 (1990)
3. Bieberbach, L.: *Theorie der Geometrischen Konstruktionen*. Springer, Basel (1952). <https://doi.org/10.1007/978-3-0348-6910-2>
4. Borel, E.: Le calcul des intégrales définies. *J. Math. pures appl. ser. 6* **8**(2), 159–210 (1912)
5. Cauer, D.: Über die Konstruktion des Mittelpunktes eines Kreises mit dem Lineal allein. *Math. Annalen* **73**, 90–94 (1913). A correction: **74**, 462–464
6. Courant, R., Robbins, H., revised by Stewart, I.: *What is Mathematics? An Elementary Approach to Ideas and Methods*. Oxford University Press, Oxford (1996)
7. Engeler, E.: Remarks on the theory of geometrical constructions. In: Barwise, J. (ed.) *The Syntax and Semantics of Infinitary Languages*. LNM, vol. 72, pp. 64–76. Springer, Heidelberg (1968). <https://doi.org/10.1007/BFb0079682>
8. Gram, C.: A remark on the construction of the centre of a circle by means of the ruler. *Math. Scand.* **4**, 157–160 (1956)
9. Hilbert, D.: *The foundations of geometry*, authorized translation by E.J. Townsend, Ph.D., University of Illinois (1902)
10. Kac, M., Ulam, S.M.: *Mathematic and Logic*. Dover publications, New York (1992)
11. Kijne, D.: *Plane construction field theory*. Ph.D. thesis, promotor H. Freudenthal, van Gorcum & Co., N.V., G.A. Hak, H.J. Prakke, 28 May 1956
12. Kutuzov, B.V.: *Studies in mathematics, vol. IV, Geometry* (trans. by L.I. Gordon, E.S. Shater) School Mathematics Study Group, Chicago (1960)
13. Manin, Y.: On the decidability of geometric construction problems using compass and straightedge [Russian]. *Encyclopedia of Elementary Mathematics, Geometry*, Moscow, vol. IV, pp. 205–227 (1963)
14. Martin, G.E.: *Geometric Constructions*. Springer, New York (1998). <https://doi.org/10.1007/978-1-4612-0629-3>
15. Rademacher, H., Toeplitz, O.: *Von Zahlen und Figuren*, 2nd edn. Springer, Heidelberg (1933). <https://doi.org/10.1007/978-3-662-36239-6>
16. Schreiber, P.: *Theorie der Geometrischen Konstruktionen*. VEB Deutscher Verlag der Wissenschaften, Berlin (1975)
17. Shen, A.: Hilbert’s Error? (2018). <https://arxiv.org/abs/1801.04742>
18. Tao, T.: A geometric proof of the impossibility of angle trisection. <https://terrytao.wordpress.com/2011/08/10/a-geometric-proof-of-the-impossibility-of-angle-trisection-by-straightedge-and-compass/>
19. Tietze, H.: Über die Konstruierbarkeit mit Lineal und Zirkel, *Sitzungsberichte der Kaiserlichen Akademie der Wissenschaften, Abt. Ila*, 735–757 (1909). <https://www.biodiversitylibrary.org/item/93371>
20. Tietze, H.: Über die mit Lineal und Zirkel und die mit dem rechten Zeichenwinkel lösbaren Konstruktionsaufgaben I. *Math. Zeitschrift* **46**, 190–203 (1940). <http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002379074>

21. Tietze, H.: Zur Analyse der Lineal- und Zirkelkonstruktionen. I. Sitzungsberichte der mathematisch-naturwissenschaftlichen Abteilung der Bayrischen Akademie der Wissenschaften zu München, 1944, Heft III, Sitzungen Oktober–Dezember, pp. 209–231, München (1947). <http://publikationen.badw.de/003900992.pdf>
22. Wantzel, M.L.: Recherches sur les moyens de reconnaître si un problème de Géométrie peut se résoudre avec la règle et le compas. J. Math. pures Appl. 1re série **2**, 366–372 (1837)



Computing with Multisets: A Survey on Reaction Automata Theory

Takashi Yokomori¹(✉) and Fumiya Okubo²

¹ Department of Mathematics, Faculty of Education, Integrated Arts and Sciences,
Waseda University, 1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8050, Japan
yokomori@waseda.jp

² Faculty of Business Administration, Takachiho University, 2-19-1 Ohmiya,
Suginami-ku, Tokyo 168-8508, Japan
fokubo@takachiho.ac.jp

1 Introduction

In Natural Computing research [18], as mathematical modeling tools of biochemical reactions, Ehrenfeucht and Rozenberg introduced a formal model called reaction systems [6] for investigating the functioning of the living cell, where two basic components (reactants and inhibitors) play a key role as a regulation mechanism in controlling interactions.

Inspired by the notion of reaction systems, *reaction automata* (RAs) have been first introduced in [10] as computing devices for accepting string languages. The notion of RAs is an extension of reaction systems in that RAs employ reactions which are defined by triples (consisting of reactants, inhibitors and products), however they entail dealing with multisets for reactants and products (rather than usual sets as reaction systems do). Thus, RAs are computing models based on multiset rewriting that accept string languages. Another feature that distinguishes RAs from reaction systems is that a reaction automaton receives its input by feeding one symbol of an input string at each step of the computation. In this respect, RAs have the taste similar to P automata, P systems with accepting configurations, in which the idea of taking input sequences of multisets into the systems from the environment was introduced (e.g., [5]).

This survey paper is primarily based on the research works on RAs achieved in [9–12, 15] in which various classes of RAs are considered in four types of computation process: with/without λ -input modes in rule application of the maximally parallel manner and the sequential manner.

In what follows, we make a quick survey of the results presented in this paper. The first result in the series of papers was that RAs are computationally Turing universal [10]. In the paper, space-bounded complexity classes of RAs was also introduced, and in the follow-up paper [11], it eventually turned out that

F. Okubo—The work of T. Yokomori was in part supported by JSPS KAKENHI, Grant-in-Aid for Scientific Research (C) JP17K00021, and by Waseda University grant for Special Research Project: 2017K-121. The work of F. Okubo was in part supported by JSPS KAKENHI Grant Number JP16K16008.

exponential-bounded RAs can exactly characterize the class of context-sensitive languages. Further characterization result of the class of recursively enumerable languages was developed by using the class of linear-bounded RAs together with homomorphisms. The paper [11] also considered some of the closure properties of language classes defined by linear-bounded RAs, showing that the class forms an AFL, while the issue of the computing powers of RAs with λ -input mode in sequential manner was taken up in [9] and investigated to prove that they have again the Turing universal computability. Further investigations was made to a simpler variant of RAs called Chemical reaction automata (CRAs), and it was proved that CRAs with λ -input mode in parallel manner have the Turing universal computability, while their computing powers coincide with the class of Petri net languages when working in sequential manner. In [15], several subclasses of CRAs was introduced and their computing powers was studied. We leave more details of the results to later sections.

The rest of this section is devoted to a brief preliminary. We assume that the reader is familiar with the basic notions of reaction systems as well as of formal language theory. For unexplained details in the theory of reaction systems and in formal language theory, refer to, e.g., [6,7], respectively.

We use the basic notations and definitions concerning multisets that follow [4]. A *multiset* over an alphabet V is a mapping $\mu : V \rightarrow \mathbf{N}$, where \mathbf{N} is the set of non-negative integers, and for each $a \in V$, $\mu(a)$ represents the number of occurrences of a in the multiset μ . The set of all multisets over V is denoted by $V^\#$, including the empty multiset denoted by μ_λ , where $\mu_\lambda(a) = 0$ for all $a \in V$. A multiset μ over V can be represented by any permutation of the string $x = a_1^{\mu(a_1)} \dots a_n^{\mu(a_n)}$, where $V = \{a_1, a_2, \dots, a_n\}$. (Note that for each $a \in V$, a^0 is an empty string λ .) In this sense, a multiset μ is often identified with its string representation x_μ or any permutation of x_μ . A usual set $U (\subseteq V)$ is regarded as a multiset μ_U such that $\mu_U(a) = 1$ if a is in U and $\mu_U(a) = 0$ otherwise. In particular, for each symbol $a \in V$, a multiset $\mu_{\{a\}}$ is often denoted by a itself.

2 Reaction Automata

Inspired by the works of reaction systems (initiated by [6]), the notion of reaction automata has been introduced in [10] by extending sets in each reaction (of a reaction system) to multisets. Here, we start by recalling basic notions concerning reaction automata.

Definition 1. For a set S , a *reaction* in S is a 3-tuple $\mathbf{a} = (R_{\mathbf{a}}, I_{\mathbf{a}}, P_{\mathbf{a}})$ of finite multisets such that $R_{\mathbf{a}}, P_{\mathbf{a}} \in S^\#$, $I_{\mathbf{a}} \subseteq S$ and $R_{\mathbf{a}} \cap I_{\mathbf{a}} = \emptyset$. The multisets $R_{\mathbf{a}}$ and $P_{\mathbf{a}}$ are called the *reactant* of \mathbf{a} and the *product* of \mathbf{a} , respectively, while the set $I_{\mathbf{a}}$ is called the *inhibitor* of \mathbf{a} .

These notations are extended to a multiset of *reactions* as follows: For a set of reactions A and a multiset α over A (i.e., $\alpha \in A^\#$),

$$R_\alpha = \sum_{\mathbf{a} \in A} R_{\mathbf{a}}^{\alpha(\mathbf{a})}, \quad I_\alpha = \bigcup_{\mathbf{a} \subseteq \alpha} I_{\mathbf{a}}, \quad P_\alpha = \sum_{\mathbf{a} \in A} P_{\mathbf{a}}^{\alpha(\mathbf{a})},$$

Notes. (1). A is often identified with its label set and is used as an alphabet.
 (2). The symbol $\sum_{\mathbf{a} \in A}$ denotes the sum of multisets.

In the existing works on reaction automata, two ways of applying reactions have been considered: the sequential manner and the maximally parallel manner. The former manner was adopted in [10, 11], while the literatures [9, 12, 15] studied the latter as well.

Definition 2. Let A be a set of reactions in S and $\alpha \in A^\#$ be a multiset of reactions over A . Then, for a finite multiset $T \in S^\#$, we say that

- (1) α is *enabled by T* if $R_\alpha \subseteq T$ and $I_\alpha \cap T = \emptyset$.
- (2) α is *enabled by T in sequential manner* if α is enabled by T with $|\alpha| = 1$.
- (3) α is *enabled by T in maximally parallel manner* if there is no $\beta \in A^\#$ such that $\alpha \subset \beta$, and α and β are enabled by T .
- (4) By $En_A^{sq}(T)$ and $En_A^{mp}(T)$, we denote the sets of all multisets of reactions $\alpha \in A^\#$ which are enabled by T in sequential manner and in maximally parallel manner, respectively.
- (5) The *results of A on T* , denoted by $Res_A^X(T)$ with $X \in \{sq, mp\}$, is defined as follows:

$$Res_A^X(T) = \{T - R_\alpha + P_\alpha \mid \alpha \in En_A^X(T)\}.$$

We note that $Res_A^X(T) = \{T\}$ if $En_A^X(T) = \emptyset$. Thus, if no multiset of reactions $\alpha \in A^\#$ is enabled by T , then T remains unchanged.

Definition 3. A *reaction automaton* (RA) \mathcal{A} is a 5-tuple $\mathcal{A} = (S, \Sigma, A, D_0, f)$, where

- S is a finite set, called the *background set of \mathcal{A}* ,
- $\Sigma (\subseteq S)$ is called the *input alphabet of \mathcal{A}* ,
- A is a finite set of reactions in S ,
- $D_0 \in S^\#$ is an *initial multiset*,
- $f \in S$ is a special symbol which indicates the final state.

Unlike the reaction system, a reaction automaton takes its input symbol from the environment into the current multiset (or state) representing current configuration, from time to time during the computing process. This idea was already considered and realized in the P automata theory (e.g., [5]).

Definition 4. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA, $w = a_1 \cdots a_n \in \Sigma^*$ and $X \in \{sq, mp\}$. An *interactive process in \mathcal{A} with input w in X manner* is an infinite sequence $\pi = D_0, \dots, D_i, \dots$, where

$$\begin{cases} D_{i+1} \in Res_A^X(a_{i+1} + D_i) & (\text{for } 0 \leq i \leq n - 1), \text{ and} \\ D_{i+1} \in Res_A^X(D_i) & (\text{for all } i \geq n). \end{cases}$$

In order to represent an interactive process π , we also use the ‘‘arrow notation’’ for $\pi : D_0 \xrightarrow{a_1} D_1 \xrightarrow{a_2} D_2 \xrightarrow{a_3} \cdots \xrightarrow{a_{n-1}} D_{n-1} \xrightarrow{a_n} D_n \rightarrow D_{n+1} \rightarrow \cdots$.

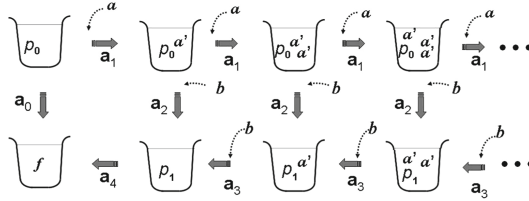


Fig. 1. An illustration of interactive processes for accepting the language $L = \{a^n b^n \mid n \geq 0\}$ in terms of the reaction automaton \mathcal{A} .

By $IP_X(\mathcal{A}, w)$ we denote the set of all interactive processes in \mathcal{A} with input w in X manner. If for an input string $w = a_1 \cdots a_n$, it is allowed that $a_i = \lambda$ for some $1 \leq i \leq n$, then an interactive process is said to be *with λ -input mode*. By $IP_X^\lambda(\mathcal{A}, w)$ we denote the set of all interactive processes in \mathcal{A} with λ -input mode in X manner for the input w ([11]).

For an interactive process π in \mathcal{A} with input w , if $En_A^X(D_m) = \emptyset$ for some $m \geq |w|$, then we have that $Res_A^X(D_m) = \{D_m\}$ and $D_m = D_{m+1} = \cdots$. In this case, considering the smallest m , we say that π *converges on D_m* (at the m -th step). If an interactive process π converges on D_m , then D_m is called the *converging state* of π and the successive multisets D_{m+1}, D_{m+2}, \dots are omitted.

Definition 5. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA and $X \in \{sq, mp\}$. The *languages accepted by \mathcal{A}* are defined as follows:

$$L_X(\mathcal{A}) = \{w \in \Sigma^* \mid \text{there exists } \pi \in IP_X(\mathcal{A}, w) \text{ such that } \pi \text{ converges on } D_m \text{ for some } m \geq |w| \text{ and } f \text{ is included in } D_m\}$$

$$L_X^\lambda(\mathcal{A}) = \{w \in \Sigma^* \mid \text{there exists } \pi \in IP_X^\lambda(\mathcal{A}, w) \text{ satisfying the same as } L_X(\mathcal{A})\}.$$

Example 1. Let us consider a reaction automaton $\mathcal{A} = (\{p_0, p_1, a, b, a', f\}, \{a, b\}, \{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}, p_0, f)$, where $\mathbf{a}_0 = (p_0, aba', f)$, $\mathbf{a}_1 = (p_0 a, b, p_0 a')$, $\mathbf{a}_2 = (p_0 a' b, \emptyset, p_1)$, $\mathbf{a}_3 = (p_1 a' b, a, p_1)$, $\mathbf{a}_4 = (p_1, aba', f)$. Figure 1 illustrates the whole view of possible interactive processes in \mathcal{A} with inputs $a^n b^n$ for $n \geq 0$. Let $w = aaabbb \in \Sigma^*$ be the input string and consider an interactive process π in sequential manner such that

$$\pi : p_0 \xrightarrow{a} p_0 a' \xrightarrow{a} p_0 a'^2 \xrightarrow{a} p_0 a'^3 \xrightarrow{b} p_1 a'^2 \xrightarrow{b} p_1 a' \xrightarrow{b} p_1 \rightarrow f.$$

It is easily seen that $\pi \in IP_{sq}(\mathcal{A}, w)$ and $w \in L_{sq}(\mathcal{A})$. Further, we see that $L_{sq}(\mathcal{A}) = \{a^n b^n \mid n \geq 0\}$ which is a context-free language.

We remark that this interactive process can be also performed by \mathcal{A} in maximally parallel manner, i.e. $\pi \in IP_{mp}(\mathcal{A}, w)$. In fact, it holds that $L_{mp}(\mathcal{A}) = \{a^n b^n \mid n \geq 0\}$.

3 Main Results on RAs

This section presents some results on reaction automata (RAs) that have been established in an earlier stage. In what follows, a language accepted by an RA is often referred to as an *RA language*.

3.1 Computing Powers of RAs and Their Subclasses

It has been proven that the accepting power of reaction automata with both manners of applying reactions coincides with that of Turing machines.

Theorem 1 ([9,10]). *The followings hold :*

- (1) *Every recursively enumerable language is accepted by a reaction automaton in maximally parallel manner.*
- (2) *Every recursively enumerable language is accepted by a reaction automaton in sequential manner with λ -input mode.*
- (3) *There exists a recursively enumerable language which cannot be accepted by any reaction automaton in sequential manner.*

The proofs are based on two facts: (i) every recursively enumerable language is accepted by a *restricted* two-stack machine (Theorem 8.13 in [7]) and (ii) one can devise an injective function to encode strings into multisets. The result (2) is in marked contrast to (3), which clarifies the computing power of λ -input mode in sequential manner.

The space complexity issues of reaction automata (RAs) have been considered. By restricting the volume of each multiset that is produced in an interactive process by RA, some subclasses of RAs were introduced and investigated on relations between classes of languages accepted by those subclasses of RAs and language classes in the Chomsky hierarchy.

Let \mathcal{A} be an RA and $X \in \{sq, mp\}$. Motivated by the notion of a workspace for a phrase-structure grammar, we define the counterpart of RA as follows: For $w \in L_X(\mathcal{A})$ with $n = |w|$, the *workspace of \mathcal{A} for w* is defined as:

$$WS(w, \mathcal{A}) = \min_{\pi} \{ \max_i \{ |D_i| \mid D_i \text{ appears in } \pi \}, \pi \in IP_X(\mathcal{A}, w) \}.$$

Definition 6. Let f be a function defined on \mathbf{N} and $X \in \{sq, mp\}$.

- (1) An RA \mathcal{A} is *$f(n)$ -bounded* if for any $w \in L_X(\mathcal{A})$ with $n = |w|$, $WS(w, \mathcal{A}) \leq f(n)$.
- (2) If a function $f(n)$ is a constant k (linear, exponential), then \mathcal{A} is termed constant-bounded (resp. linear-bounded, exponential-bounded).
- (3) The class of languages accepted by constant-bounded RAs (linear-bounded, exponential-bounded, arbitrary RAs) in X manner is denoted by $CoRA_X$ (resp. $\mathcal{LRA}_X, \mathcal{ERA}_X, \mathcal{RA}_X$).
- (4) The class of languages accepted by constant-bounded RAs (linear-bounded, exponential-bounded, arbitrary RAs) with λ -input mode in X manner is denoted by $CoRA_X^\lambda$ (resp. $\mathcal{LRA}_X^\lambda, \mathcal{ERA}_X^\lambda, \mathcal{RA}_X^\lambda$).

In order to explore and make clearer inclusion relations among language classes (considered so far), it is necessary to find a family of languages (or at least a particular language) with certain properties which plays a role of witness to distinguish one class from the other. The following lemma is useful for the purpose.

Lemma 1 ([9]). *Let Σ be an alphabet with $|\Sigma| \geq 2$ and $h : \Sigma^* \rightarrow \Sigma^*$ be an injection, and consider $L_h = \{wh(w) \mid w \in \Sigma^*\}$. Then, L_h is not in \mathcal{RA}_{sq} .*

Let us denote by \mathcal{REG} (\mathcal{CF} , \mathcal{CS} , \mathcal{RE}) the class of regular (resp. context-free, context-sensitive, recursively enumerable) languages.

Theorem 2 ([9, 10]). *The following inclusions hold :*

- (1). $\mathcal{REG} = \mathit{CoRA}_{mp} \subset \mathcal{LRA}_{mp} \subset \mathcal{ERA}_{mp} \subseteq \mathcal{RA}_{mp} = \mathcal{RE}$
- (2). $\mathcal{REG} = \mathit{CoRA}_{sq} \subset \mathcal{LRA}_{sq} \subset \mathcal{RA}_{sq} \subset \mathcal{RA}_{sq}^\lambda = \mathcal{RE}$.
- (3). $\mathcal{ERA}_{sq}^\lambda = \mathcal{ERA}_{mp} = \mathcal{CS}$.
- (4). \mathcal{CF} , \mathcal{LRA}_{mp} and \mathcal{RA}_{sq} are incomparable.

Thus, new characterizations of the classes \mathcal{REG} , \mathcal{CS} and \mathcal{RE} have been established in terms of the subclasses of RA languages CoRA_{mp} , \mathcal{ERA}_{mp} , \mathcal{RA}_{mp} , respectively. As seen later, however, the class \mathcal{CF} has been proved incomparable to *any* known class of languages defined by RAs so far, exhibiting a unique position within the RA language hierarchy.

3.2 Some Other Characterizations of RA Language Classes

One of the primary issues in the formal language theory is to investigate the closure properties of a language class under various language operations. When featuring the classes \mathcal{LRA}_{mp} and $\mathcal{LRA}_{mp}^\lambda$, the following has been proven.

Theorem 3 ([10, 11])

- (1). \mathcal{LRA}_{mp} is closed under union, intersection, concatenation, derivative, λ -free morphisms, λ -free gsm-mappings and shuffle, while **not** closed under complementation, quotient by regular languages, morphisms or gsm-mappings.
- (2). $\mathcal{LRA}_{mp}^\lambda$ is closed under union, intersection, concatenation, Kleene +, Kleene *, derivative, λ -free morphisms, inverse morphisms, λ -free gsm-mappings and shuffle.

We remark that in order to prove some of the negative closure properties of \mathcal{LRA}_{mp} , the following lemma is of crucial importance.

Lemma 2 ([10]). *For an alphabet Σ with $|\Sigma| \geq 2$, let $h : \Sigma^* \rightarrow \Sigma^*$ be an injection such that for any $w \in \Sigma^*$, $|h(w)|$ is bounded by a polynomial of $|w|$. Then, the language $L_h = \{wh(w) \mid w \in \Sigma^*\}$ is not in \mathcal{LRA}_{mp} .*

Further characterization results of \mathcal{RE} have been developed by using \mathcal{LRA}_{mp} and \mathcal{RA}_{sq} together with homomorphisms and regular languages.

Theorem 4 ([9, 11])

- (1). For any recursively enumerable language L , there exists an LRA \mathcal{A} such that $L = h(L_{mp}(\mathcal{A}))$ for some projection h .
- (2). For any recursively enumerable language L , there exists an LRA \mathcal{A} such that $L = R \setminus L_{mp}(\mathcal{A})$ (or $L_{mp}(\mathcal{A})/R$) for some regular language R .
- (3). For any recursively enumerable language L , there exists an RA \mathcal{A} such that $L = h(L_{sq}(\mathcal{A}))$ for some projection h .

4 Chemical Reaction Automata

As a simple and modified version of a reaction automaton, a *chemical reaction automaton* (CRA) has been introduced and investigated [12]. Rather lately, this computing model CRA turned out to be important, because it can provide an online computational model for a molecular programming language called Chemical reaction networks (CRNs [19]). It is known that CRNs involve wet implementations by a molecular reaction primitive called DNA strand displacement (DSD) systems.

Specifically, a CRA is a 5-tuple (S, Σ, A, D_0, F) , where each reaction in A is of the form (R, \emptyset, P) (each reaction in CRA has *no inhibitor*), and F is a *finite set* of final multisets. For convenience, each reaction in A is denoted by $R \rightarrow P$. In an interactive process of CRA, if $En_A^X(D) = \emptyset$, $Res_A^X(D)$ is undefined. A language accepted by a CRA $\mathcal{A} = (S, \Sigma, A, D_0, F)$ is defined by

$$L_X^\lambda(\mathcal{A}) = \{w \in \Sigma_\lambda^* \mid \pi : D_0 \rightarrow^{a_1} D_1 \rightarrow^{a_2} \dots \rightarrow^{a_n} D \in IP_X^\lambda(A, w), D \in F\}.$$

Remarks: The acceptance condition of CRA computation is slightly different from that of RA computation. A CRA accepts an input string if the final multiset *coincides* with an element of F , while an RA accepts an input string if the final multiset *includes* a particular symbol f . This difference is significant to obtain the results in our paper.

4.1 The Computation Power of CRAs

It has been shown that CRAs working in maximally parallel manner are computationally Turing universal. Our proof requires the fact that (i) a two-counter machine is equivalent to a Turing machine as a language accepting device [7] as well as the result that (ii) for any k -counter machine, there effectively exists an equivalent CRA. From (i) and (ii), the following is derived:

Theorem 5 ([12]). *The computational power of CRAs with λ -input mode in maximally parallel manner is equivalent to that of Turing machines.*

A naive question now arises: Concerning the computing power of CRAs whether or not there exists a real gap between working in maximally parallel manner and in sequential manner. This question is solved as a corollary of the next theorem.

Theorem 6 ([12]). *A language L is generated by a Petri net system if and only if L is accepted by a CRA with λ -input mode in sequential manner.*

Note that among several types of Petri net languages, here we mean a language of L^λ -type in [17]. Since the class of Petri net languages is strictly included in the class of context-sensitive languages and is incomparable to the class of context-free languages [17], it turns out that the computational power of CRAs with λ -input mode in sequential manner is *less* powerful than that of CRAs with λ -input mode in maximally parallel manner.

4.2 Determinism and Reversibility in CRAs

In this section we introduce the notions of determinism and reversibility into CRAs, and investigate the computational powers of those classes of CRAs in comparison with the language classes of Chomsky hierarchy.

The computing power of reversible CRAs involves the physical realization of molecular programming of chemical reaction networks with DNA strand displacement system implementation [19], and therefore, it is of great significance to elucidate the computing capabilities of both deterministic and reversible CRAs from the theoretical viewpoint of molecular computing.

Unlike the determinism of conventional computation models such as push-down automata, since a reactant is not divided into “input” part and “memory (multiset)” part, the determinism of CRAs cannot be decided only by a form of transition rules. This comes from the property of multiset memory, that is, from the current configuration alone, a CRA cannot identify a reactant of the next reaction to be applied. Therefore, the determinism of CRA has to be defined so as to exclude any branching computation, regardless of a non-empty input or empty input.

A CRA is said to be *deterministic* if for every input symbol $a \in \Sigma$ and every reachable configuration, the resultant multiset after a reaction is unique. Similar to the definition of deterministic pushdown automata, this condition is extended to the case of λ -input mode.

It is not trivial, only from their definitions, to recognize the difference of computing powers of the determinism between realtime CRA and CRA with λ -input mode, where “realtime” means “no λ input is allowed”. The following result might be rather unexpected in some sense.

Lemma 3 ([15]). *If a language L is accepted by a DCRA with λ -input mode, then L is also accepted by a realtime DCRA.*

Note that by our definition of DCRA, without receiving an input symbol, no configuration of a DCRA with λ -input mode can have an enabled reaction, while this is not the case for a realtime DCRA. In order to make a reaction, a realtime DCRA always requires an input symbol, even if its configuration has an enabled reaction with no input symbol.

Information preserving computations (forward and backward deterministic computations) are very important and are considered as “reversibility” in

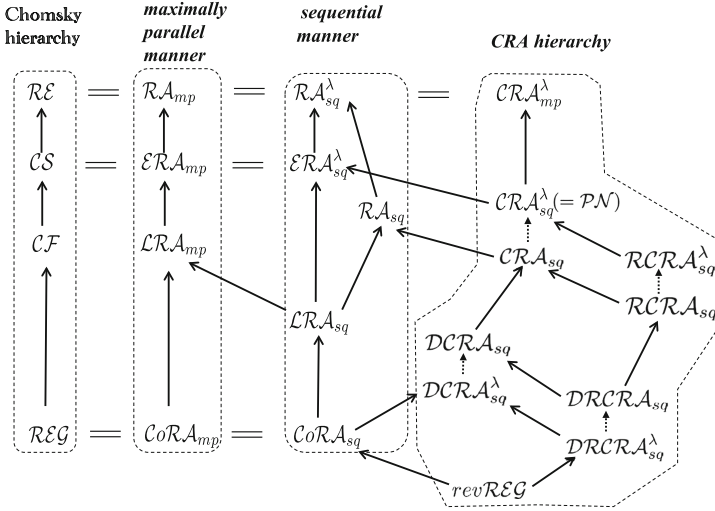


Fig. 2. Inclusion relations among a variety of RA language classes at large.

many existing research papers (e.g., [2,3]). However, in order to understand their properties in more details, we want to take a position to think of them apart(i.e., separate into forward and backward determinisms). Thus, in our view the “reversibility” simply means that the previous configuration of computation can be uniquely determined (backward determinism).

A CRA is said to be *reversible* if for every input symbol $a \in \Sigma$ and every reachable configuration D , the set of configurations which directly reaches D with a is a singleton. For the case of λ -input mode, this condition is extended to the case of λ -input mode in a natural manner.

The following lemma holds true for a deterministic and reversible CRA (abbrev. DRCRA).

Lemma 4 ([15]). *If a language L is accepted by a DRCRA with λ -input mode, then L is also accepted by a realtime DRCRA.*

By CRA_{sq} , CRA_{sq}^λ , $DCRA_{sq}$, $DCRA_{sq}^\lambda$, $RCRA_{sq}$, $RCRA_{sq}^\lambda$, $DRCRAsq$, and $DRCRAsq^\lambda$, we denote the classes of languages accepted by realtime CRAs, CRAs with λ -input mode, realtime DCRAs, DCRAs with λ -input mode, realtime RCRAs, RCRAs with λ -input mode, realtime deterministic and reversible CRAs (DRCRAs), DRCRAs with λ -input mode, respectively.

Remarks: Due to the space limitation, most of the details on the results of DCRAs and RCRAs are omitted and the reader is advised to refer to [15]. Instead, Fig. 2 summarizes the inclusion relations among various classes of CRA languages discussed in this paper, where \mathcal{PN} is the class of Petri net languages of L^λ -type [17] and $revREG$ is the class of zero-reversible regular languages [1].

Last but not least, it should be noted the following:

- (1). Another type of “finite automata with multiset memory (FAMMs)” was proposed and investigated in [14] which employs a rule application mode similar to (but different from) that of RAs, and with FAMM framework a new characterization of Chomsky hierarchy was established.
- (2). The reader is kindly advised to refer to another survey paper on reaction automata theory [13] for more details on the results and discussion left out in this paper because of the space limit.

5 Future Research Topics

Many subjects remain to be investigated along the research direction suggested by reaction automata.

- **Inclusion relations and Computing powers:**
 - Further refinements of the hierarchy of RA language classes and CRA language classes are strongly encouraged to clarify the inclusion relations in Fig. 2.
 - It is of great importance to explore the relationships between subclasses of \mathcal{RA} and others defined by computing devices based on the multiset rewriting, such as a variety of P-systems and their variants (e.g., P automata and dP automata [5, 16]).
 - It is also intriguing to clarify the relationships between subclasses of CRAs studied here and others defined by reversible computing models such as reversible pushdown automata [8].
 - It remains open whether or not deterministic CRAs (with/without λ -input mode) in maximally parallel manner are Turing universal.
- **Complexity issues:** There remain open time complexity issues to be studied in the hierarchies of \mathcal{RA} and \mathcal{CRA} classes. For example, no efforts have been made yet for investigating the *time* complexity of any class from the hierarchies.
- **Decision problems:** One may be encouraged to study a variety of the decision problems on subclasses within \mathcal{RA} hierarchy. For example, it is an interesting question to explore the equivalence problem for the class \mathcal{LRA} or the classes of deterministic/reversible CRAs.
- **Other issues:** It would also be useful to develop methods for simulating a variety of chemical reactions in the real world application, by the use of the framework based on reaction automata. For that purpose, investigating stochastic models based on RAs has to be conducted, and such stochastic versions of RAs may provide useful simulation tools for analyzing any natural phenomena modeled by RAs.

Finally, considering the natural correspondence to (or analogy of) classic theory of automata, we conclude this survey with our firm belief that Reaction Automata are computational devices which deserve much further research efforts.

References

1. Angluin, D.: Inference of reversible languages. *J. ACM* **29**(3), 741–765 (1982)
2. Alhazov, A., Freund, R., Morita, K.: Sequential and maximally parallel multiset rewriting: reversibility and determinism. *Nat. Comput.* **11**, 95–106 (2012)
3. Bennett, C.H.: Logical reversibility of computation. *IBM J. Res. Dev.* **17**(6), 525–532 (1973)
4. Calude, C.S., Păun, G., Rozenberg, G., Salomaa, A. (eds.): WMC 2000. LNCS, vol. 2235. Springer, Heidelberg (2001). <https://doi.org/10.1007/3-540-45523-X>
5. Cshaj-Varju, E., Vaszil, G.: P automata. In: *The Oxford Handbook of Membrane Computing*, pp. 145–167 (2010)
6. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. *Fundam. Inform.* **75**, 263–280 (2007)
7. Hopcroft, J.E., Motwani, T., Ullman, J.D.: *Introduction to Automata Theory, Language and Computation*, 2nd edn. Addison-Wesley, Boston (2003)
8. Kutrib, M., Malcher, A.: Reversible pushdown automata. *J. Comput. Syst. Sci.* **78**, 1814–1827 (2012)
9. Okubo, F.: Reaction automata working in sequential manner. *RAIRO Theor. Inform. Appl.* **48**, 23–38 (2014)
10. Okubo, F., Kobayashi, S., Yokomori, T.: Reaction automata. *Theor. Comput. Sci.* **429**, 247–257 (2012)
11. Okubo, F., Kobayashi, S., Yokomori, T.: On the properties of language classes defined by bounded reaction automata. *Theor. Comput. Sci.* **454**, 206–221 (2012)
12. Okubo, F., Yokomori, T.: The computational capability of chemical reaction automata. In: Murata, S., Kobayashi, S. (eds.) *DNA 2014*. LNCS, vol. 8727, pp. 53–66. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11295-4_4. Also, in *Natural Computing*, vol. 15, pp. 215–224 (2016)
13. Okubo, F., Yokomori, T.: Recent developments on reaction automata theory: a survey. In: Suzuki, Y., Hagiya, M. (eds.) *Recent Advances in Natural Computing*. MI, vol. 9, pp. 1–22. Springer, Tokyo (2015). https://doi.org/10.1007/978-4-431-55105-8_1
14. Okubo, F., Yokomori, T.: Finite automata with multiset memory: a new characterization of chomsky hierarchy. *Fundam. Inform.* **138**, 31–44 (2015)
15. Okubo, F., Yokomori, T.: The computing power of determinism and reversibility in chemical reaction automata. In: Adamatzky, A. (ed.) *Reversibility and Universality*. ECC, vol. 30, pp. 279–298. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73216-9_13
16. Paun, G., Rozenberg, G., Salomaa, A.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York (2010)
17. Peterson, J.L.: *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, Englewood Cliffs (1981)
18. Rozenberg, G., Back, T., Kok, J.N. (eds.): *Handbook of Natural Computing*. Section IV: Molecular Computation, vol. 3, pp. 1071–1355. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-3-540-92910-9>
19. Thachuk, C., Condon, A.: Space and energy efficient computation with DNA strand displacement systems. In: Stefanovic, D., Turberfield, A. (eds.) *DNA 2012*. LNCS, vol. 7433, pp. 135–149. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32208-2_11

Author Index

- Alaev, Pavel 20
Ambos-Spies, Klaus 30
Antunes, Luís 397
- Bazhenov, Nikolay 40
Becker, Florent 50
Benzmüller, Christoph 60
Berger, Ulrich 70
Berndt, Sebastian 81, 89
Beros, Achilles A. 97
Bonizzoni, Paola 107
- Carl, Merlin 118, 126
Cenzer, Douglas 136
- Davis, Martin 146
Della Vedova, Gianluca 107
- Erlebach, Thomas 156
- Fernau, Henning 161, 172
Fluschnik, Till 161, 183
- Georgiev, Ivan 194
Graça, Daniel S. 204
- Hermelin, Danny 161
Hoyrup, Mathieu 214
- Khan, Mushfeq 97
Khoussainov, Bakh 1
Kjos-Hanssen, Bjørn 97
Klein, Kim-Manuel 89
Kones, Ishai 224
Korovina, Margarita 234
Krebs, Andreas 161
Kristiansen, Lars 194, 244
Kudinov, Oleg 234
Kuppusamy, Lakshmanan 172
- Lauria, Massimo 254
Levin, Asaf 224
- Maldonado, Diego 50
Marchuk, Margarita 40
Marques-Silva, Joao 264
Mateus, Paulo 397
McNicholl, Timothy H. 277
Mertzios, George B. 183
Milovanov, Alexey 287
Molter, Hendrik 161
Monath, Martin 297
Murwanashyaka, Juvenal 244
- Nicholson, Marie 307
Nichterlein, André 183
Nicosia, Serena 107
Niedermeier, Rolf 161
Nies, André 97
Normann, Dag 318
- Okubo, Fumiya 421
Oladele, Rufus O. 172
Ollinger, Nicolas 50
Ouazzani, Sabrina 126
- Parent, Xavier 60
Pauly, Arno 328
Petrovska, Olga 70
Pirola, Yuri 107
Previtali, Marco 107
- Quinon, Paula 338
- Rizzi, Raffaella 107
Rojas, Diego A. 136
Rossegger, Dino 349
Rybalov, Alexander 359
- Sanders, Sam 365
Selivanov, Victor L. 376
Selivanov, Victor 20
Selivanova, Svetlana V. 376
Shen, Alexander 410
Sieg, Wilfried 386

Souto, André [397](#)
Stephan, Frank [194](#)
Stull, Don [277](#)

Teixeira, Andreia [397](#)
Theyssier, Guillaume [50](#)

Uspenskiy, Vladimir [410](#)

van der Torre, Leendert [60](#)

Welch, Philip [126](#)

Yokomori, Takashi [421](#)

Zhong, Ning [204](#)