Lecture Notes in Computer Science 3526

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

Editorial Board

S. Barry Cooper Benedikt Löwe
Leen Torenvliet (Eds.)

New Computational Paradigms

First Conference on Computability in Europe, CiE 2005
Amsterdam, The Netherlands, June 8-12, 2005
Proceedings

Volume Editors

S. Barry Cooper
University of Leeds
School of Mathematics
Leeds, LS2 9JT, UK
E-mail: pmt6sbc@maths.leeds.ac.uk

Benedikt Löwe
Leen Torenvliet
Universiteit van Amsterdam
Institute for Logic, Language and Computation
Plantage Muidergracht 24, 1018 TV Amsterdam, NL
E-mail: {bloewe,leen}@science.uva.nl

Library of Congress Control Number: 2005926498

CR Subject Classification (1998): F.1.1-2, F.2.1-2, F.4.1, G.1.0, I.2.6, J.3

ISSN 0302-9743
ISBN-10 3-540-26179-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26179-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11494645 06/3142 5 4 3 2 1 0

Preface

CiE 2005: New Computational Paradigms

`http://www.illc.uva.nl/CiE/`

The cooperation *Computability in Europe* (**CiE**) is an informal European network covering computability in theoretical computer science and mathematical logic, ranging from application of novel approaches to computation to set-theoretic analyses of infinitary computing models. The cooperation consists of eleven main nodes and includes over 400 researchers; it is coordinated from Leeds (UK). More information about **CiE** can be found in Barry Cooper's introductory paper to this volume (p. 1) and at

`http://www.amsta.leeds.ac.uk/pure/staff/cooper/cie.html`

CiE 2005 was a conference on the special topic "New Computational Paradigms" and was held in Amsterdam in June 2005. It was initiated by and served as a focus point for the informal cooperation **CiE**. The topic of "New Computational Paradigms" covers connections between computation and physical systems (e.g., quantum computation, neural nets, molecular computation) but also higher mathematical models of computation (e.g., infinitary computation or real computation).

Computability theory is central to large areas of theoretical computer science and mathematical logic. Traditionally, the computational model of the Turing machine (or mathematically equivalent models) has been used to reason about computation or computability. For general computability inquiries (with unbounded resources), the choice of the model of computation hardly matters (this fact is encapsulated in the so-called "Church-Turing thesis"); this could change as soon as questions of efficiency are investigated. In all areas of computability theory, alternative models of computation have been investigated, ranging from the most abstract (generalized recursion theory, Infinite Time Turing Machines) to the very concrete (physical constructions of quantum computers, applications in neuroscience and learning theory, fine-grained parallelism, swarm intelligence, neural nets, agents, games).

We understand **CiE 2005** as an interdisciplinary venue for researchers from computer science and mathematics to exchange ideas, approaches and techniques

in their respective work, thereby generating a wider community for work on new computational paradigms that allows uniform approaches to diverse areas, the transformation of theoretical ideas into applicable projects, and general cross-fertilization transcending disciplinary borders. The goal to reach out to both computer scientists and mathematicians resulted in many surprises for both communities: after many years and decades of experience in academia, we were astonished about the differences in general practice between two research communities that seem to be so close in content:

- Mathematicians issue a *Call for Papers* at a conference asking for submissions and publish a proceedings volume after the conference (in most cases, these will not appear until several years after the conference); in contrast, computer scientists distribute a proceedings volume at a conference.
- As a consequence, mathematical conferences have a submission deadline for abstracts about six to eight weeks before the conference, and computer science conferences need about half a year to prepare the proceedings volume.
- Also, the typical mathematical submission to a conference is between two lines and half a page describing the general idea of the talk, whereas in computer science submissions are papers of ten pages length.
- It is customary in mathematics that submitted abstracts are just checked cursorily to weed out nonsensical submissions – as a consequence, the acceptance rate is typically between 95% and 100%, the real quality control happens after the conference when the submissions for the proceedings volume are refereed up to journal standards; in computer science, acceptance rates have to be below 50%, and the lower the acceptance rate the higher the quality of the conference.
- For a mathematician, it is only invited talks and publications that count as an indicator of academic standing, a standard mathematical CV of a reasonably senior researcher wouldn't even mention contributed talks at conferences (many mathematicians consider submissions to conferences as something that only PhD students and recent graduates would do); in computer science, the list of conferences at which you have presented papers is a prime indicator of your success.

Depending on whether the reader is a mathematician or computer scientist, he or she will have nodded at some of the points just mentioned and wrinkled his or her forehead at others. One of the biggest surprises of our work was how little the two communities know about the practice of the other. Our Programme Committee consisted of mathematicians and computer scientists, and all members spent a considerable amount of time trying to understand the workings of the other community, with all the puzzlement, perplexity, irritations, and the exciting impression of pioneering that this involves.

It is obvious that bringing together two communities with so diametrically opposed ways of organizing conferences would require compromises and a lot of innovative ideas on how to make things work. We tried to walk on the ridge between the two communities and accommodate the needs of both.

As a consequence, this volume only contains the invited papers and a selection of 47.2% of the submitted papers. This may sound like a high percentage to computer scientists, but has to be seen in the context of the intercommunity aspect of this conference project. As a result, we are proud to have produced such a high-quality volume that truly stands as a testament to the interdisciplinary nature of the conference. In the style of mathematics conferences, there will be several *postproceedings* publications that will be reviewed according to journal standards. There will be a monograph of edited papers entitled *New Computational Paradigms* (edited by B. Cooper, B. Löwe, and A. Sorbi) that contains surveys and expository papers. For the research papers, we will have three special issues of journals: a special issue of *Theoretical Computer Science C* (edited by T. Bäck and B. Löwe) focusing on computation and the natural sciences, a special issue of *Mathematical Structures in Computer Science* (edited by B. Cooper, B. Löwe, and D. Normann) focusing on the more mathematical aspects, and a special issue of *Theory of Computing Systems* (edited by B. Cooper, B. Löwe, and P. van Emde Boas). We shall invite the authors of the best papers at the conference to submit their full versions to these special issues and referee them to the high standards of the respective journals.

For the most current information about the conference, we refer the reader to our webpage

`http://www.illc.uva.nl/CiE/`

The **CiE** enterprise will continue to thrive: we already received enthusiastic feedback from the members of the **CiE** cooperation, and **CiE 2005** developed into a conference series that is already being planned for the years 2006 to 2009. The next **CiE** conference is planned for Swansea (UK) from June 30 to July 5, 2006; after that, we'll reconvene in Siena (Italy) in June 2007. This volume is just the beginning of a larger project bringing mathematics and computer science back together by trying to accommodate the special styles of both communities. We felt that there was a need for this, and the positive feedback proves that we were right.

Scientific Structure of the Conference

The template for the scientific organization of **CiE 2005** was the meetings of the *Association for Symbolic Logic* with tutorials, plenary talks and special sessions. The Programme Committee invited speakers for two tutorials (three hours each), eight plenary talks (one hour each) and organizers for six special sessions (with four talks of half an hour each). The special session organizers were then asked to invite their four speakers. All 35 invited speakers were asked to contribute either an abstract or a paper to this proceedings volume, and most did.

Our tutorial speakers were Harry Buhrman (Universiteit van Amsterdam, CWI; p. 68) and Klaus Weihrauch (FernUniversität Hagen; p. 530). Plenary talks were given by Samson Abramsky (Oxford University), Joel D. Hamkins (City University of New York; p. 180), Ulrich Kohlenbach (Technische Universität

Darmstadt; p. 233), Jan van Leeuwen (Universiteit Utrecht), Yuri Matiyasevich (Steklov Institute of Mathematics; p. 310), Yiannis Moschovakis (Ethnikon and Kapodistriakon Panepistimion Athinon, and University of California at Los Angeles: p. 350), Gheorghe Paun (The Romanian Academy, Bucharest; p. 396), and Uwe Schöning (Universität Ulm; p. 429).

The topics of the six special sessions were selected by the Programme Committee in a way to involve mathematical logic and computer science at the same time as offering the methodological foundations for models of computation. The speakers were invited by the special session organizers:

Special Session on Biological Computation, *organized by T. Bäck (Leiden):*

Paola Bonizzoni (Milan; with Clelia De Felice & Giancarlo Mauri; p. 65), Marian Gheorghe (Sheffield; with Francesco Bernardini, Natalio Krasnogor, German Terrazas; p. 49), Tero Harju (Turku; p. 188), Natalio Krasnogor (Nottingham; with German Terrazas, Francesco Bernardini, Marian Gheorghe, Steve Diggle, Miguel Cámara; p. 479).

Special Session on Complexity, *organized by Elvira Mayordomo Cámara (Zaragoza):*

Ricard Gavaldà (Barcelona/Montréal, QC; p. 150), Jack Lutz (Ames, IA; p. 299), Peter Bro Miltersen (Aarhus; p. 342), Jacopo Torán (Ulm; p. 495).

Special Session on Epistemology and Methodology of Computing, *organized by Hartmut Fitz (Amsterdam) and Guglielmo Tamburrini (Pisa):*

Angelo Cangelosi (Plymouth; p. 69), Artur d'Avila Garcez (London; p. 139), Wilfried Sieg (Pittsburgh, PA; p. 440), Giuseppe Trautteur (Naples; p. 507).

Special Session on Proofs and Computation, *organized by A. Beckmann (Swansea) and L. Crosilla (Florence):*

Ulrich Berger (Swansea; p. 23), Thierry Coquand (Göteborg; p. 86), Jan Johannsen (Munich), Stan Wainer (Leeds; with Geoff E. Ostrin; p. 378).

Special Session on Real Computation, *organized by A. Edalat (London):*

Amin Farjudian (Tehran; p. 128), André Lieutier (Aix-en-Provence/Grenoble; p. 297), Milad Niqui (Nijmegen; p. 368), Dirk Pattinson (Munich; p. 385), Ning Zhong (Cincinnati, OH; p. 552).

Special Session on Relative Computability, *organized by B. Cooper (Leeds) and A. Sorbi (Siena):*

Denis Hirschfeldt (Chicago, IL; p. 209), Iskander Kalimullin (Kazan; p. 221), Andrew E. M. Lewis (Leeds; p. 275), Andrey Morozov (Novosibirsk; p. 349).

A number of 62 contributed talks were accepted for presentation at this conference of which the best appear in the volume. In addition to this, we allowed researchers to announce *informal talks* in the style of mathematics conferences. Abstracts or extended abstracts of all talks not included in this volume will be published in a booklet appearing in the *ILLC Publications*.

Among the papers that will appear in that additional booklet are the following:

- José L. Balcazar, *Query Learning of Horn Formulas Revisited*;
- Giulia Battilotti, Paola Zizzi, *The Internal Logic of Bell's States*;
- Yi-Xiang Chen, Jie Zhou, *Fuzzy Interval-Valued Processes Algebra*;
- Carmen Graciani, Agustín Riscos-Núñez, *Looking for Simple Common Schemes to Design Recognizer P Systems with Active Membranes That Solve Numerical Decision Problems*;
- Vince Grolmusz, *Defying Dimensions Modulo 6*;
- Miguel Ángel Gutiérrez-Naranjo, Mario Pérez-Jiménez, Francisco José Romero-Campero, *Solving* **SAT** *with Membrane Creation*;
- Montserrat Hermo, Joxe Gaintzarain, Marisa Navarro, *Learning Conjunctions of* $\text{Horn}^{\supset}$ *Clauses*;
- Eiju Hirowatari, Kouichi Hirata, Tetsuhiro Miyahara, Setsuo Arikawa, *On the Prediction of Recursive Real-Valued Functions*;
- Paulin Jacobe de Naurois, Olivier Bournez, Felipe Cucker, Jean-Yves Marion, *Logical Characterizations of* **P** *and* **NP** *over an Arbitrary Structure K*;
- Viv Kendon, William J. Munro, *Entanglement and Its Role in Shor's Algorithm*;
- Tien D. Kieu, *Hypercomputability in Quantum Mechanics*;
- Branimir Lambov, *Complexity in a Type-1 Framework for Computable Analysis*;
- Pierluigi Minari, *Proof-Theoretical Methods in Combinatory Logic and Lambda-Calculus*;
- Erich Monteleone, *On the Infinitary Formal Systems and Infinite Time Turing Machines*;
- Marcin Mostowski, *Potential Infinity and the Church Thesis*;
- Benedek Nagy, *An Interval-Valued Computing Device*;
- Stela Nikolova; *On the Notion of* $\forall$*-definedness of Non-deterministic Programs*;
- Harumichi Nishimura, Tomoyuki Yamakami, *Quantum Minimal One-Way Information for Combinatorial Tasks*;
- Igor Potapov, Oleksiy Kurganskyy, *Universality of Walking Automata on a Class of Geometric Environments*;
- Dimiter Skordev, *A Computability Notion for Locally Finite Lattices*;
- Boris Solon, *Non-total Enumeration Degrees*;
- Haibin Sun, Wenhui Li, *Rules-Based Spatial Reasoning Combining Topological and Cardinal Directional Relations*;
- John V. Tucker, Edwin Beggs, *Newtonian Systems, Bounded in Space, Time, Mass and Energy Can Compute All Functions*;
- Raymond Turner, *Computability in Specification*;
- Puzarenko Vadim, *Computable Principles in Admissible Structures*;
- Andreas Weiermann, *A Very Slow Growing Hierarchy for the Howard Bachmann Ordinal*;
- Paola Zizzi, *Computability at the Planck scale.*

Organization and Acknowledgements

The conference **CiE 2005** was organized by Stefan Bold (Amsterdam), Barry Cooper (Leeds), Peter van Emde Boas (Amsterdam), Samson de Jager (Amsterdam), Benedikt Löwe (Amsterdam), Leen Torenvliet (Amsterdam), and Marjan Veldhuisen (Amsterdam).

The Programme Committee consisted of K. Ambos-Spies (Heidelberg), A. Atserias (Barcelona), J. van Benthem (Amsterdam), B. Cooper (Leeds, *Chair*), P. van Emde Boas (Amsterdam), S. Goncharov (Novosibirsk), B. Löwe (Amsterdam, *Chair*), D. Normann (Oslo), H. Schwichtenberg (Munich), A. Sorbi (Siena), I. Soskov (Sofia), L. Torenvliet (Amsterdam), J. Tucker (Swansea), and J. Wiedermann (Prague).

The conference was sponsored by the *Koninklijke Nederlandse Akademie van Wetenschappen* (KNAW), the *Nederlandse Organisatie voor Wetenschappelijk Onderzoek* (NWO), the *European Association for Theoretical Computer Science* (EATCS) and the *Association for Symbolic Logic* (ASL). We would like to thank the organizers of the special sessions for the communications between the editors and the invited special session speakers. For help and support with various matters during the production of this volume, we would like to thank the Institute for Logic, Language and Computation (ILLC), Stefan Bold, Christine Günther (Springer), Tanja Kassenaar, Anna Kramer (Springer), Peter van Emde Boas, Ingrid van Loon, Wil van Zijl, Marjan Veldhuisen, and Marco Vervoort (GreenLight Solutions).

The high quality of the volume was achieved through the hard work of the members of the Programme Committee, with the help of external referees, among them Inge Bethke, Sander M. Bohte, Robert Brijder, Erzsébet Csuhaj-Varjú, Giovanni Curii, Ronald de Wolf, Pascal Hitzler, Mojmir Kretinsky, Antonin Kucera, Clemens Kupke, Wolfgang Merkle, Michael Muskulus, Martin Pelikan, Jan Reimann, Hans Ulrich Simon, Petr Sosik, Sebastiaan Terwijn, Tereza Tusarova, and Duong Vu.

Last, but definitely not least, we would like to thank Samson de Jager (Amsterdam) who was the main layouter and typesetter for this volume and communicated with the authors and editors on all TEXnical matters.

Amsterdam and Leeds
March 2005

S.B. Cooper
B. Löwe
L. Torenvliet

Table of Contents

If CiE Did not Exist, It Would be Necessary to Invent It *

S. Barry Cooper

School of Mathematics, University of Leeds,
Leeds LS2 9JT, U.K
pmt6sbc@leeds.ac.uk
http://www.maths.leeds.ac.uk/~pmt6sbc

As it happens, "Computability in Europe" *was* invented, just over two years ago, and in a short time has grown beyond all expectations. But even though the surprise of finding together so many researchers into different aspects of computability has not worn off, **CiE** does represent a strand of scientific endeavour going back to the earliest times. Even before Euclid of Alexandria devised his algorithm for finding the greatest common divisor of two integers, human survival depended on the identification of *algorithmic content* in the everyday world. What distinguished Euclid, and successors like Newton, Leibniz, Frege, Peano, Babbage, Russell, Hilbert, Gödel and Turing, is the reaching for control over that content through theory and abstraction. Perhaps Albert Einstein had something like this in mind in 1950 when he wrote (p.54 of *Out of My Later Years*, Philosophical Library, New York):

> "When we say that we understand a group of natural phenomena, we mean that we have found a constructive theory which embraces them."

What is peculiarly contemporary about **CiE** is the scrutiny it brings to bear on the *quest* for algorithmic content, something that was not possible before Turing and his fellow 1930s pioneers in the area.

Through the work of computability theorists, the search for algorithmic content goes beyond the ad hoc, and develops into an activity guided by an expanded *consciousness* of what we are doing. We can now explain why certain basic problems are harder than others. We can use our knowledge of logical structure and language to devise more efficient computer programs. We can relate the structures of computability theory to real-world situations, and find models which aid prediction, or make problems in making predictions mathematically explicit. And, the hope is, we can get enough insight into how physical systems 'compute' to ease us past the computational barriers our theory has brought to our notice. The questions surrounding 'New Computational Paradigms' are indeed fundamental ones. Answers, as so often in the past, will depend on the sort of mix of the practical and the theoretical that Alan Turing, if he were still with us, would have recognised, and found fascinating.

* With apologies to Voltaire ...

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 1–7, 2005.

In current terminology, the scientific approach of **CiE** is interdisciplinary, approaching real-world problems from different perspectives and using diverse techniques. **CiE** seeks to bridge the theoretical divide between mathematics and computer science, and between computability theory and science, which are traceable back almost to the birth of computability theory in the mid-1930s. Of course, the natural scientist of the Enlightenment would have had no problem with the so-called interdisciplinarity of **CiE**. It is only since the sixteenth and seventeenth centuries that scientific specialisms have solidified into exclusive disciplines, regulated by senior figures whose role it is to perseverate the assumptions and conventions of their areas, complete with their own technical priorities.

Even 'new paradigms' constitutes a project started by Turing — the natural scientist par excellence, at least to the computability theorist. On the one hand, the yawning gap between computation and the real-world played a key role in both his scientific and personal lives, just as it now dominates the work of **CiE**. This was a gap he was ever, both practically and conceptually, seeking to bridge, and many of his ideas anticipate current research. On the other hand, this was a preoccupation which took him — and now promises to take us — beyond the safe confines of what Thomas Kuhn calls 'normal science'. Here is how Kuhn describes normal science in his influential book *The Structure of Scientific Revolutions* (pp.162–164 of the Third Edition, The University of Chicago Press, 1996):

> "Normally, the members of a mature scientific community work from a single paradigm or from a closely related set. . . . once the reception of a common paradigm has freed the scientific community from the need constantly to re-examine its first principles, the members of that community can concentrate exclusively upon the subtlest and most esoteric of the phenomena that concern it. Inevitably, that does increase both the effectiveness and the efficiency with which the group as a whole solves new problems."

What has become problematic in many areas of science and the humanities is dealing with globally determined phenomena. Everywhere, we see nonlinear development, breakdown in inductive and predictive structures, computer simulation replacing mathematical solutions, and the puzzle of emergence of new relations in the midst of turbulence. We also see the ad hoc development of particular solutions to everyday problems which seem to challenge existing conceptual frameworks. And we have quite basic obstacles to raising the capabilities of present-day computers to the needs of the working scientist. All this is reflected in the wide variety of theoretical directions to be found within **CiE**. The aim is to bring a new understanding to existing developments, and to establish the sort of consciousness of computational issues upon which exciting new practical innovations can be based.

However, the reader coming to this volume for the weird and wonderful from today's scientific fringe will be disappointed. There are indeed contributions which acknowledge the extent to which the Turing machine paradigm is already

shifting — see, for instance, Dina Goldin and Peter Wegner on *The Church-Turing Thesis: Breaking the Myth* — but this tends to be work which has gone through a long period of gestation, and received a measure of acceptance and respect within the computer science community. Here is how one of our reviewers of Goldin and Wegner's article described the situation:

> "Fifty years ago the Turing Thesis was OK, now it is still OK provided we know to what computational scenario it should be related. If we are changing the scenario (as the practice of computing prompts us), we have to update the notion of a Turing machine (or of any other fundamental model of computation) as well - that's all."

Of course, that may be 'all'. But extracting useful models from new computational situations, or — to approach things more theoretically — to develop new abstractions and make innovative real-world connections, is the essence of the challenge. So we also see here papers dealing with more overtly mathematical extensions of the standard Turing model of computation, such as infinite time Turing machines — see Joel Hamkins on *Infinitary computability with infinite time Turing machines*, and the paper of Philip Welch — and, at the other extreme, a number of contributions dealing with natural computation. Membrane computing is represented by Gheorghe Păun (a seminal figure in this area), Marian Gheorghe et al, and Shankara Narayanan Krishna, and we have Paola Bonizzoni, Felice, and Mauri on DNA computing, and Natalio Krasnogor (with Gheorghe again, and others) on computational modelling of the important microbiological phenomenon of 'quorum sensing'. The resurgent topic of analog computers is touched on by Jérôme Durand-Lose, Giuseppe Trautteur (on *Beyond the Super-Turing Snare: Analog Computation and Virtual Digitality*) and Jeffery Zucker with John Tucker, and neural networks, another area with a long history, is represented by Angelo Cangelosi, and Krzysztof Michalak with Halina Kwasnicka. And, of course, quantum computation is a key topic at **CiE 2005** (where from Harry Buhrman we get just a taster from his talks). New paradigms of computation come in all shapes and sizes, and the growth of quantum computing reminds us that even quite modest improvements (theoretically speaking) in computing efficiency promise big changes in the world we live in.

Although a number of people associated **CiE** are involved with these different areas, most of us do not actually set out to *do* quantum computing, membrane computing, neural networks, evolutionary computation, and so on. The agenda is not so piecemeal. This is not computational tourism, the Readers Digest Condensed Books version, with the grown-ups head for the real thing — WCIT, CINC, CEC, and other myriad specialist meetings. The intervention of **CiE** is aimed at using logical and mathematical methods to reveal underlying structures and unities, to develop general frameworks and conceptual aids — to build the sort of theoretical and practical synergies which gave Turing and von Neumann such a key role in the early days of the first computing revolution. For example, one can recognise within most of the existing proposals for new computational paradigms a high degree of interactivity, as is picked up on in Goldin and Wegner's paper. One aspect of this is the importance now given to connectionist

models of computation, anticipated by Turing himself in his discussion of 'unorganised machines' in his 1948 National Physical Laboratory Report *Intelligent machinery*. In 1988 Paul Smolensky observed in his influential *Behavioral and Brain Sciences* paper *On the proper treatment of connectionism* (p.3) that:

> "There is a reasonable chance that connectionist models will lead to the development of new somewhat-general-purpose self-programming, massively parallel analog computers, and a new theory of analog parallel computation: they may possibly even challenge the strong construal of Church's Thesis as the claim that the class of well-defined computations is exhausted by those of Turing machines."

This kind of challenge is being met on a number of levels. On the one hand one has the work on neural networks and logic reported on by Artur Garcez in this volume, while on the other one has models based on computational reducibilities (such as that derived from Turing's oracle machines), which promises a new relevance to the sort of classical computability featuring in the contributions from Finkel, Harris, Kalimullin, Lewis, Angsheng Li, Selivanov, Soskov, Alexandra Soskova, and Terwijn (describing an interesting application of the Medvedev lattice). As our anonymous reviewer pointed out above, "Fifty years ago the Turing Thesis was OK, now it is still OK provided we know to what computational scenario it should be related." The world has not lost its algorithmic content, but there is needed a fundamental process of readjustment to new realities, involving full use of our powers of theoretical deconstruction of computationally complex environments. Relevant here is the paper of Udi Boker and Nachum Dershowitz. As one of our reviewers commented: "The subject of this paper is very central to the topic of the conference: How can we compare the computational power of different computational paradigms?".

As already suggested, we can think of **CiE** as *computation with consciousness*, in the sense that there is a relatively high level of detachment and abstraction involved. Wonderful things can be achieved without consciousness. Bert Hölldobler and Edward O. Wilson's book on *The Ants* runs to over eight-hundred pages, and ants and similar biological examples have inspired new problem-solving strategies based on 'swarm intelligence'. But the limits to what a real-life ant colony can achieve are more apparent than those of more obviously conscious beings. As the constructors move in and tarmac over our richly structured ant colony, the ants have no hope of expanding their expertise to deal with such eventualities. For us algorithmic content gives rise to new emergent forms, which themselves become victim to our algorithmic appetites, and even the inevitable limits on this inductive process we hope to decode. Maybe it is going *too* far to think of **CiE** as the conscious and interventionist observers of the ant-like activities of our more ad hoc computational colleagues! But any sceptics might remember how Turing himself put even this aspect of the computational process under the mathematical microscope. When Turing says in his 1939 paper:

> "Mathematical reasoning may be regarded ... as the exercise of a combination of ... *intuition* and *ingenuity*. ... In pre-Gödel times it was

> thought by some that all the intuitive judgements of mathematics could be replaced by a finite number of ... rules. The necessity for intuition would then be entirely eliminated. In our discussions, however, we have gone to the opposite extreme and eliminated not intuition but ingenuity, ..."

he is talking about what happens when one persistently transcends a particular context by iterating an overview, such as that of Gödel for first-order Peano arithmetic. We can trace back to this paper the genesis of powerful proof-theoretic methods which have both benefitted from, and fed back into, computability theoretic perspectives. One tends to think of computability as being a language-independent notion, but the need to describe what is going on computationally reasserts the human dimension, and leads us to appropriate proof theoretic hierarchies. This direction is represented here by *Proofs and Computation* Special Sessions contributors Ulrich Berger, Coquand, and Wainer (with Geoff Ostrin), and by proof mining expert Ulrich Kohlenbach.

An important recent development has been the growth of proof complexity since the appearance of Sam Buss' thesis on *Bounded Arithmetic* back in 1985, showing that basic complexity classes could be allied with levels of relatively easily described proof theoretic heirarchies. And if ever there was an area in need of new paradigms, it is computational complexity. There are, of course, deep and intactable problems, basic to how we compute in the future, for which no one seems to be able to even get close to a solution. The appearance of Yuri Matiyasevich and Yiannis Moschovakis (with the intriguing title *Recursion and complexity*) on our list of authors reminds us of similarly basic computational issues arising from traditional logical frameworks. Contributions from Barra and Kristiansen, Ricardo Gavaldá (on computational learning theory), Gibson and Woods, Kristiansen again, Jack Lutz (on *The Dimension of a Point: Computability Meets Fractal Geometry*), Peter Bro Miltersen, Victor Mitrana et al, Pheidas and Vidaux, and Jacobo Torán give just an indication of the great variety of output to be encountered.

Also to be found here are articles and abstracts dealing with *real computation* — another key topic at **CiE 2005** — an implicit acknowledgement of gap between how the working scientist describes the universe in terms of real numbers, and the way in which present-day computers are constrained to work with discrete data. Richard Feynman may have commented, characteristically provocative as ever, in a 1982 article on *Simulating physics with computers* in the *International Journal of Theoretical Physics*:

> "It is really true, somehow, that the physical world is representable in a discretized way, and ... we are going to have to change the laws of physics."

But the practical realities which faced Feynman the scientist in dealing with continuous mathematical models of physical systems have not changed. This area also sees a variety of theoretical approaches to the practical problems involved, some easily located within the familiar framework of what has been known as

recursive analysis, and others much more immediately geared to applications. At the more applied end of the spectrum we have Edalat-Khanban-Lieutier, Amin Farjudian, Lieutier, Pattinson, and Ning Zhong. More mathematical approaches, including work on computable and c.e. reals, etc., show up in nice contributions from George Barmpalias, Klaus Meer, Guohua Wu, Zheng Xizhong and Martin Ziegler. Korovina and Kudinov take us in the direction of computability over higher type continuous data, connecting up with more general and set theoretical work, such as that of Peter Koepke, and Alexey Stukachev.

Amongst other important computational notions not yet mentioned are computable models (Hirschfeldt, Morozov), randomness (Reimann), reverse mathematics (Joseph Berger), and other riches too many to itemise in detail.

In the end, the overall impression is one of *normal science at its best* — in its particularities inventive, relevant and soundly based, but a confluence of perspectives exceeding the sum of its parts. This is the essence of most paradigm shifts in history. In retrospect we may recognise a particular 'eureka' moment, but closer inspection often reveals revolutionary new ideas *emerging* out of a number of contributory and seemingly unrelated developments. Only when the picture is focused and comprehensive enough one can one clearly distinguish both its failings and potentialities. As Kuhn says (p.92):

> "... scientific revolutions are inaugurated by a growing sense, ... often restricted to a narrow subdivision of the scientific community, that an existing paradigm has ceased to function adequately in the exploration of an aspect of nature to which that paradigm itself had previously led the way."

Paradigm shifts are not easily come by. Their underlying ideas must be connected, justified, validated, formed in to a persuasive whole, through the detailed and selfless work of many people. Some of this work may be anticipatory, brave, but wrong, and in putting together this volume the editors have been all too aware of this. In particular cases, we have preferred to err on the side of caution. Again, this is a usual feature of paradigm shifts, and we hope our readers (and contributors) will understand this. We do believe **CiE** to provide a home for those exploring the developing real-world relevance of computability and complexity, and we hope this volume is a first sign of what we can achieve as a more coherent scientific community.

It is appropriate to give Thomas Kuhn the (almost) final word (pp.167–168):

> "The very existence of science depends upon investing the power to choose between paradigms in the members of a special kind of community. Just how special that community must be if science is to survive and grow may be indicated by the very tenuousness of humanity's hold on the scientific enterprise. Every civilization of which we have records has possessed a technology, an art, a religion, a political system, laws, and so on. In many cases those facets of civilization have been as developed as our own. But only the civilizations that descend from Hellenic Greece have possessed more than the most rudimentary science. The bulk of

> scientific knowledge is a product of Europe in the last four centuries. No other place and time has supported the very special communities from which scientific productivity comes."

Even if Europe is now but one part of an interconnecting global scientific community, computability continues to be an area in which the European contribution is something quite special.

Computably Enumerable Sets in the Solovay and the Strong Weak Truth Table Degrees

George Barmpalias

School of Mathematics,
University of Leeds,
Leeds LS2 9JT, U.K
georgeb@maths.leeds.ac.uk
http://www.maths.leeds.ac.uk/~georgeb

Abstract. The strong weak truth table reducibility was suggested by Downey, Hirschfeldt, and LaForte as a measure of relative randomness, alternative to the Solovay reducibility. It also occurs naturally in proofs in classical computability theory as well as in the recent work of Soare, Nabutovsky and Weinberger on applications of computability to differential geometry. Yu and Ding showed that the relevant degree structure restricted to the c.e. reals has no greatest element, and asked for maximal elements. We answer this question for the case of c.e. sets. Using a doubly non-uniform argument we show that there are no maximal elements in the sw degrees of the c.e. sets. We note that the same holds for the Solovay degrees of c.e. sets.

1 Introduction

The strong weak truth table reducibility was suggested by Downey, Hirsch feldt, and LaForte as a measure of relative randomness. Versions of this reducibility are present in computability theory; for instance, these are automatically produced by the basic technique of 'simple permitting' and one of them was used in the recent work of Soare, Nabutovsky and Weinberger on applications of computability theory to differential geometry. The strong weak truth table reducibility naturally induces a degree structure, the sw degrees. Yu and Ding showed that the sw degrees restricted to the c.e. reals have no greatest element, and asked for maximal elements. We solve this question for the case of c.e. sets. Using a doubly non-uniform argument we show that there are no maximal elements in the sw degrees of the c.e. sets. The strong weak truth table reducibility was originally suggested as an alternative for the Solovay (or *domination*) reducibility which has been very successful tool for the study the complexity of c.e. reals but presents various shortcomings outside this class. Of course, the sw degrees present other difficulties (as the lack of join operator, see below) but they are nevertheless very interesting to study from a wider perspective. Moreover, Downey, Hirschfeldt and LaForte [2] noticed that as far as the computably enumerable sets are concerned, the sw degrees *coincide* with the Solovay degrees.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 8–17, 2005.

So we also show that the Solovay degrees of c.e. sets have no maximal element. In the following we assume basic computability theory background; knowledge of algorithmic randomness is not essential but can be useful. For definitions, motivation and history of related notions as the Solovay degrees we refer mainly to [1] and secondly to [4].

Studying relative randomness, Downey, Hirschfeldt and LaForte [2] found Solovay reducibility insufficient, especially as far as non-c.e. reals are concerned. One of the two new measures for relative randomness they suggested is a strengthening of the weak truth table reducibility, which they called *strong weak truth table reducibility* or sw for short. This reducibility is quite natural since it occurs in many proofs in classical computability theory: it follows when we apply simple permitting for the construction of a set 'below' a given one.

Definition 1. *(Downey, Hirschfeldt and LaForte [2]) We say $A \leq_{sw} B$ if there is a Turing functional Γ and a constant c such that $\Gamma^B = A$ and the use of this computation on any argument n is bounded by $n + c$.*

The special case when $c = 0$ gives a stronger reducibility which was used by Soare, Nabutovsky and Weinberger (see [7]) in applying computability theory to differential geometry.

We remind the definition of a c.e. real.

Definition 2. *A real number is* computably enumerable (c.e.) *if it is the limit of a computable increasing sequence of rationals.*

The main justification for $\leq_{\mathrm{sw}}$ as a measure of relative randomness was the following

Proposition 3. *(Downey, Hirschfeldt, LaForte [2]) If $a \leq_{sw} b$ are c.e. reals then for all n, the prefix-free complexity of $a \upharpoonright n$ is less than or equal to that of $b \upharpoonright n$ (plus a constant).*

So $\leq_{\mathrm{sw}}$ arguably qualifies as a *measure of relative randomness* for the c.e. reals (and in particular, it preserves randomness). Downey, Hirschfeldt, LaForte [2] have showed that Solovay reducibility (also known as *domination*) and strong weak truth table reducibility *coincide* on the c.e. sets. But, as we see below, this is not true for the c.e. reals.

Yu and Ding proved the following

Theorem 4. *(Yu and Ding [6]) There is no sw-complete c.e. real.*

By a 'uniformization' of their proof they got two c.e. reals which have no c.e. real sw-above them. Hence

Corollary 5. *(Downey, Hirschfeldt, LaForte [2]) The structure of sw-degrees is not an upper semi-lattice.*

They also asked whether there are maximal sw-degrees of c.e. reals. They conjectured that there are such, and they are exactly the ones that contain random c.e. reals. The main idea of their proof of theorem 4 can be applied for the case of c.e. sets in order to get an analogous result. Using different ideas we prove the following stronger result.

Theorem 6. *There are no sw-maximal c.e. sets. That is, for every c.e. set A, there exists a c.e. set W such that $A <_{sw} W$.*

Since the Solovay degrees and sw-degrees coincide on the c.e. sets we get

Corollary 7. *The substructure of the Solovay degrees consisting of the ones with c.e. members (i.e. containing c.e. sets) has no maximal elements.*

2 About the Structure

We state some easy results about the c.e. sets and reals in the structure of sw degrees. We remind the following definition:

Definition 8. *A (total) Solovay test is a c.e. set of binary strings S such that $\sum_{\sigma\in S} 2^{-|\sigma|} < \infty$ (and computable). A real a avoids S if*

$$\exists^{<\infty}\sigma \in S(\sigma \sqsubset a).$$

(Schnorr) Random is a real which avoids all (total) Solovay tests.

After the discussion in the previous section, it is natural to ask: are there c.e. reals above all c.e. sets?

Proposition 9. *Every random c.e. real is sw-above every set in the finite levels of the difference hierarchy.*

But are there non-random c.e. reals with this property?

Proposition 10. *There are non-random c.e. reals sw-above every set in the finite levels of the difference hierarchy.*

E.g. $a = \sum_{e\in\mathbb{N}} \sum_{n\in W_e} 2^{-(e+n+2)}$ is non-random and sw-above all c.e. sets.

Question 1. Are the c.e. reals above the c.e. sets necessarily Schnorr random?

3 About the Proof of Theorem 6

Before we get into the technical part of the proof, we draw a map of it. Given a c.e. set A we construct three c.e. sets W_1, W_2, W_3 *one of which* will be strictly sw-above A. Figure 1 illustrates this idea and shows the double non-uniform nature of the proof, which will become more clear in the technical part (in particular, W_1 will be *qualitatively different* from W_2, W_3). We note that some of W_i may not be able to even compute A. Given a c.e. set A we construct a c.e. W which satisfy the requirements

$$\mathcal{Q} : A \leq_{\text{sw}} W$$
$$\mathcal{N}_e : \Phi_e^A \neq W$$

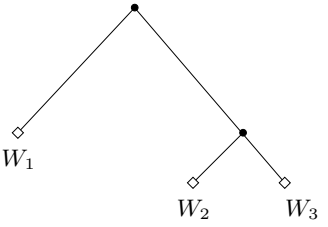


Fig. 1. Double non-uniformity in the proof of theorem 6

where Φ_e is the e-th sw-functional in an effective enumeration and has use (bounded by) $n + e$ on argument n. Although $\mathcal{N}_e$ can be satisfied with a non-uniform proof in the style of [6] (thus showing that there is no sw-maximum c.e. set), adding $\mathcal{Q}$ makes the situation more difficult. In particular, for any number of W-witnesses that we reserve for $\mathcal{N}_e$, we need to reserve roughly the same number of W-witnesses for A-coding (i.e. $\mathcal{Q}$), and these must be roughly of the same level of magnitude as those for $\mathcal{N}_e$. But this is impossible since once you have occupied an interval of $\mathbb{N}$ for $\mathcal{N}_e$ (as in [6]) you can't always find another equally big (disjoint) interval of numbers not much larger than the ones in the first interval. Note that building a large (finite) number of (candidates for) W doesn't help much since each of these W will have the need of *double space* discussed above.

To illustrate the above, consider an attempt to diagonalize against $\Phi_0^A = W$ with a witness w. W.l.o.g. assume that the constant associated with our reduction $A \leq_{\mathrm{sw}} W$ is 0 (i.e. the use is the identity function). We wait until $\Phi_0^A(w) \downarrow = 0$ and put $w \searrow W$. Then $A \upharpoonright w$ may change (in order to rectify the computation) and this change must be coded in W *below* w. So one diagonalization requires two witnesses.

To deal with this situation we will require A to be 'sufficiently charged' in the sense that the 1s in its characteristic sequence are sufficiently dense. Relying on this assumption, we won't enumerate an axiom for the reduction $A \leq_{\mathrm{sw}} W$ unless we witness a certain amount of enumeration in A. This way we save positions in W which would have to be used for the A-coding, had we not waited for this enumeration to occur. If our hypothesis is true, the construction will build one W which fulfils the requirements.

Of course, this 'density' hypothesis (which is the base of *case A* of the proof) is not without loss of generality. A second construction (*case B*) will assume the failure of the density hypothesis, and produce two sets W_1, W_2; if indeed the hypothesis fails, one of these will satisfy all the requirements. Overall we construct three different sets and so this proof is non-uniform.

Case A. The hypothesis is

$$\forall e, c \exists \ell (|\overline{A} \upharpoonright (\ell + e)| + |\overline{A} \cap (c, \ell]| < \ell - c) \tag{1}$$

and the instance of it used by $\mathcal{N}_e$ is

$$\exists \ell(|\overline{A} \upharpoonright (\ell + e)| + |\overline{A} \cap (c, \ell]| < \ell - c) \tag{2}$$

where c is the largest number reserved for diagonalization and A-coding by the higher priority requirements $\mathcal{N}_i, i < e$. Every requirement reserves a full interval of $\mathbb{N}$ for enumeration into W and $\mathcal{N}_e$ in particular reserves $(c, \ell]$, where ℓ is the least witness for (2). So, in this case $(-1, c]$ is full of numbers reserved by $\mathcal{N}_i, i < e$. Every time a requirement reserves an interval $(c, \ell]$, it automatically starts sw-coding $A \cap (c, \ell]$ into $W \cap (c, \ell]$ (the use function of the reduction being the identity). This means that from now on every time that a new element n appears in $A \cap (c, \ell]$, we enumerate a number $t \leq n$ into $W \cap (c, \ell]$. In this setting, condition 2 guarantees that although we will need to spend part of $(c, \ell]$ for A-coding, there will still be enough witnesses for a successful ripple of $\mathcal{N}_e$-diagonalizations (i.e. one that finishes with a diagonalization which is not rectified).

Since the largest number reserved by $\mathcal{N}_e$ is the ℓ of 2, if c_e, ℓ_e are the c, ℓ of condition 2 for $\mathcal{N}_e$ then $c_e = \ell_{e-1}$ where $\ell_{-1} = -1$. It is easy to see that a list

$$\ell_0 < \ell_1 < \ell_2 < \dots$$

of suitable endpoints for all the requirements can be effectively obtained by choosing an ℓ_e to be one of the ℓ of 2 for $c = \ell_{e-1}$ (say the first that occurs during the given enumeration of A). So we divide $\mathbb{N}$ into

$$(-1, \ell_0], (\ell_0, \ell_1], \dots,$$

sw-code $A \cap (\ell_{e-1}, \ell_e]$ into $W \cap (\ell_{e-1} \ell_e]$ for each e, and use the rest of the witnesses for a diagonalization ripple for $\mathcal{N}_e$ (i.e. a sequence of diagonalizations where each of them is performed after the previous one has been rectified). It is straightforward to use an initial segment of $(\ell_{e-1}, \ell_e]$ for the A coding and the rest of it for diagonalizations. So we injectively map (the current value of) $\overline{A} \cap (\ell_{e-1}, \ell_e]$ *onto* an initial segment of $(\ell_{e-1}, \ell_e]$ in an *order-preserving* way and for the sake of $\mathcal{Q}$ require that whenever an element of that $\overline{A} \cap (\ell_{e-1}, \ell_e]$ appears in A, the corresponding element (which belongs to the same interval) is enumerated into W. It is obvious that each element of $(\ell_{e-1}, \ell_e]$ is mapped to a number less than or equal to itself and so the coding is sw with the identity as use function.

$\mathcal{N}_e$-Module.

1. *(Set up)* Wait until ℓ_{e-1} has been defined and there is an $\ell > \ell_{e-1}$ as in 2 with $c = \ell_{e-1}$. Define $\ell_e = \ell$ and the attack interval

   $$I_e = (\ell_{e-1}, \ell_e].$$

   Injectively map $I_e \cap \overline{A}$ onto an initial segment of I_e in an order-preserving way (this can be done in a unique way).
2. *(Diagonalization)* Wait until $\ell(\Phi_e^A, W) > \ell_e$ and put $\max(\overline{W} \cap I_e) \searrow W$.

Each of these strategies require attention when they are ready to move on to the next step (note that part 2 is a loop). Step 1 is performed only once for each requirement and so there will be no injury.

$\mathcal{Q}$-Module. Let Γ be the functional we build for the reduction $A \leq_{\mathrm{sw}} W$.

1. *(Γ rectification)* Search Γ up to the finite current level of its definition and find the least n with
$$\Gamma^W(n) \downarrow \neq A(n).$$
Then $n \in (\ell_i, \ell_{i+1}]$ for some i; enumerate into W the corresponding element of n under the injective mapping defined during the definition of ℓ_{i+1}.
2. *(Γ enumeration)* Let i be the largest number such that $\ell_i \downarrow$. If $\Gamma^W(\ell_i) \uparrow$ then enumerate axioms $\Gamma^W = A$ up to ℓ_i with use function the identity.

Construction. At each stage:

- Run $\mathcal{Q}$-module
- Run $\mathcal{N}_e$-module for the highest $\mathcal{N}$ requiring attention.

Verification. By induction we show that for every e, $\mathcal{N}_e$ is satisfied and Γ is defined and correct up to ℓ_e. Since by 1 ℓ_e is eventually defined for all e and $\ell_e < \ell_{e+1}$, this is all we need to show. Supposing that it holds for all $i < e$, we show that it is true for e. Suppose that ℓ_e is defined at stage s_0. At this stage $W \cap (\ell_{e-1}, \ell_e]$ is empty and according to the $\mathcal{Q}$-module the only numbers in $(\ell_{e-1}, \ell_e]$ enumerated into W by this strategy will be because of numbers appearing in $A \cap (\ell_{e-1}, \ell_e]$ *after* stage s_0.

By the first step of the strategy $\mathcal{N}_e$, at s_0 we have

$$|\overline{A} \upharpoonright (\ell_e + e)| + |\overline{A} \cap (\ell_{e-1}, \ell_e]| < |(\ell_{e-1}, \ell_e]|. \tag{3}$$

Strategy $\mathcal{Q}$ can enumerate into $W \cap (\ell_{e-1}, \ell_e]$ no more than the first $|\overline{A}[s_0] \cap (\ell_{e-1}, \ell_e]|$ elements of $(\ell_{e-1}, \ell_e]$. Also, no other strategy apart from $\mathcal{N}_e$ can enumerate numbers of this interval into W. So according to 3 there will be more than $|\overline{A}[s_0] \upharpoonright (\ell_e + e)|$ for the use of $\mathcal{N}_e$. Each time the agreement $\Phi_e^A = W$ exceeds ℓ_e, this strategy will perform a diagonalization. After each diagonalization, the length of agreement can only exceed ℓ_e again if a number enters A below $\ell_e + e$. Hence there will be a diagonalization which cannot be rectified and this shows that $\mathcal{N}_e$ succeeds.

On the other hand, since $\mathcal{N}_e$ chooses as diagonalization witness the largest element of $(\ell_{e-1}, \ell_e]$ not yet in W, it follows from 3 that the first $|\overline{A}[s_0] \cap (\ell_{e-1}, \ell_e]|$ elements will not be used by this strategy (since, by the time it would need to use them it will have reached a diagonalization which cannot be rectified). And of course they are not going to be used by other $\mathcal{N}$ strategies nor by $\mathcal{Q}$ for the sake of numbers appearing in A outside $(\ell_{e-1}, \ell_e]$. So any of these will stay outside W until (if ever) its corresponding element (under the injective mapping defined in step 1 of $\mathcal{N}_e$, which is greater or equal to it) enters A. So $\mathcal{Q}$ will always be able to rectify (and refresh) Γ on $(\ell_{e-1}, \ell_e]$. So, using the induction hypothesis, eventually Γ will be defined and correct up to ℓ_e. This concludes the induction step and the verification. Note that the reduction of A to W just described is also a many-one reduction.

Case B**.** Suppose that 1 does not hold and so

$$\exists e, c \forall \ell (|\overline{A} \upharpoonright (\ell + e)| + |\overline{A} \cap (c, \ell]| > \ell - c)$$

which can be written as

$$\exists e, c \forall \ell > e, (|\overline{A} \upharpoonright \ell| + |\overline{A} \cap (c, \ell - e]| > \ell - e - c)$$

which implies

$$\exists c \forall \ell > c, (2|\overline{A} \upharpoonright \ell| > \ell - c).$$

But the latter can be written as

$$\exists c \forall \ell > c, (2|A \upharpoonright \ell| < \ell + c).$$

If there is some $0 \leq c_1 < c$ such that

$$\exists^\infty \ell > c, (2|A \upharpoonright \ell| \geq \ell + c_1)$$

there will be a greatest such. For that one it would be

$$\exists^\infty \ell > c, (2|A \upharpoonright \ell| = \ell + c_1)$$

for a possibly different constant c. But then A would be computable which is a trivial case. So we may assume that there is no such c_1 and hence

$$\exists c \forall \ell > c, (2|A \upharpoonright \ell| < \ell).$$

By finitely modifying A (e.g. set it empty up to c) we get

$$\forall \ell > 0, (2|A \upharpoonright \ell| < \ell). \tag{4}$$

This extra hypothesis does not restrict the result, since if it holds for a set then it holds for any finite modification of it. Now 1 allows us to sw-code A into W by only using the even numbers. So we reserve $2\mathbb{N}$ for $\mathcal{Q}$ and define the coding as follows.

$\mathcal{Q}$-Module. If some n has just been enumerated into A, put the largest even number $\leq n$ of $\overline{W}$ into W.

By this procedure for any n, $A \upharpoonright n$ is coded in the even positions of $W \upharpoonright n$. We may fix an enumeration of A in which at most one number appears in it at each stage. To see that the coding succeeds we prove the following

Lemma 11. *If $W(2k) = 0$ at some stage, then*

$$|A \upharpoonright 2k| \geq |\{2t < 2k \mid W(2t) = 1\}|$$

at the same stage.

Proof of lemma. Indeed, each even number in $W \upharpoonright 2k$ must have been the code of some number n enumerated in A in a previous stage. But no such n (i.e. one that triggered enumeration into $W \upharpoonright 2k$) can be $\geq 2k$ since $\mathcal{Q}$ would have chosen $2k$ or greater (since $W(2k) = 0$ such codes were available). □

Now the coding works unless there is a stage s where some $n \searrow A$ and all even numbers $\leq n$ are already in W. If $2k$ is the least even not yet in W, $2k > n \geq 0$ and lemma 11 implies that

$$|A \upharpoonright 2k| \geq k$$

holds at this stage, which contradicts 4. So $\mathcal{Q}$ indeed makes sure that $A \leq_{\mathrm{sw}} W$. Of course this is independent with what we do with the odd numbers in relation to W, which we are going to use for satisfying the $\mathcal{N}$ requirements.

We will construct two sets W_1, W_2 both sw-above A (via the $\mathcal{Q}$-strategy) one of which will satisfy all $\mathcal{N}$ requirements. So we can replace $\mathcal{N}$ by

$$\mathcal{N}'_e : \Phi_e^A \neq W_1 \ \vee \ \Psi_e^A \neq W_2.$$

Each $\mathcal{N}'_e$ will occupy the odd numbers of an interval $[2c_e + 1, 2c_{e+1} + 1)$ of $\mathbb{N}$ and use them as diagonalization witnesses. So the e-th requirement will have

$$c_{e+1} - c_e := k \tag{5}$$

numbers available for each of W_1, W_2, from $2c_e + 1$ on. To find a k sufficiently big to guarantee the success of this diagonalization ripple we consider the rectification resources of A below

$$[2(c_{e+1} - 1) + 1 + e] + 1 = 2(c_e + k) + e$$

which bounds the use of any of our e-witnesses. By 4,

$$|A \upharpoonright 2(c_e + k) + e| < c_e + k + \frac{e}{2}$$

and so there can only be less than $c_e + k + \frac{e}{2}$ rectifications to the e-diagonalizations. Since we play with two sets W_1, W_2 (and $\mathcal{N}'_e$ is a disjunction) we have $2k$ witnesses available. Hence it suffices to choose k so that

$$2k \geq c_e + k + \frac{e}{2}$$

i.e. $k \geq c_e + \frac{e}{2}$. By setting $k = c_e + \frac{e}{2}$, $c_0 = 0$ and using 5 we get an appropriate sequence (c_i) (where c_e is the number of witnesses reserved by $\mathcal{N}'_i$, $i < e$) and are able to proceed with the $\mathcal{N}'_e$-strategy.

$\mathcal{N}'_e$-Module.

1. Wait until $\ell(\Phi_e^A, W_i) > 2c_{e+1} + 1$ for both $i = 1, 2$.
2. Consider the maximum witness of $\mathcal{N}'_e$ not yet in W_1 or W_2; that is,

$$\max(2\mathbb{N} + 1) \cap [2c_e + 1, 2c_{e+1} + 1) \cap (\overline{W}_1 \cup \overline{W}_2).$$

Put it into W_1 if it is not in already; otherwise enumerated into W_2.

The above strategy requires attention when the condition in the first step is fulfilled. Now the *construction* is straightforward: at stage s run the $\mathcal{Q}$-module *for both* W_i, and the highest priority $\mathcal{N}'_e$-module requiring attention.

Verification. First note that $\mathcal{Q}$ uses only even numbers and each $\mathcal{N}'_e$ only odd ones. So $\mathcal{Q}$ does not have any interaction with the rest of the construction and so $A \leq_{\mathrm{sw}} W_i$ can be derived as explained above; note that the characteristic sequences of W_1, W_2 are identical on the even positions. Moreover there is no interaction between pairs of $\mathcal{N}'$ strategies since their witness intervals are disjoint. Each of these requirements succeeds because of the choice of witness intervals, as explained above. The operation of such a requirement is a sequence of diagonalizations against *one* of the reductions $\Phi_e^A = W_1$, $\Psi_e^A = W_2$, each of which (except the first one) takes place after an A-change below a certain level. When we defined the parameters c_e we showed that these A-changes cannot be as many as the number of witnesses for both W_i. This means that one of these reductions will stop having expansionary stages and thus the $\mathcal{N}'$ strategy will start waiting in step 1 indefinitely, after a certain stage. This is obviously a successful outcome.

Further Remarks. One of the referees has pointed out that theorem 6 can be proved as follows (this approach is closer to the construction of W_1 above). Fix the density of A

$$\alpha = \limsup_{n\to\infty} \frac{|\{m \leq n : A(m) = 1\}|}{n}$$

and a rational approximation d of it with error less than ϵ (say $= \frac{1}{10}$). Then we can effectively choose diagonalization intervals I_k as we did for the construction of W_1, so that after we enumerate axioms on I_k for the computation of A from W, only a small number (relative to the length of I_k) of extra elements can enter A below $\max I_k$ (e.g. $\frac{1}{10}$th of $\max I_k$). Then, choosing $\max I_k$ big enough we can ensure that there is space in I_k for A-coding and enough A-diagonalizations, even for the case when we diagonalize against a functional with use $x + c$ for arbitrary constant c.

Note that roughly speaking the smaller the error ϵ of the approximation to α is, the more non-uniform the proof is. E.g. if we choose $\epsilon \leq \frac{1}{n}$ we can divide the unit interval into n equal parts and consider the centers of these as our rational approximations d. For any given A one of these must be correct and so the corresponding construction is successful. Setting $\epsilon \leq \frac{1}{3}$ suffices: we get three sets W_1, W_2, W_3 one of which satisfies the requirements. It is interesting that there is no obvious way to succeed with less than three attempts.

References

1. R. Downey; D. Hirschfeldt; G. LaForte, Randomness and reducibility. *Journal of Computer and System Sciences* **68** (2004) 96–114
2. R. Downey; D. Hirschfeldt; G. LaForte, Randomness and reducibility. Mathematical foundations of computer science, 2001, 316–327, *Lecture Notes in Comput. Sci.*, 2136, Springer, Berlin, 2001.
3. R. Downey, D. Hirschfeldt and A. Nies, Randomness, computability, and density *SIAM Journal of Computation*, Vol. **31** No. 4, pp. 1169–1183 (2002)

4. R. Downey, Some Recent Progress in Algorithmic Randomness. Preprint, 2004.
5. P. Odifreddi, 'Classical recursion theory' Amsterdam Oxford: North-Holand, 1989
6. L. Yu and D. Ding, There is no *SW*-complete c.e. real. to appear in *J. Symb. Logic*.
7. R. Soare, Computability Theory and Differential Geometry, to appear in the *Bulletin of Symbolic Logic*.
8. R. Soare, *Recursively enumerable sets and degrees*, Berlin London: Springer-Verlag, 1987.

The Fan Theorem and Uniform Continuity

Josef Berger*

Mathematisches Institut,
Universität München, Theresienstraße 39,
D-80333 München, Germany
Josef.Berger@mathematik.uni-muenchen.de

Abstract. In presence of continuous choice the fan theorem is equivalent to each pointwise continuous function f from the Cantor space to the natural numbers being uniformly continuous. We investigate whether we can prove this equivalence without the use of continuous choice. By strengthening the assumption of pointwise continuity of f to the assertion that f has a modulus of pointwise continuity which itself is pointwise continuous, we obtain the desired equivalence.

We work entirely in the system **BISH** of Bishop's constructive mathematics [2], which means we are using intuitionistic logic and an appropriate set-theoretic foundation like Aczel's CZF, see [1]. We are interested in the constructive content of the following assertion:

UC Each pointwise continuous function $f : 2^{\mathbb{N}} \to \mathbb{N}$ is uniformly continuous.

For a binary sequence[1] $\alpha = (\alpha_0, \alpha_1, \alpha_2, \dots)$ we write $\overline{\alpha}n$ for the initial segment $(\alpha_0, \dots, \alpha_{n-1})$. The **Cantor space** $2^{\mathbb{N}}$ is the set of binary sequences equipped with the compact metric

$$d(\alpha, \beta) = \inf \left\{ 2^{-n} \mid \overline{\alpha}n = \overline{\beta}n \right\}.$$

By Theorem 1.2 in Chapter 5 of [4], a function $f : 2^{\mathbb{N}} \to \mathbb{N}$ is pointwise continuous if and only if

$$\forall \alpha \exists N \forall \beta \left(\overline{\alpha}N = \overline{\beta}N \Rightarrow f(\alpha) = f(\beta) \right).$$

In this case N is called a **witness for f being continuous at α**. Another version of this theorem is that a function $f : 2^{\mathbb{N}} \to \mathbb{N}$ is uniformly continuous if and only if

$$\exists N \forall \alpha, \beta \left(\overline{\alpha}N = \overline{\beta}N \Rightarrow f(\alpha) = f(\beta) \right).$$

* The author thanks Hajime Ishihara, Robert Lubarsky, Peter Schuster, and Wim Veldman for fruitful discussions about uniform continuity as well as he thanks the anonymous referees for helpful comments. Furthermore, he thanks the Graduiertenkolleg *Logik in der Informatik* for support.

[1] We use Greek letters α, β, γ for infinite binary sequences. For finite binary sequences we use the letters u, v, w. We use the letters n and N for natural numbers.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 18–22, 2005.

We now formulate two major principles of Brouwer's intuitionism, namely the **fan theorem** for detachable bars and the principle of **continuous choice**. The fan theorem reads as follows:

FAN Each detachable bar is uniform.

A set B of finite binary sequences is **detachable** if

$$\forall u\,(u \in B \vee u \notin B)\,.$$

A detachable set B is a **bar** if

$$\forall \alpha \exists n\,(\overline{\alpha}n \in B)\,.$$

A bar is **uniform** if

$$\exists N \forall \alpha \exists n \leq N\,(\overline{\alpha}n \in B)\,.$$

Note that by the compactness of the Cantor space, **FAN** holds classically. Note further that **FAN** does not hold recursively[2]; see Corollary 4.7.6 in [8].

The **principle of continuous choice** consists of two parts:[3]

$\mathbf{CC}_1$ Each function from $2^{\mathbb{N}}$ to $\mathbb{N}$ is pointwise continuous.

$\mathbf{CC}_2$ If $P \subset 2^{\mathbb{N}} \times \mathbb{N}$, and for each α there exists n such that $(\alpha, n) \in P$, then there is a function $f : 2^{\mathbb{N}} \to \mathbb{N}$ such that $(\alpha, f(\alpha)) \in P$ for all α.

Thus continuous choice divides into a continuity part $\mathbf{CC}_1$ and a choice part $\mathbf{CC}_2$. Using *tertium non datur*, we can define a discontinuous function; thus continuous choice fails classically. It also fails recursively; see Theorem 2.2 in Chapter 5 of [4] or Proposition 4.6.7 in [8]. Bridges and Richman showed that under continuous choice, **FAN** and **UC** are equivalent; see Section 3 of Chapter 5 in [4]. Assuming continuous choice implicitly in their concept of function, Iris Loeb [3] and Wim Veldman [9] obtain the equivalence of **FAN** and **UC** as well.

Proposition 1. *Under continuous choice, **UC** and **FAN** are equivalent.*

We want to investigate how far we can get without continuous choice. First we mention that the proof of $\mathbf{UC} \Rightarrow \mathbf{FAN}$ does not require this additional choice assumption; see Theorem 3.3 in Chapter 5 of [4].

Lemma 2. ***UC** implies **FAN**.*

Proof. Let B be a detachable bar. We define

$$f : 2^{\mathbb{N}} \to \mathbb{N},\ \alpha \mapsto \min\{n \mid \overline{\alpha}n \in B\}\,.$$

[2] That means, when you add the Church-Markov-Turing thesis to **BISH**.

[3] This principle is often formulated in terms of sequences of natural numbers, instead of binary sequences.

We stress that this definition only makes sense because B is detachable. Note that f is pointwise continuous, since

$$\forall \alpha, \beta \left(\overline{\alpha} f(\alpha) = \overline{\beta} f(\alpha) \Rightarrow f(\alpha) = f(\beta) \right). \tag{1}$$

Thus by **UC**, f is even uniformly continuous. By Proposition 4.2 in Chapter 4 of [2], f is bounded. Let N be a bound for the range of f. This implies that

$$\forall \alpha \exists n \leq N \left(f(\alpha) = n \right),$$

which in turn implies that

$$\forall \alpha \exists n \leq N \left(\overline{\alpha} n \in B \right).$$

Thus B is a uniform bar.

The next question is whether **FAN** implies **UC**. A first step in this direction is to show that a stronger form of **FAN** implies **UC**.

F-FAN Each bar is uniform.

The **F** in **F-FAN** stands for *full* and indicates that **F-FAN** is the full form of the fan theorem. From the very formulation it is clear that **F-FAN** implies **FAN**. Since classically every set is detachable, the statements **FAN** and **F-FAN** are equivalent, hence true. For a deeper discussion about this implication, see [7]. It is well-known that **F-FAN** implies **UC**; see for example Theorem 3.6 in Chapter 6 of [8].

Lemma 3. ***F-FAN*** *implies* ***UC****.*

Proof. Let $f : 2^{\mathbb{N}} \to \mathbb{N}$ be pointwise continuous. Set

$$B = \{ \overline{\alpha} n \mid \alpha \in 2^{\mathbb{N}} \text{ and } n \text{ is a witness for } f \text{ being continuous at } \alpha \}.$$

This B is a bar because f is pointwise continuous. By **F-FAN**, B is uniform; that is, there is N such for each α there is a witness n for f being continuous at α with $n \leq N$. But this amounts to the uniform continuity of f.

Thus **UC** lies somewhere in the no-man's-land between **FAN** and **F-FAN**. What we are doing now is to strengthen the property of f being continuous in the formulation of **UC**. This gives rise to a weaker condition **MUC** which still implies **FAN**, but now is derivable from **FAN**. This is in the spirit of constructive reverse mathematics as practised by Hajime Ishihara [5], [6]. See also the recent papers of Iris Loeb [3] and Wim Veldman [9].

For this purpose we define:

MUC Each function $f : 2^{\mathbb{N}} \to \mathbb{N}$ which has a modulus of continuity which itself is pointwise continuous is uniformly continuous.

If $f, g : 2^{\mathbb{N}} \to \mathbb{N}$ are functions, then g is a **modulus of pointwise continuity of** f if the following holds:

$$\forall \alpha, \beta \left(\overline{\alpha} g(\alpha) = \overline{\beta} g(\alpha) \Rightarrow f(\alpha) = f(\beta) \right).$$

Classically, each pointwise continuous function has a modulus of pointwise continuity. But this step requires a strong form of choice, namely $\mathbf{CC}_2$. Thus inside **BISH** it makes a significant difference to require that the witnesses of pointwise continuity are given by a function.

We obtain the equivalence of **MUC** and **FAN** inside **BISH**, without any use of continuous choice.

Proposition 4. ***MUC*** *and* ***FAN*** *are equivalent.*

Proof. First we prove that **MUC** implies **FAN**. Considered carefully, our proof of Lemma 2 still works in this altered situation. Let B be a detachable bar. Define again

$$f : 2^{\mathbb{N}} \to \mathbb{N},\ \alpha \mapsto \min\{n \mid \overline{\alpha}n \in B\}\,.$$

By (1) we obtain that $g \equiv f$ is a modulus of continuity for f. By **MUC**, f is uniformly continuous. Now the proof that B is uniform proceeds just like the proof of Lemma 2.

Now we show that **FAN** implies **MUC**. Our proof is similar to the proof of Theorem 3.2 in Chapter 5 of [4], where the authors show that under **FAN** and continuous choice each function $f : 2^{\mathbb{N}} \to \mathbb{N}$ is uniformly continuous. Here we have weakened the assumption of continuous choice to the assumption that f has a continuous modulus of continuity. Assume that functions f, g from $2^{\mathbb{N}}$ to $\mathbb{N}$ are given such that g is a modulus of continuity for f and that g is pointwise continuous. We show that f is uniformly continuous. Set

$$B = \{u \mid g(u^*) \leq |u|\}.$$

We must explain what we mean by u^* and by $|u|$. Suppose that $u = (u_0, \dots, u_{n-1})$. Then

$$u^* = (u_0, \dots, u_{n-1}, 0, 0, 0, \dots)$$

and $|u|$ is the length of u, that means n. Clearly B is detachable. We show that B is a bar. Fix any α. Let n be a witness for g being continuous at α, and assume that $n \geq g(\alpha)$. Then

$$g(\overline{\alpha}n^*) = g(\alpha) \leq n = |\overline{\alpha}n|\,,$$

thus $\overline{\alpha}n \in B$. By **FAN**, there exists N such that

$$\forall\alpha\exists n \leq N\,(\overline{\alpha}n \in B)\,.$$

Now we can show that

$$\forall\alpha,\beta\,\big(\overline{\alpha}N = \overline{\beta}N \Rightarrow f(\alpha) = f(\beta)\big)\,,$$

which is just the uniform continuity of f. To this end fix α, β with $\overline{\alpha}N = \overline{\beta}N$. There is $n \leq N$ such that $\overline{\alpha}n \in B$. Set $\gamma \equiv \overline{\alpha}n^*$. Thus $g(\gamma) \leq n \leq N$. We obtain

$$\overline{\gamma}g(\gamma) = \overline{\alpha}g(\gamma) = \overline{\beta}g(\gamma),$$

which implies that

$$f(\alpha) = f(\gamma) = f(\beta).$$

References

1. Peter Aczel and Michael Rathjen. *Notes on Constructive Set Theory*. Report No. 40, Institut Mittag-Leffler, The Royal Swedish Academy of Sciences, 2000/2001.
2. Errett Bishop and Douglas Bridges. *Constructive Analysis*. Grundlehren der mathematischen Wissenschaften, 279. Springer, 1967.
3. Iris Loeb *Equivalents of the (Weak) Fan Theorem*. Annals of Pure and Applied Logic, Volume 132, Issue 1, February 2005, Pages 51-66
4. Douglas Bridges and Fred Richman. *Varieties of Constructive Mathematics*. Cambridge University Press, 1987.
5. Hajime Ishihara. *Constructive Reverse Mathematics: Compactness Properties*. In: L. Crosilla and P. Schuster (eds). *From Sets and Types to Topology and Analysis*. Oxford University Press. To appear.
6. Hajime Ishihara, *Informal constructive reverse mathematics*, Feasibility of theoretical arguments of mathematical analysis on computer (Japanese) (Kyoto 2003), Sūrikaisekikenkyūsho Kōkyūroku **1381** (2004), 108–117.
7. Ieke Moerdijk, *Heine-Borel Does not Imply the Fan Theorem*, J. Symb. Log. 49(2): 514-519 (1984)
8. A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics. An Introduction. Vol 1.* North-Holland Publ. Co., Amsterdam, 1988.
9. Wim Veldman. *The Fan Theorem as an Axiom*. Preprint.

Continuous Semantics for Strong Normalization

Ulrich Berger

Department of Computer Science,
University of Wales Swansea,
Singleton Park,
Swansea SA2 8PP, UK
u.berger@swan.ac.uk

Abstract. We prove a general strong normalization theorem for higher type rewrite systems based on Tait's strong computability predicates and a strictly continuous domain-theoretic semantics. The theorem applies to extensions of Gödel's system T, but also to various forms of bar recursion for which strong normalization was hitherto unknown.

1 Introduction

The problem of proving strong normalization for typed λ-calculi and higher type rewrite systems has been studied extensively in the literature [14, 9, 15, 10, 17, 7, 5, 8, 1, 16, 6, 11]. In this paper we present a new method for proving strong normalization of higher type rewrite systems based on a strict domain-theoretic semantics. The idea is similar to Plotkin's adequacy proof for PCF [12]: One gives a suitable interpretation of terms in a domain-theoretic model and uses a continuity argument to show that any term not denoting $\bot$ normalizes. The main difference between Plotkin's and our result is that while Plotkin considers terms with a general fixed point operator, for which, of course, only *weak* normalization can be proven, we consider recursion schemes defined by pattern matching and we prove *strong* normalization.

Another new aspect of our method is that it allows for a modular normalization proof: First one proves strong normalization for the underlying typed λ-calculus with *stratified constants* only, i.e. constants with conversion rules that do not involve recursion. This can be done by an extension of Tait's computability method [14]. Then one uses a continuity argument to lift strong normalization to recursively defined constants that have a total value w.r.t. to a strict domain-theoretic semantics.

We will apply our results to a λ-calculus formulation of Gödel's system T extended by two versions of bar recursion in finite types: Spector's original version [13], and a version due to Berardi, Bezem and Coquand [2]. For this system strong normalization was hitherto unknown.

In this paper we consider a simply typed system over the booleans and the integers, closed under the formation of list and function types. The motivation for this (somewhat ad hoc) choice is that this allows for a convenient formulation of

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 23–34, 2005.

bar recursion. Our results could be easily extended to type systems closed under arbitrary strictly positive definitions. Also extensions to second-order polymorphic types seem to be possible.

2 Extended Gödel's System T

The set of *types*, $\rho, \sigma, \ldots$ is generated from the base types boole and nat by the formation of *list types*, ρ^*, and *function types*, $\rho \to \sigma$. As usual we write $\boldsymbol{\rho} \to \sigma$ for $\rho_1 \to \ldots \to \rho_n \to \sigma$ where $\to$ associates to the right. Types which are not function types, i.e. boole, nat and list types ρ^*, are called *inductive* (because their elements will be generated inductively). We let the letter ι range over inductive types.

The term language is determined by a set $\mathcal{C}$ of *typed constants* c^ρ. *Typed terms* are constructed from *typed variables*, x^ρ, and constants, c^ρ, by abstraction, $(\lambda x^\rho M^\sigma)^{\rho\to\sigma}$, application, $(M^{\rho\to\sigma} N^\rho)^\sigma$, and *constructor term* formation, 0^{nat}, $\mathsf{S}(M^{\mathsf{nat}})^{\mathsf{nat}}$, $[]^{\rho^*}$, $\mathsf{cons}(M^\rho, N^{\rho^*})^{\rho^*}$. Type information will often be omitted provided this doesn't cause ambiguities. Instead of M^ρ we will sometimes write $M\colon\rho$. We let the symbol co range over the constructors 0, S, [] and cons. In a constructor term $\mathsf{co}(\boldsymbol{M})^\iota$ the terms $\boldsymbol{M}$ are called *arguments*.

β-conversion is defined as usual by

$$(\lambda x\, M)N \mapsto M[N/x]$$

where by $M[N/x]$ we mean the usual substitution of every free occurrence of x in M by N renaming bound variables in M if necessary. More general we will consider substitutions θ, which are mappings from variables to terms of the same type, and define $M\theta$ as the simultaneous replacement in M of x by $\theta(x)$ renaming bound variables in M if necessary.

The operational meaning of a constant $c \in \mathcal{C}$ of type $\rho_1 \to \ldots \to \rho_n \to \sigma$ is determined by *constant-conversion rules* of the form

$$c\, L_1^{\rho_1} \ldots L_n^{\rho_n} \;\mapsto\; R^\sigma$$

We require that for any constant c the number n above is fixed, i.e. if $c\boldsymbol{L} \mapsto R$ and $c\boldsymbol{L}' \mapsto R'$ are rules, then the vectors $\boldsymbol{L}$ and $\boldsymbol{L}'$ must have the same length. In the situation above we say that c *takes* n *arguments*.

Consider, for example, the constants $\mathsf{if}\colon \mathsf{boole} \to \rho \to \rho \to \rho$, $<\colon \mathsf{nat} \to \mathsf{nat} \to \mathsf{boole}$, $\mathsf{lh}\colon \rho^* \to \mathsf{nat}$, $\mathsf{get}\colon \rho^* \to \mathsf{nat} \to \rho$, and ${+\!\!+}\colon \rho^* \to \rho^* \to \rho^*$,

$$\begin{aligned}
\mathsf{if}\,\mathsf{T}\,x\,y &\mapsto x\\
\mathsf{if}\,\mathsf{F}\,x\,y &\mapsto y\\
n < 0 &\mapsto \mathsf{F}\\
0 < \mathsf{S}(m) &\mapsto \mathsf{T}\\
\mathsf{S}(n) < \mathsf{S}(m) &\mapsto n < m\\
\mathsf{lh}\,[] &\mapsto 0
\end{aligned}$$

$$\begin{aligned}
\mathsf{lh}\,\mathsf{cons}(x,s) &\mapsto \mathsf{S}(\mathsf{lh}(s))\\
\mathsf{get}\,[]\,n &\mapsto 0^\rho\\
\mathsf{get}\,\mathsf{cons}(x,s)\,0 &\mapsto x\\
\mathsf{get}\,\mathsf{cons}(x,s)\,\mathsf{S}(n) &\mapsto \mathsf{get}\,s\,n\\
[] +\!\!+\, t &\mapsto t\\
\mathsf{cons}(x,s) +\!\!+\, t &\mapsto \mathsf{cons}(x, s +\!\!+ t)
\end{aligned}$$

where 0^ρ is some closed term of type ρ. These are examples *primitive recursion in higher types*. Gödel's system T summarizes this definition pattern by constants for primitive recursion, $\mathsf{R}_{\mathsf{nat},\rho}\colon \rho \to (\mathsf{nat} \to \rho \to \rho) \to \mathsf{nat} \to \rho$, with the conversion rules

$$\begin{aligned}
\mathsf{R}_{\mathsf{nat},\rho} xy0 &\mapsto x\\
\mathsf{R}_{\mathsf{nat},\rho} xy\mathsf{S}(z) &\mapsto yz(\mathsf{R}_{\mathsf{nat},\rho} xyz)
\end{aligned}$$

Similar rules can be introduced for recursion constants for the other inductive types. In sections 5 we will also consider constants with rules that cannot be derived from primitive recursion.

By a *conversion* we mean a β-conversion or an instance of a constant-conversion rule, i.e. $L\theta \mapsto R\theta$ for some constant-conversion rule $L \mapsto R$ and substitution θ. We write $M \to_1 N$ if N is obtained from M by replacing one subterm occurrence of the left hand side of a conversion by its right hand side. We call a term M *strongly normalizing*, $\mathsf{SN}(M)$, if M is in the accessible part of the relation $\to_1$, i.e. there is no infinite reduction sequence $M \to_1 M_1 \to_1 M_2 \to_1 \ldots$. Equivalently, the predicate SN can be inductively defined by the rule

$$\frac{\forall K\,(M \to_1 K \quad \to \quad \mathsf{SN}(K))}{\mathsf{SN}(M)}$$

We call a system of constant-conversion rules $\mathcal{R}$ strongly normalizing if every term is strongly normalizing with respect to $\mathcal{R}$.

It is well-known that Gödel's system T, i.e. the system of conversion rules for primitive recursion in finite types is strongly normalizing. In the next section we will reexamine the proof of this fact using Tait's strong computability predicates and generalize it so as to accommodate further constants and conversions.

3 Proving Strong Normalization Using Strong Computability

We define for every type ρ what it means for a term M^ρ to be *strongly computable*, $\mathsf{SC}_\rho(M)$. The definition is by recursion on (the built up of) ρ. For an inductive type ι the predicate SC_ι is defined inductively. We only give the rules for a list type ρ^*. For boole and nat the rules are similar.

$$\mathsf{SC}_{\rho^*}([]) \qquad \frac{\mathsf{SC}_\rho(M) \qquad \mathsf{SC}_{\rho^*}(N)}{\mathsf{SC}_{\rho^*}(\mathsf{cons}(M,N))}$$

$$\frac{\forall K\,(M \to_1 K \;\to\; \mathsf{SC}_{\rho^*}(K))}{\mathsf{SC}_{\rho^*}(M)} \quad (M \mathit{notaconstructorterm})$$

$\mathsf{SC}_{\rho\to\sigma}$ is defined explicitely from SC_ρ and SC_σ.

$$\mathsf{SC}_{\rho\to\sigma}(M) \equiv \forall N\,(\mathsf{SC}_\rho(N) \to \mathsf{SC}_\sigma(MN))$$

Lemma 1. *(a) If* $\mathsf{SC}_\rho(M)$ *and* $M \to_1 M'$*, then* $\mathsf{SC}_\rho(M')$*.*
(b) A constructor term is strongly computable iff all its arguments are.

Proof. (a) Easy induction on ρ. If ρ is an inductive type the assertion is proved by a side induction on the definition of SC_ρ. For function types we use the (main) induction hypothesis.

(b) Obvious.

Lemma 2. *(a)* $\mathsf{SC}_\rho(M) \to \mathsf{SN}(M)$*.*
(b) $\mathsf{SC}_\rho(x)$ *for every variable* x *of type* ρ*.*

Proof. Induction on ρ. In order to get the proof through we need to strengthen part (b) to

(b') $\mathsf{SN}(A) \to \mathsf{SC}_\rho(A)$ for every term A with 'variable head',

where terms with variable head are variables and terms of the form AM where A is a term with variable head.

(a) If ρ is an inductive type, then the implication follows easily by a side induction on the definition of $\mathsf{SC}_\rho(M)$, possibly using the main induction hypothesis. *Case* $\rho \to \sigma$. Assume $\mathsf{SC}_{\rho\to\sigma}(M)$. By i.h. (b') we have $\mathsf{SC}(x^\rho)$. Hence $\mathsf{SC}_\sigma(Mx)$. By i.h. (a), $\mathsf{SN}(Mx)$. Hence $\mathsf{SN}(M)$.

(b') Let A be a strongly normalizing term with variable head. If A has an inductive type, then we show $\mathsf{SC}(A)$ by a side induction on $\mathsf{SN}(A)$. Since A is not a constructor term, it suffices to show $\mathsf{SC}(B)$ for all one step reducts B of A. Clearly B has variable head, hence $\mathsf{SC}(B)$ by side induction hypothesis. If A has type $\rho \to \sigma$, we assume $\mathsf{SC}\rho(M)$ and have to show $\mathsf{SC}_\sigma(AM)$. By induction hypothesis (a) we have $\mathsf{SN}(M)$. Hence $\mathsf{SN}(AM)$ (one easily proves $\mathsf{SN}(A) \wedge \mathsf{SN}(M) \to \mathsf{SN}(AM)$ for terms $A^{\rho\to\sigma}$ with variable head, since a reduction of AM can only take place in A or in M and any reduct of A has variable head). Hence $\mathsf{SC}(AM)$, by induction hypothesis (b').

We call a term *reactive* if it is an abstraction, or of the form $(cL_1 \dots L_k)\theta$ for some conversion rule $cL_1 \dots L_n \mapsto R$ with $n > k$ and some substitution θ. The property of a term M to be *neutral* is defined by recursion on M. If M is not a constructor term, then M is neutral if M is not reactive. If M is a constructor term, then M is neutral iff all its arguments are neutral. Clearly, if $M^{\rho\to\sigma}$ is

neutral, then for any term N^ρ the term is MN again neutral and any one step reduction of MN must happen by converting either M or N. However, neutral terms are *not* closed under one step reduction.

Lemma 3. *A neutral term is strongly computable iff all its one step reducts are.*

Proof. Because of Lemma 1 (a) it suffices to show that a neutral term M is strongly computable provided all of its one step reducts are. The proof is by induction on the type of M. For inductive types ι the assertion is proved by a side induction on neutral terms of type ι. Take, for example, we may assume $\iota = \rho^*$ If M^ι is not a constructor term, then the assertion holds by definition of SC_ι. Now let $M^\iota = \mathsf{cons}(M_1, M_2)$. Since we assume that all one step reducts of M are strongly computable, it follows, by Lemma 1 (b), that all one steps reducts of the arguments M_i are strongly computable. Hence, by main- respectively side-induction hypothesis, M_1 and M_2 are strongly computable. Let finally M be of type $\rho \to \sigma$. We show $\mathsf{SC}_\sigma(MN)$ for all strongly computable terms N by a side induction on $\mathsf{SN}(N)$. Since the term MN is neutral it suffices, by the main induction hypothesis, to show the strong computability of all its one step reducts. If M is reduced, we are done by assumption on M, if N is reduced, we use the side induction hypothesis.

Lemma 4. *If $M[N/x]$ is strongly computable for all strongly computable terms N, then $\lambda x\, M$ is strongly computable.*

Proof. Let $M^{\rho\to\sigma}$ fulfill the assumption of the lemma, and assume $\mathsf{SC}_\rho(N)$. We have to show $\mathsf{SC}_\sigma((\lambda x\, M)N)$. Since the latter term is neutral it suffices to show that all its one step reducts are strongly computable. By Lemma 2 (a) and Lemma 1 (a) we may argue by induction on $\mathsf{SN}(M, N)$. Assume $(\lambda x\, M)N \to_1 K$. If the conversion has happened within M or N, then we may use the induction hypothesis. If not, then we must have $K = M[N/x]$ which is strongly computable by assumption.

Proposition 5. *A term containing only strongly computable constants is strongly normalizable.*

Proof. By induction on terms M containing only strongly computable constants we show that $M\theta$ is strongly computable for every substitution θ such that $\theta(x)$ is strongly computable for all variables x in the domain of θ. For variables and constants this holds by assumption. For constructor terms and applications we use the induction hypothesis and the definition of strong computability. Abstractions are taken care of by the induction hypothesis and Lemma 4. The proposition now follows with the empty substitution and Lemma 2 (a).

Proposition 6. *Gödel's system T is strongly normalizing.*

Proof. By proposition 5 it suffices to show that all constants, i.e. the recursors are strongly normalizing. We have to show that $\mathsf{R}_{\sigma^*,\rho}MNL$ is strongly computable for all strongly computable terms M, N, L of appropriate types. Using

Lemma 2 (a) and Lemma 1 (a) we argue by induction on $\mathsf{SN}(M, N, L)$. We also use a side induction on L. Since $\mathsf{R}_{\sigma^*,\rho}MNL$ is a neutral term it suffices, by Lemma 3, to show $\mathsf{SC}_\rho(K)$ for all K such that $\mathsf{R}_{\sigma^*,\rho}MNL \to_1 K$. If the conversion took place within one of the terms in M, N, L, then we use the main induction hypothesis and Lemma 1 (a). Otherwise the visible recursor was involved in the conversion. If $L = []$ and $K = M$, then we are done since, by assumption, M is strongly computable. If $L = \mathsf{cons}(H, T)$ and $K = NHT(\mathsf{R}_{\sigma^*,\rho}MNT)$, then H and T are strongly computable, in particular $\mathsf{SC}_\rho(\mathsf{R}_{\sigma^*,\rho}MNT)$ by the side induction hypothesis. Again it follows that K is strongly computable.

4 Stratified Terms

For the rest of this paper we will restrict constant conversion rules to the form

$$cP_1 \dots P_n \mapsto R$$

where $\mathsf{FV}(cP_1 \dots P_n) \subseteq \mathsf{FV}(R)$ and the P_i are *constructor patterns*, i.e. terms built from variables by constructor application only. All examples of constant conversion rules we have seen so far are of this form.

The set of *stratified terms* is defined inductively as follows: Every variable is stratified; a constant c is stratified if for every rule $cP_1 \dots P_n \mapsto R$ the term R is stratified; a composite term is stratified if all its immediate subterms are.

Clearly a term is stratified iff it contains stratified constants only.

Note that stratification is a severe restriction. For example any constant with a recursive conversion rule, i.e. the constant reappears on the right hand side of the rule, is not stratified. We do not claim that stratified terms are of particular interest as such. We will just use them as a technical tool in our termination proof based on strict semantics (section 5).

Proposition 7. *Every stratified term is strongly normalizing.*

Proof. We proceed similarly as in the proof of proposition 5. By induction on the stratification of M we show that $M\theta$ is strongly computable for every substitution θ such that $\theta(x)$ is strongly computable for all variables $x \in \mathsf{FV}(M)$. Only the case that M is a constant is interesting. All other cases are as in proposition 5, that is, we use the induction hypothesis. Let c be a constant that takes n arguments. We have to show that $cM_1 \dots M_n$ is strongly computable for all strongly computable M_i. We do a side induction on the strong normalizability of the M_i (using Lemma 2 (a)). Since $cM_1 \dots M_n$ is neutral it suffices to show that all one step reducts of this term are strongly computable. If one of the M_i is reduced, we apply the side induction hypothesis. Otherwise there is a rule $cP_1 \dots P_n \mapsto R$ and a substitution θ with $(cP_1 \dots P_n)\theta = cM_1 \dots M_n$ and the reduct is $R\theta$. Since the P_i are constructor patterns, it follows from the strong computability of the M_i, by repeated application of Lemma 1 (b), that $\theta(x)$ is strongly computable for each $x \in \mathsf{FV}(M)$. Hence $R\theta$ is strongly computable, by the main induction hypothesis.

5 Strong Normalization Based on Strict Semantics

We will now develop a general semantic method for proving strong normalization of higher type rewrite systems. To begin with we discuss the rewrite systems we will apply this method to: Spector's bar recursion [13] and a version of bar recursion due to Berardi, Bezem and Coquand [2].

Let us write ρ^ω for $\mathsf{nat} \to \rho$, $\mathsf{if}\, B\, \mathsf{then}\, M\, \mathsf{else}\, N$ for $\mathsf{if}\, BMN$, $|M|$ for $\mathsf{lh}\, M$, $M{*}N$ for $M +\!\!+ \langle N\rangle$ where $\langle N\rangle := \mathsf{cons}(N, [])$, and $\widehat{M}$ for $\mathsf{get}\, M$. *Spector's bar recursion in finite types* is given (for each pair of types ρ, σ) by a constant

$$\Phi : (\rho^\omega \to \mathsf{nat}) \to (\rho^* \to \sigma) \to (\rho^* \to (\rho \to \sigma) \to \sigma) \to \rho^* \to \sigma$$

with the following defining equation

$$\Phi yghs = \mathsf{if}\, y\widehat{s} < |s|\, \mathsf{then}\, gs\, \mathsf{else}\, hs(\lambda x.\Phi ygh(s{*}x))$$

Turning this into a conversion rule would clearly not be strongly normalizing. Therefore we replace the right hand side by a call of an auxiliary constant Ψ with an extra boolean argument in order to force evaluation of the test $y\widehat{s} < |s|$ before the subterm $\Phi ygh(s{*}x)$ may be reduced further (Vogel's trick).

$$\begin{aligned} \Phi yghs &\mapsto \Psi yghs(y\widehat{s} < |s|) \\ \Psi yghs\mathsf{T} &\mapsto gs \\ \Psi yghs\mathsf{F} &\mapsto hs(\lambda x.\Phi ygh(s{*}x)) \end{aligned}$$

We denote the rewrite system above by *BR*.

Berardi, Bezem and Coquand's variant of bar recursion, which is also discussed in [4], is given by

$$\begin{aligned} &\Phi\colon (\rho^\omega \to \mathsf{nat}) \to (\mathsf{nat} \to (\rho \to \mathsf{nat}) \to \rho) \to \rho^* \to \mathsf{nat} \\ &\Phi ygs = y(\lambda k.\mathsf{if}\ k < |s|\, \mathsf{then}\, s_k\, \mathsf{else}\, gk(\lambda x.\Phi yg(s{*}x))) \end{aligned}$$

where $s_k := \mathsf{get}\, s\, k$. Applying Vogel's trick again we obtain the rewrite system

$$\begin{aligned} \Phi ygs &\mapsto y(\lambda k.\Psi ygsk(k < |s|)) \\ \Psi ygsk\mathsf{T} &\mapsto s_k \\ \Psi ygsk\mathsf{F} &\mapsto gk(\lambda x.\Phi yg(s{*}x)) \end{aligned}$$

which we call *MBR* (modified bar recursion).

The rewrite systems BR and MBR (and all rewrite systems discussed earlier) are instances of a class of rewrite systems which are distinguished by the fact that they induce a semantic interpretation of constants in a canonical way: A *functional rewrite system* is a system of constant-conversion rules

$$cP_1 \dots P_n \mapsto R$$

(P_i constructor patterns with $\mathsf{FV}(cP_1 \dots P_n) \subseteq \mathsf{FV}(R)$) which are *left linear*, i.e. a variable occurs at most once in the left hand side of a rule, and *mutually disjoint*, i.e. the left hand sides of two different rules are non-unifiable.

Our semantic method will work for arbitrary functional rewrite systems. The idea is to deduce the termination of a term M directly from the totality of the constants in M with respect to a strict interpretation of terms as total elements in the domain theoretic model $\hat{\mathrm{C}}$ of partial continuous functionals. The model $\hat{\mathrm{C}}$ assigns to every type ρ a Scott domain $\hat{\mathrm{C}}(\rho)$ such that $\hat{\mathrm{C}}(\rho \to \sigma) \equiv [\hat{\mathrm{C}}(\rho) \to \hat{\mathrm{C}}(\sigma)]$, the domain of continuous functions form $\hat{\mathrm{C}}(\rho)$ to $\hat{\mathrm{C}}(\sigma)$ where '$\equiv$' means 'isomorphic', and $\hat{\mathrm{C}}(\rho^*)$ is defined by the 'recursive domain equation'

$$\hat{\mathrm{C}}(\rho^*) \equiv 1 + \hat{\mathrm{C}}(\rho) \times \hat{\mathrm{C}}(\rho^*)$$

where '+' means the domain theoretic disjoint sum (adding a new bottom element). We denote the canonical injection of the one point space 1 into $\hat{\mathrm{C}}(\rho^*)$ by [] and the other canonical injection by cons. The definitions of $\hat{\mathrm{C}}(\mathsf{boole})$ and $\hat{\mathrm{C}}(\mathsf{nat})$ are similar.

The *total elements* in $\hat{\mathrm{C}}(\rho)$ are defined by recursion on ρ in the obvious way: A continuous function $f \in \hat{\mathrm{C}}(\rho \to \sigma)$ is total if $f(a)$ is total for all total arguments a. The set of total elements of $\hat{\mathrm{C}}(\rho^*)$ is given by an inductive definition: [] is total, and if $a \in \hat{\mathrm{C}}(\rho)$ and $b \in \hat{\mathrm{C}}(\sigma)$ are total, then $\mathsf{cons}(a, b)$ is total. Similar definitions apply to the other inductive types. Note that the total elements of $\hat{\mathrm{C}}(\rho^*)$ may be viewed as finite lists of total elements of $\hat{\mathrm{C}}(\rho)$ and the total elements of $\hat{\mathrm{C}}(\mathsf{boole})$ and $\hat{\mathrm{C}}(\mathsf{nat})$ are copies of the usual boolean values and the natural numbers respectively.

Let CEnv denote the domain of all *constant environments*, that is, families α assigning to each constant $c^\rho \in \mathcal{C}$ some $\alpha(c) \in \hat{\mathrm{C}}(\rho)$. Similarly, VEnv denotes the domain of all *variable environments*, i.e. families η assigning to each variable x^ρ some $\eta(x) \in \hat{\mathrm{C}}(\rho)$. For every term M^ρ we define the *strict semantics*, $[M]\colon \mathsf{CEnv} \to \mathsf{VEnv} \to \hat{\mathrm{C}}(\rho)$, by

$$\begin{aligned}
[x]^\alpha\eta &= \eta(x)\\
[c]^\alpha\eta &= \alpha(c)\\
[\lambda x\, M]^\alpha\eta(a) &= [M]^\alpha\eta^a_x\\
[MN]^\alpha\eta &= \begin{cases} [M]^\alpha\eta(a) & \text{if } a := [N]^\alpha\eta \neq \bot\\ \bot & \text{otherwise}\end{cases}\\
[\mathsf{co}(M_1, \ldots, M_k)]^\alpha\eta &= \begin{cases} \mathsf{co}(a_1, \ldots, a_k) & \text{if } a_i := [M_i]^\alpha\eta \neq \bot \text{ for all } i\\ \bot & \text{otherwise}\end{cases}
\end{aligned}$$

Lemma 8. *(a) If $\alpha(c)$ is total for all constants c in M and $\eta(x)$ is total for all $x \in \mathsf{FV}(M)$, then $[M]^\alpha\eta$ is total.*
(b) If $\alpha(c) = \bot$ for some constant c occurring in M, then $[M]^\alpha\eta = \bot$.
(c) $[M]^\alpha([\theta]^\alpha\eta) = [M\theta]^\alpha\eta$ where $([\theta]^\alpha\eta)(x) := [\theta(x)]^\alpha\eta$.
(d) $[M]^{[\zeta]^\alpha\eta}\eta = [M\zeta]^\alpha\eta$ where ζ is a 'constant substitution', i.e. $\zeta(c^\rho)$ is a term of type ρ for each constant c^ρ, and $([\zeta]^\alpha\eta)(c) := [\zeta(c)]^\alpha\eta$.

Proof. Easy inductions on M.

Next we define the semantics of constants induced by their conversion rules.

We let $\bot$ denote the 'undefined' variable environment, i.e. $\bot(x) = \bot_\rho$ for all variables x^ρ.

For a vector $\boldsymbol{P}\colon\boldsymbol{\rho}$ of constructor patterns containing each variable at at most one place and $\boldsymbol{a} \in \hat{\mathrm{C}}(\boldsymbol{\rho})$ we define the $\boldsymbol{P}$-*predecessor* of $\boldsymbol{a}$, $\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{a}) \in \mathsf{VEnv}$, by recursion on the number of constructors occurring in $\boldsymbol{P}$.

$$\begin{aligned}
\mathsf{pred}_{\boldsymbol{x}}(\boldsymbol{a}) &= \bot^{\boldsymbol{a}}_{\boldsymbol{x}} \\
\mathsf{pred}_{\boldsymbol{x},\mathsf{co}(\boldsymbol{Q}),\boldsymbol{P}}(\boldsymbol{a},\mathsf{co}(\boldsymbol{b}),\boldsymbol{c}) &= \mathsf{pred}_{\boldsymbol{x},\boldsymbol{Q},\boldsymbol{P}}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}) \\
\mathsf{pred}_{\boldsymbol{x},\mathsf{co}(\boldsymbol{Q}),\boldsymbol{P}}(\boldsymbol{a},b,\boldsymbol{c}) &= \bot \quad \text{if } b \text{ is not of the form } \mathsf{co}(\boldsymbol{b})
\end{aligned}$$

We say $\boldsymbol{a}$ *matches* $\boldsymbol{P}$ if in the definition of $\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{\alpha})$ the last clause has never been used.

Let $\mathcal{R}$ be a functional rewrite system. For a given vector $\boldsymbol{a} \in \hat{\mathrm{C}}$ there can be at most one rule $c\boldsymbol{P} \mapsto R \in \mathcal{R}$ such that $\boldsymbol{a}$ matches $\boldsymbol{P}$, because the rules in $\mathcal{R}$ are mutually disjoint. Therefore the following operator $\Gamma_{\mathcal{R}}\colon \mathsf{CEnv} \to \mathsf{CEnv}$ is welldefined and continuous.

$$\Gamma_{\mathcal{R}}(\alpha)(c)(\boldsymbol{a}) := \bigsqcup\{[R]^{\alpha}\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{a}) \mid c\boldsymbol{P} \mapsto R \in \mathcal{R},\ \boldsymbol{a} \text{ matches } \boldsymbol{P}\}$$

We define the constant environment $\alpha_{\mathcal{R}}$ as the least fixed point of $\Gamma_{\mathcal{R}}$.

We now state the main the result of this paper.

Theorem 9. *Let $\mathcal{R}$ be a functional rewrite system. If $\alpha_{\mathcal{R}}(c)$ is total for every constant in M, then M is strongly normalizing.*

The proof of this theorem needs some preparation. In the following we fix a functional rewrite system $\mathcal{R}$.

Lemma 10. *Let $c\boldsymbol{P} \mapsto R \in \mathcal{R}$ be a rule, θ a substitution and η a variable environment. Set $\boldsymbol{a} := [\boldsymbol{P}\theta]^{\alpha_{\mathcal{R}}}\eta$. If $\bot \notin \boldsymbol{a}$, then $\boldsymbol{a}$ matches $\boldsymbol{P}$ and $\alpha_{\mathcal{R}}(c)(\boldsymbol{a}) = [R]^{\alpha_{\mathcal{R}}}\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{a})$.*

Proof. That $[\boldsymbol{P}\theta]^{\alpha_{\mathcal{R}}}\eta$ matches $\boldsymbol{P}$ is easily shown by induction on the number of constructors in $\boldsymbol{P}$. The rest follows immediately from the definition of $\Gamma_{\mathcal{R}}$.

Lemma 11. *If $M \to_1 N$, then $[M]^{\alpha_{\mathcal{R}}}\eta \sqsubseteq [N]^{\alpha_{\mathcal{R}}}\eta$.*

Proof. Induction on M, where w.l.o.g. we assume $[M]^{\alpha_{\mathcal{R}}}\eta \neq \bot$.

Case $(\lambda xM)N \to_1 M[N/x]$. Setting $a := [N]^{\alpha_{\mathcal{R}}}\eta$ we have $[(\lambda xM)N]^{\alpha_{\mathcal{R}}}\eta \sqsubseteq ([\lambda xM]^{\alpha_{\mathcal{R}}}\eta)(a) = [M]^{\alpha_{\mathcal{R}}}\eta^a_x = [M[N/x]]^{\alpha_{\mathcal{R}}}\eta$, by Lemma 8 (c).

Case $c\boldsymbol{P}\theta \to_1 R\theta$ for some rule $c\boldsymbol{P} \mapsto R \in \mathcal{R}$. $[c\boldsymbol{P}\theta]^{\alpha_{\mathcal{R}}}\eta \sqsubseteq \alpha_{\mathcal{R}}(c)(\boldsymbol{a}) = [R]^{\alpha_{\mathcal{R}}}\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{a}) = [R\theta]^{\alpha_{\mathcal{R}}}\eta$. The last two equations hold by Lemmas 10 and 8 (c).

All other cases (i.e. conversion of a proper subterm) follow immediately from the induction hypothesis and the fact that constructors and application are interpreted strictly.

We now introduce a stratified variant $\mathcal{R}_\omega$ of $\mathcal{R}$. Let $\mathcal{C}$ be the set of constants of $\mathcal{R}$. For each $c \in \mathcal{C}$ and every natural number n we introduce a new constant

c_n. Set $\mathcal{C}_\omega := \{c_n \mid c \in \mathcal{C},\ n \in \omega\}$. For any $\mathcal{C}$-term M let $M_{[n]}$ be the $\mathcal{C}_\omega$-term obtained from M by replacing every occurring constant c by c_n. We set

$$\mathcal{R}_\omega := \{c_{n+1}\boldsymbol{P} \mapsto R_{[n]} \mid c\boldsymbol{P} \mapsto R \in \mathcal{R},\ n \in \omega\}$$

Clearly $\mathcal{R}_\omega$ is again a functional rewrite system. Furthermore all constants c_n are stratified (induction on n). We write $A \preceq M$ if M is a $\mathcal{C}$-term and A is a $\mathcal{C}_\omega$-term obtained from M by replacing every occurrence of a constant c by some c_n (different occurrences of the same constant may receive different indices).

Lemma 12. *If $A \preceq M$ and A contains no constant of the form c_0, then to every $\mathcal{C}$-term M' such that $M \to_1 M'$ there is a $\mathcal{C}_\omega$-term A' such that $A \to_1 A'$ and $A' \preceq M'$.*

Proof. Easy induction on M.

We define the $\mathcal{C}_\omega$-constant environment $\alpha_{\mathcal{R}_\omega}$ like the $\mathcal{C}$-constant environment $\alpha_{\mathcal{R}_\omega}$, but with $\mathcal{R}$ replaced by $\mathcal{R}_\omega$. Hence

$$\alpha_{\mathcal{R}_\omega}(c_{n+1})(\boldsymbol{a}) = \bigsqcup\{[R_{[n]}]^{\alpha_{\mathcal{R}_\omega}}\,\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{a}) \mid c\boldsymbol{P} \mapsto R \in \mathcal{R},\ \boldsymbol{a} \text{ matches } \boldsymbol{P}\}$$

and $\alpha_{\mathcal{R}_\omega}(c_0) = \bot$ (there is no rule for c_0).

Lemma 13. *$\alpha_{\mathcal{R}}(c) = \bigsqcup_{n\in\omega} \alpha_{\mathcal{R}_\omega}(c_n)$ and $\alpha_{\mathcal{R}_\omega}(c_n) \sqsubseteq \alpha_{\mathcal{R}_\omega}(c_{n+1})$ for every constant $c \in \mathcal{C}$.*

Proof. Set $\alpha_n(c) := \alpha_{\mathcal{R}_\omega}(c_n)$. Since $\alpha_{\mathcal{R}} = \bigsqcup_{n\in\omega} \Gamma^n_{\mathcal{R}}\bot$ and $\Gamma^n_{\mathcal{R}}\bot \sqsubseteq \Gamma^{n+1}_{\mathcal{R}}\bot$, it suffices to show $\alpha_n = \Gamma^n_{\mathcal{R}}\bot$ for all n. We prove this by induction on n. For $n = 0$ both sides are $\bot$.

$$\begin{aligned}
\alpha_{n+1}(c)(\boldsymbol{a}) &= \alpha_{\mathcal{R}_\omega}(c_{n+1})(\boldsymbol{a}) \\
&= \bigsqcup\{[R_{[n]}]^{\alpha_{\mathcal{R}_\omega}}\,\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{a}) \mid c\boldsymbol{P} \mapsto R \in \mathcal{R},\ \boldsymbol{a} \text{ matches } \boldsymbol{P}\} \\
&= \bigsqcup\{[R]^{\alpha_n}\,\mathsf{pred}_{\boldsymbol{P}}(\boldsymbol{a}) \mid c\boldsymbol{P} \mapsto R \in \mathcal{R},\ \boldsymbol{a} \text{ matches } \boldsymbol{P}\} \text{ (Lemma 8 (d))} \\
&= \Gamma_{\mathcal{R}}(\alpha_n)(c)(\boldsymbol{a}) \\
&= (\Gamma^{n+1}_{\mathcal{R}}\bot)(c)(\boldsymbol{a}) \text{ (induction hypothesis)}
\end{aligned}$$

Lemma 14. *$[M]^{\alpha_{\mathcal{R}}}\eta = \bigsqcup_{n\in\omega}[M_{[n]}]^{\alpha_{\mathcal{R}_\omega}}\eta$ for every $\mathcal{C}$-term M and every variable environment η.*

Proof. Set, as in the previous proof, $\alpha_n(c) := \alpha_{\mathcal{R}_\omega}(c_n)$. By Lemma 13 we have $\alpha_{\mathcal{R}} = \bigsqcup_{n\in\omega} \alpha_n$ where $\alpha_n \sqsubseteq \alpha_{n+1}$. Hence, because $[M]$ is a continuous function,

$$[M]^{\alpha_{\mathcal{R}}}\eta = \bigsqcup_{n\in\omega} [M]^{\alpha_n}\eta \overset{8\,(\mathrm{d})}{=} \bigsqcup_{n\in\omega} [M_{[n]}]^{\alpha_{\mathcal{R}_\omega}}\eta$$

Now we are ready to prove Theorem 9. Let M be a ($\mathcal{C}$-)term such that $\alpha_{\mathcal{R}}(c)$ is total for every constant in M. Let η be any total environment. Then $[M]^{\alpha_{\mathcal{R}}}\eta$ is

total, by Lemma 8 (a), and therefore different from $\bot$. By Lemma 14 it follows that there is some n such that $[M_{[n]}]^{\alpha\mathcal{R}_\omega}\eta \neq \bot$. Clearly $M_{[n]} \preceq M$. Therefore it suffices to show that whenever $A \preceq N$ and $[A]^{\alpha\mathcal{R}_\omega}\eta \neq \bot$, then N is strongly normalizing. We prove this by induction on the strong normalizability of the $\mathcal{C}_\omega$-term A, using proposition 7. We need to show that all one step reducts of N are strongly normalizing. So, assume $N \to_1 N'$. Since $[A]^{\alpha\mathcal{R}_\omega}\eta \neq \bot$ we know, by Lemma 8 (b), that A does not contain a constant of the form c_0. It follows with Lemma 12 that $A \to_1 A'$ with $A' \preceq N'$ for some $\mathcal{C}_\omega$-term A'. By Lemma 11 $[A']^{\alpha\mathcal{R}_\omega}\eta \neq \bot$, hence we can apply the induction hypothesis to A' and N'.

Let us now apply Theorem 9 to prove strong normalization for bar recursion.

Theorem 15. *Gödel's systemT extended by BR and MBR is strongly normalizing.*

Proof. We only carry out the proof for MBR. For BR the proof is similar and slightly simpler. Our strict semantics interprets the constants Φ and Ψ of MBR as continuous functionals φ and ψ which satisfy for *total arguments* y, g, s, k (in $\hat{\mathrm{C}}$) the equations

$$\begin{aligned}
\varphi ygs &= y(\lambda k.\psi ygsk(k < |s|)) \\
\psi ygsk\mathsf{T} &= s_k \\
\psi ygsk\mathsf{F} &= \begin{cases} gk(\lambda x.\varphi yg(s{*}x)) & \text{if } \varphi yg(s{*}x) \neq \bot \text{ for some } x \\ \bot & \text{otherwise} \end{cases}
\end{aligned}$$

By a continuity argument one shows that for every total y the the binary relation $\gg_y$ on the total elements of type ρ^* defined by

$$s \gg_y t :\equiv y(\lambda k.\mathsf{if}\ k < |s|\ \mathsf{then}\ s_k\ \mathsf{else}\ \bot) = \bot \wedge s{*}a = t \text{ for some total } a$$

is wellfounded. Now the totality of φygs and ψygs for total y, g, s can be proven easily by induction on $\gg_y$. With Theorem 9 strong normalization follows.

Remarks. Tait [14], Vogel [17], Luckhardt [10] and Bezem [5] proved strong normalization for BR formulated in a combinatorial calculus. Our result is slightly stronger since we work in a λ-calculus framework which allows more reductions. Strong normalization for MBR is completely new. Further interesting rewrite rules where Theorem 9 applies to are realizers of the negative- and A-translations of the axiom schemes of countable choice [2] and open induction [3].

From a logical point of view our proof is roughly equivalent to the proofs in the work cited, since the partial continuous functionals can be defined primitive recursively (finite neighborhoods, or compact elements of Scott domains) and totality in $\hat{\mathrm{C}}(\rho)$ has the same logical complexity as, say the definition of strong computability for infinite terms of type ρ.

References

1. H. Barendregt. Lambda calculi with types. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Clarendon Press, Oxford, 1992.
2. S. Berardi, M. Bezem, and T. Coquand. On the computational content of the axiom of choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998.
3. U. Berger. A computational interpretation of open induction. In F. Titsworth, editor, *Proceedings of the Nineteenth Annual IEEE Symposium on Logic in Computer Science*, pages 326–334. IEEE Computer Society, 2004.
4. U. Berger and P. Oliva. Modified bar recursion and classical dependent choice. In *Logic Colloquium 2001*. Springer, to appear.
5. M. Bezem. Strong normalization of barrecursive terms without using infinite terms. *Archive for Mathematical Logic*, 25:175–181, 1985.
6. F Blanqui, J-P. Jouannaud, and M. Okada. The calculus of algebraic constructions. In P. Narendran and M. Rusinowitch, editors, *Proceedings of RTA'99*, number 1631 in LNCS, pages 301–316. Springer Verlag, Berlin, Heidelberg, New York, 1999.
7. T. Coquand. *Une théorie des constructions*. PhD thesis, Université Paris VII, 1985.
8. H. Geuvers and M.J. Nederhof. A modular proof of strong normalization for the calculus of constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.
9. J-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J.E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 63–92. North–Holland, Amsterdam, 1971.
10. H. Luckhardt. *Extensional Gödel Functional Interpretation – A Consistency Proof of Classical Analysis*, volume 306 of *Lecture Notes in Mathematics*. Springer, 1973.
11. R. Matthes. Monotone inductive and coinductive constructors of rank 2. In L Fribourg, editor, *Computer Science Logic (Proceedings of the Fifteenth CSL Conference)*, number 2142 in LNCS, pages 600–615. Springer Verlag, Berlin, Heidelberg, New York, 2001.
12. G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.
13. C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In F. D. E. Dekker, editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island, 1962.
14. W.W. Tait. Normal form theorem for barrecursive functions of finite type. In J.E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 353–367. North–Holland, Amsterdam, 1971.
15. A.S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer, 1973.
16. J. van de Pol and H. Schwichtenberg. Strict functionals for termination proofs. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Typed Lambda Calculi and Applications*, volume 902 of *LNCS*, pages 350–364. Springer Verlag, Berlin, Heidelberg, New York, 1995.
17. H. Vogel. Ein starker Normalisationssatz für die barrekursiven Funktionale. *Archive for Mathematical Logic*, 18:81–84, 1985.

A Thread Algebra with Multi-level Strategic Interleaving

Jan A. Bergstra[1,2] and C.A. (Kees) Middelburg[3]

[1] Programming Research Group, University of Amsterdam,
P.O. Box 41882, 1009 DB Amsterdam, the Netherlands
janb@science.uva.nl
[2] Department of Philosophy, Utrecht University,
P.O. Box 80126, 3508 TC Utrecht, the Netherlands
janb@phil.uu.nl
[3] Computing Science Department, Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
keesm@win.tue.nl

Abstract. In a previous paper, we developed an algebraic theory of threads and multi-threads based on strategic interleaving. This theory includes a number of plausible interleaving strategies on thread vectors. The strategic interleaving of a thread vector constitutes a multi-thread. Several multi-threads may exist concurrently on a single host in a network, several host behaviors may exist concurrently in a single network on the internet, etc. Strategic interleaving is also present at these other levels. In the current paper, we extend the theory developed so far with features to cover multi-level strategic interleaving.

1 Introduction

A thread is the behavior of a deterministic sequential program under execution. Multi-threading refers to the concurrent existence of several threads in a program under execution. Multi-threading is the dominant form of concurrency provided by recent object-oriented programming languages such as Java [1] and C# [2]. Arbitrary interleaving, on which theories about concurrent processes such as ACP [3] are based, is not the appropriate intuition when dealing with multi-threading. In the case of multi-threading, some deterministic interleaving strategy is used. In [4], we introduced a number of plausible deterministic interleaving strategies for multi-threading. We also proposed to use the phrase strategic interleaving for the more constrained form of interleaving obtained by using such a strategy.

The strategic interleaving of a thread vector constitutes a multi-thread. In conventional operating system jargon, a multi-thread is called a process. Several multi-threads may exist concurrently on the same machine. Multi-processing refers to the concurrent existence of several multi-threads on a machine. Such machines may be hosts in a network, and several host behaviors may exist concurrently in

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 35–48, 2005.

the same network. And so on and so forth. Strategic interleaving is also present at these other levels. In the current paper, we extend the theory developed so far with features to cover multi-level strategic interleaving. There is a dependence on the interleaving strategy considered. We extend the theory only for the simplest case: cyclic interleaving. Other plausible interleaving strategies are treated in [4]. They can also be adapted to the setting of multi-level strategic interleaving.

Threads proceed by performing steps, in the sequel called basic actions, in a sequential fashion. Performing a basic action is taken as making a request to a certain service provided by the execution environment to process a certain command. The service produces a reply value which is returned to the thread concerned. A service may be local to a single thread, local to a multi-thread, local to a host, or local to a network. In this paper, we introduce thread-service composition in order to bind certain basic actions of a thread to certain services.

An axiomatic description of multi-level strategic interleaving and thread-service composition, as well as a structural operational semantics, is provided. One of our objectives is to develop a simplified, formal representation schema of the design of systems that consist of several multi-threaded programs on various hosts in different networks. We propose to use the term formal design prototype for such a schema. Evidence of the correctness of the presented schema is obtained by a simulation lemma, which states that a finite thread consisting of basic actions that will not be processed by any available service is simulated by any instance of the schema that contains the thread in one of its thread vectors.

Thread algebra with multi-level strategic interleaving is a design on top of BPPA (Basic Polarized Process Algebra) [5, 6]. BPPA is far less general than ACP-style process algebras and its design focuses on the semantics of deterministic sequential programs. The semantics of a deterministic sequential program is supposed to be a polarized process. Polarization is understood along the axis of the client-server dichotomy. Basic actions in a polarized process are either requests expecting a reply or service offerings promising a reply. Thread algebra may be viewed as client-side polarized process algebra because all threads are viewed as clients generating requests for services provided by their environment.

The structure of this paper is as follows. After a review of BPPA, we extend it to a basic thread algebra with cyclic interleaving, but without any feature for multi-level strategic interleaving. Next, we extend this basic thread algebra with thread-service composition and other features for multi-level strategic interleaving. Following this, we discuss how two additional features can be expressed and give a formal representation schema of the design of systems that consist of several multi-threaded programs on various hosts in different networks. Finally, we make some concluding remarks.

2 Basic Polarized Process Algebra

In this section, we review BPPA (Basic Polarized Process Algebra), a form of process algebra which is tailored to the use for the description of the behavior of deterministic sequential programs under execution.

Table 1. Axiom of BPPA

$x \trianglelefteq \mathsf{tau} \trianglerighteq y = x \trianglelefteq \mathsf{tau} \trianglerighteq x$	T1

In BPPA, it is assumed that there is a fixed but arbitrary finite set of *basic actions* $\mathcal{A}$ with $\mathsf{tau} \notin \mathcal{A}$. We write $\mathcal{A}_{\mathsf{tau}}$ for $\mathcal{A} \cup \{\mathsf{tau}\}$. BPPA has the following constants and operators:

- the *deadlock* constant D;
- the *termination* constant S;
- for each $a \in \mathcal{A}_{\mathsf{tau}}$, a binary *postconditional composition* operator $\_ \trianglelefteq a \trianglerighteq \_$.

We use infix notation for postconditional composition. We introduce *action prefixing* as an abbreviation: $a \circ p$, where p is a term of BPPA, abbreviates $p \trianglelefteq a \trianglerighteq p$.

The intuition is that each basic action is taken as a command to be processed by the execution environment. The processing of a command may involve a change of state of the execution environment. At completion of the processing of the command, the execution environment produces a reply value. This reply is either T or F and is returned to the polarized process concerned. Let p and q be closed terms of BPPA. Then $p \trianglelefteq a \trianglerighteq q$ will proceed as p if the processing of a leads to the reply T (called a positive reply), and it will proceed as q if the processing of a leads to the reply F (called a negative reply). If the reply is used to indicate whether the processing was successful, a useful convention is to indicate successful processing by the reply T and unsuccessful processing by the reply F. The action tau plays a special role. Its execution will never change any state and always produces a positive reply.

BPPA has only one axiom. This axiom is given in Table 1. Using the abbreviation introduced above, axiom T1 can be written as follows: $x \trianglelefteq \mathsf{tau} \trianglerighteq y = \mathsf{tau} \circ x$.

Following [6], a CPO structure can be imposed on the domain of BPPA. Then guarded recursion equations represent continuous operators having appropriate fixed points. These matters will not be repeated here, taking for granted that guarded systems of recursion equations allow one to define unique polarized processes. Guardedness is the requirement that repeated substitution of the right-hand sides of equations for the left-hand side variables eventually produces an expression of the form D, S or $p \trianglelefteq a \trianglerighteq q$. For each guarded system of recursion equations E and each variable X that occurs as the left-hand side of an equation in E, we add to the constants of BPPA a constant standing for the unique solution of E for X. This constant is denoted by X_E.

The projective limit characterization of process equivalence on polarized processes is based on the notion of a finite approximation of depth n. When for all n these approximations are identical for two given polarized processes, both processes are considered identical. This allows one to eliminate recursion in favor of the infinitary proof rule AIP. Following [5], which in fact uses the notation of [3], approximation of depth n is phrased in terms of a unary *projection* operator $\pi_n(\_)$. The projection operators are defined inductively by means of the

Table 2. Axioms for projection

$\pi_0(x) = \mathsf{D}$	P0
$\pi_{n+1}(\mathsf{S}) = \mathsf{S}$	P1
$\pi_{n+1}(\mathsf{D}) = \mathsf{D}$	P2
$\pi_{n+1}(x \trianglelefteq a \trianglerighteq y) = \pi_n(x) \trianglelefteq a \trianglerighteq \pi_n(y)$	P3
$(\bigwedge_{n \geq 0} \pi_n(x) = \pi_n(y)) \Rightarrow x = y$	AIP

axioms in Table 2. In this table and all subsequent tables with axioms in which a occurs, a stands for an arbitrary action from $\mathcal{A}_{\mathsf{tau}}$.

As mentioned above, the behavior of a polarized process depends upon its execution environment. Each basic action performed by the polarized process is taken as a command to be processed by the execution environment. At any stage, the commands that the execution environment can accept depend only on its history, i.e. the sequence of commands processed before and the sequence of replies produced for those commands. When the execution environment accepts a command, it will produce a positive reply if its processing succeeds and a negative reply if its processing fails. Whether the processing of the command succeeds or fails usually depends on the execution history. However, it may also depend on external conditions.

In the structural operational semantics, we represent an execution environment by a function $\rho : (\mathcal{A} \times \{\mathsf{T}, \mathsf{F}\})^* \to \mathcal{P}(\mathcal{A} \times \{\mathsf{T}, \mathsf{F}\})$ that satisfies the following condition: $(a, b) \notin \rho(\alpha) \Rightarrow \rho(\alpha \frown \langle (a, b) \rangle) = \emptyset$ for all $a \in \mathcal{A}$, $b \in \{\mathsf{T}, \mathsf{F}\}$ and $\alpha \in (\mathcal{A} \times \{\mathsf{T}, \mathsf{F}\})^*$.[1] We write $\mathcal{E}$ for the set of all those functions. Given an execution environment $\rho \in \mathcal{E}$ and a basic action $a \in \mathcal{A}$, the *derived* execution environment of ρ after processing a with *success*, written $\frac{\partial^+}{\partial a} \rho$, is defined by $\frac{\partial^+}{\partial a} \rho(\alpha) = \rho(\langle (a, \mathsf{T}) \rangle \frown \alpha)$; and the *derived* execution environment of ρ after processing a with *failure*, written $\frac{\partial^-}{\partial a} \rho$, is defined by $\frac{\partial^-}{\partial a} \rho(\alpha) = \rho(\langle (a, \mathsf{F}) \rangle \frown \alpha)$.

The following transition relations on closed terms are used in the structural operational semantics of BPPA:

- a binary relation $\langle \_, \rho \rangle \xrightarrow{a} \langle \_, \rho' \rangle$ for each $a \in \mathcal{A}_{\mathsf{tau}}$ and $\rho, \rho' \in \mathcal{E}$;
- a unary relation $\langle \_, \rho \rangle \downarrow$ for each $\rho \in \mathcal{E}$;
- a unary relation $\langle \_, \rho \rangle \uparrow$ for each $\rho \in \mathcal{E}$.

The three kinds of transition relations are called the *action step*, *termination*, and *deadlock* relations, respectively. They can be explained as follows:

- $\langle p, \rho \rangle \xrightarrow{a} \langle p', \rho' \rangle$: in execution environment ρ, process p is capable of first performing action a and then proceeding as process p' in execution environment ρ';

[1] We write $\langle\,\rangle$ for the empty sequence, $\langle d \rangle$ for the sequence having d as sole element, and $\alpha \frown \beta$ for the concatenation of sequences α and β. We assume that the identities $\alpha \frown \langle\,\rangle = \langle\,\rangle \frown \alpha = \alpha$ hold.

Table 3. Transition rules for BPPA with projection and recursion

$$\frac{}{\langle \mathsf{S}, \rho\rangle\downarrow} \qquad \frac{}{\langle \mathsf{D}, \rho\rangle\uparrow}$$

$$\frac{}{\langle x \trianglelefteq a \trianglerighteq y, \rho\rangle \xrightarrow{a} \langle x, \frac{\partial^+}{\partial a}\,\rho\rangle}\ (a,\mathsf{T}) \in \rho(\langle\,\rangle) \qquad \frac{}{\langle x \trianglelefteq a \trianglerighteq y, \rho\rangle \xrightarrow{a} \langle y, \frac{\partial^-}{\partial a}\,\rho\rangle}\ (a,\mathsf{F}) \in \rho(\langle\,\rangle)$$

$$\frac{}{\langle x \trianglelefteq a \trianglerighteq y, \rho\rangle\uparrow}\ (a,\mathsf{T}) \notin \rho(\langle\,\rangle),\ (a,\mathsf{F}) \notin \rho(\langle\,\rangle) \qquad \frac{}{\langle x \trianglelefteq \mathsf{tau} \trianglerighteq y, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x, \rho\rangle}$$

$$\frac{\langle x, \rho\rangle \xrightarrow{a} \langle x', \rho'\rangle}{\langle \pi_{n+1}(x), \rho\rangle \xrightarrow{a} \langle \pi_n(x'), \rho'\rangle} \qquad \frac{\langle x, \rho\rangle\downarrow}{\langle \pi_{n+1}(x), \rho\rangle\downarrow} \qquad \frac{\langle x, \rho\rangle\uparrow}{\langle \pi_{n+1}(x), \rho\rangle\uparrow} \qquad \frac{}{\langle \pi_0(x), \rho\rangle\uparrow}$$

$$\frac{\langle t_E, \rho\rangle \xrightarrow{a} \langle x', \rho'\rangle}{\langle X_E, \rho\rangle \xrightarrow{a} \langle x', \rho'\rangle}\ X{=}t \in E \qquad \frac{\langle t_E, \rho\rangle\downarrow}{\langle X_E, \rho\rangle\downarrow}\ X{=}t \in E \qquad \frac{\langle t_E, \rho\rangle\uparrow}{\langle X_E, \rho\rangle\uparrow}\ X{=}t \in E$$

- $\langle p, \rho\rangle\downarrow$: in execution environment ρ, process p is capable of terminating successfully;
- $\langle p, \rho\rangle\uparrow$: in execution environment ρ, process p is neither capable of performing an action nor capable of terminating successfully.

The structural operational semantics of BPPA extended with projection and recursion is described by the transition rules given in Table 3. In this table and all subsequent tables with transition rules in which a occurs, a stands for an arbitrary action from $\mathcal{A}_{\mathsf{tau}}$. We write t_E for t with, for all X that occur on the left-hand side of an equation in E, all occurrences of X in t replaced by X_E.

Bisimulation equivalence is defined as follows. A *bisimulation* is a symmetric binary relation B on closed terms such that for all closed terms p and q:

- if $B(p,q)$ and $\langle p, \rho\rangle \xrightarrow{a} \langle p', \rho'\rangle$, then there is a q' such that $\langle q, \rho\rangle \xrightarrow{a} \langle q', \rho'\rangle$ and $B(p', q')$;
- if $B(p,q)$ and $\langle p, \rho\rangle\downarrow$, then $\langle q, \rho\rangle\downarrow$;
- if $B(p,q)$ and $\langle p, \rho\rangle\uparrow$, then $\langle q, \rho\rangle\uparrow$.

Two closed terms p and q are *bisimulation equivalent*, written $p \leftrightarrows q$, if there exists a bisimulation B such that $B(p,q)$.

Bisimulation equivalence is a congruence with respect to the postconditional composition operators and the projection operators. This follows immediately from the fact that the transition rules for BPPA with projection and recursion constitute a transition system specification in path format (see e.g. [7]).

3 Basic Thread Algebra with Foci and Methods

In this section, we introduce a thread algebra without features for multi-level strategic interleaving. Such features will be added in subsequent sections.

Table 4. Axioms for cyclic interleaving

$\parallel(\langle\rangle) = \mathsf{S}$	CSI1
$\parallel(\langle \mathsf{S}\rangle \frown \alpha) = \parallel(\alpha)$	CSI2
$\parallel(\langle \mathsf{D}\rangle \frown \alpha) = \mathsf{S}_\mathsf{D}(\parallel(\alpha))$	CSI3
$\parallel(\langle \mathsf{tau} \circ x\rangle \frown \alpha) = \mathsf{tau} \circ \parallel(\alpha \frown \langle x\rangle)$	CSI4
$\parallel(\langle x \trianglelefteq f.m \trianglerighteq y\rangle \frown \alpha) = \parallel(\alpha \frown \langle x\rangle) \trianglelefteq f.m \trianglerighteq \parallel(\alpha \frown \langle y\rangle)$	CSI5

Table 5. Axioms for deadlock at termination

$\mathsf{S}_\mathsf{D}(\mathsf{S}) = \mathsf{D}$	S2D1
$\mathsf{S}_\mathsf{D}(\mathsf{D}) = \mathsf{D}$	S2D2
$\mathsf{S}_\mathsf{D}(\mathsf{tau} \circ x) = \mathsf{tau} \circ \mathsf{S}_\mathsf{D}(x)$	S2D3
$\mathsf{S}_\mathsf{D}(x \trianglelefteq f.m \trianglerighteq y) = \mathsf{S}_\mathsf{D}(x) \trianglelefteq f.m \trianglerighteq \mathsf{S}_\mathsf{D}(y)$	S2D4

In [5], its has been outlined how and why polarized processes are a natural candidate for the specification of the semantics of deterministic sequential programs. Assuming that a thread is a process representing a deterministic sequential program under execution, it is reasonable to view all polarized processes as threads. A thread vector is a sequence of threads.

Strategic interleaving operators turn a thread vector of arbitrary length into a single thread. This single thread obtained via a strategic interleaving operator is also called a multi-thread. Formally, however both threads and multi-threads are polarized processes. In this paper, we only cover the simplest interleaving strategy, namely *cyclic interleaving*. Other plausible interleaving strategies are treated in [4]. They can also be adapted to the features for multi-level level strategic interleaving that will be introduced in the current paper. The strategic interleaving operator for cyclic interleaving is denoted by $\parallel(\_)$. In [4], it was denoted by $\parallel_{csi}(\_)$ to distinguish it from other strategic interleaving operators.

It is assumed that there is a fixed but arbitrary finite set of *foci* $\mathcal{F}$ and a fixed but arbitrary finite set of *methods* $\mathcal{M}$. For the set of basic actions $\mathcal{A}$, we take the set $\{f.m \mid f \in \mathcal{F}, m \in \mathcal{M}\}$. Each focus plays the role of a name of a service provided by the execution environment that can be requested to process a command. Each method plays the role of a command proper. Performing a basic action $f.m$ is taken as making a request to the service named f to process the command m.

The axioms for cyclic interleaving are given in Table 4. In this table and all subsequent tables with axioms or transition rules in which f and m occur, f and m stand for an arbitrary focus from $\mathcal{F}$ and an arbitrary method from $\mathcal{M}$, respectively. In CSI3, the auxiliary *deadlock at termination* operator $\mathsf{S}_\mathsf{D}(\_)$ is used. This operator turns termination into deadlock. Its axioms appear in Table 5.

The structural operational semantics of the basic thread algebra with foci and methods is described by the transition rules given in Tables 3 and 6. Here $\langle x, \rho\rangle \not\rightarrow$

Table 6. Transition rules for cyclic interleaving and deadlock at termination

$$\frac{\langle x_1,\rho\rangle\downarrow,\ldots,\langle x_k,\rho\rangle\downarrow,\langle x_{k+1},\rho\rangle \xrightarrow{a} \langle x'_{k+1},\rho'\rangle}{\langle \| (\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha),\rho\rangle \xrightarrow{a} \langle \| (\alpha \frown \langle x'_{k+1}\rangle),\rho'\rangle} \quad (k \geq 0)$$

$$\frac{\langle x_1,\rho\rangle \not\rightarrow,\ldots,\langle x_k,\rho\rangle \not\rightarrow,\langle x_l,\rho\rangle\uparrow,\langle x_{k+1},\rho\rangle \xrightarrow{a} \langle x'_{k+1},\rho'\rangle}{\langle \| (\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha),\rho\rangle \xrightarrow{a} \langle \| (\alpha \frown \langle \mathsf{D}\rangle \frown \langle x'_{k+1}\rangle),\rho'\rangle} \quad (k \geq l > 0)$$

$$\frac{\langle x_1,\rho\rangle\downarrow,\ldots,\langle x_k,\rho\rangle\downarrow}{\langle \| (\langle x_1\rangle \frown \ldots \frown \langle x_k\rangle),\rho\rangle\downarrow} \qquad \frac{\langle x_1,\rho\rangle \not\rightarrow,\ldots,\langle x_k,\rho\rangle \not\rightarrow,\langle x_l,\rho\rangle\uparrow}{\langle \| (\langle x_1\rangle \frown \ldots \frown \langle x_k\rangle),\rho\rangle\uparrow} \quad (k \geq l > 0)$$

$$\frac{\langle x,\rho\rangle \xrightarrow{a} \langle x',\rho'\rangle}{\langle \mathsf{S_D}(x),\rho\rangle \xrightarrow{a} \langle \mathsf{S_D}(x'),\rho'\rangle} \qquad \frac{\langle x,\rho\rangle\downarrow}{\langle \mathsf{S_D}(x),\rho\rangle\uparrow} \qquad \frac{\langle x,\rho\rangle\uparrow}{\langle \mathsf{S_D}(x),\rho\rangle\uparrow}$$

stands for the set of all negative conditions $\neg(\langle x,\rho\rangle \xrightarrow{a} \langle p',\rho'\rangle)$ where p' is a closed term of BPPA, $\rho' \in \mathcal{E}$, $a \in \mathcal{A}_{\mathsf{tau}}$. Recall that $\mathcal{A} = \{f.m \mid f \in \mathcal{F}, m \in \mathcal{M}\}$.

Bisimulation equivalence is also a congruence with respect to the cyclic interleaving operator and the deadlock at termination operator. This follows immediately from the fact that the transition rules for the basic thread algebra with foci and methods constitute a complete transition system specification in relaxed panth format (see e.g. [8]).

4 Thread-Service Composition

In this section, we extend the basic thread algebra with foci and methods with thread-service composition. For each $f \in \mathcal{F}$, we introduce a *thread-service composition* operator $\_ /_f \_$. These operators have a thread as first argument and a service as second argument. $P /_f H$ is the thread that results from issuing all basic actions from thread P that are of the form $f.m$ to service H.

A service is represented by a function $H{:}\mathcal{M}^+ \to \{\mathsf{T},\mathsf{F},\mathsf{B},\mathsf{R}\}$ with the property that $H(\alpha) = \mathsf{B} \Rightarrow H(\alpha \frown \langle m\rangle) = \mathsf{B}$ and $H(\alpha) = \mathsf{R} \Rightarrow H(\alpha \frown \langle m\rangle) = \mathsf{R}$ for all $\alpha \in \mathcal{M}^+$ and $m \in \mathcal{M}$. This function is called the *reply* function of the service. Given a reply function H and a method m, the derived reply function of H after processing m, written $\frac{\partial}{\partial m}H$, is defined by $\frac{\partial}{\partial m}H(\alpha) = H(\langle m\rangle \frown \alpha)$.

The connection between a reply function H and the service represented by it can be understood as follows:

- If $H(\langle m\rangle) = \mathsf{T}$, the request to process command m is accepted by the service, the reply is positive and the service proceeds as $\frac{\partial}{\partial m}H$.
- If $H(\langle m\rangle) = \mathsf{F}$, the request to process command m is accepted by the service, the reply is negative and the service proceeds as $\frac{\partial}{\partial m}H$.
- If $H(\langle m\rangle) = \mathsf{B}$, the request to process command m is not refused by the service, but the processing of m is temporarily blocked. The request will have to wait until the processing of m is not blocked any longer.
- If $H(\langle m\rangle) = \mathsf{R}$, the request to process command m is refused by the service.

Table 7. Axioms for thread-service composition

$\mathsf{S} /_f H = \mathsf{S}$		TSC1
$\mathsf{D} /_f H = \mathsf{D}$		TSC2
$(\mathsf{tau} \circ x) /_f H = \mathsf{tau} \circ (x /_f H)$		TSC3
$(x \trianglelefteq g.m \trianglerighteq y) /_f H = (x /_f H) \trianglelefteq g.m \trianglerighteq (y /_f H)$	if $f \neq g$	TSC4
$(x \trianglelefteq f.m \trianglerighteq y) /_f H = \mathsf{tau} \circ (x /_f \frac{\partial}{\partial m} H)$	if $H(\langle m \rangle) = \mathsf{T}$	TSC5
$(x \trianglelefteq f.m \trianglerighteq y) /_f H = \mathsf{tau} \circ (y /_f \frac{\partial}{\partial m} H)$	if $H(\langle m \rangle) = \mathsf{F}$	TSC6
$(x \trianglelefteq f.m \trianglerighteq y) /_f H = \mathsf{D}$	if $H(\langle m \rangle) \in \{\mathsf{B}, \mathsf{R}\}$	TSC7

Table 8. Transition rules for thread-service composition

$$\frac{\langle x, \rho\rangle \xrightarrow{g.m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle \xrightarrow{g.m} \langle x' /_f H, \rho'\rangle}\ f \neq g \qquad \frac{\langle x, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x' /_f H, \rho'\rangle}$$

$$\frac{\langle x, \rho\rangle \xrightarrow{f.m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x' /_f \frac{\partial}{\partial m} H, \rho'\rangle}\ H(\langle m \rangle) \in \{\mathsf{T}, \mathsf{F}\},\ (f.m, H(\langle m \rangle)) \in \rho(\langle\,\rangle)$$

$$\frac{\langle x, \rho\rangle \xrightarrow{f.m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle\uparrow}\ H(\langle m \rangle) \in \{\mathsf{B}, \mathsf{R}\} \qquad \frac{\langle x, \rho\rangle\downarrow}{\langle x /_f H, \rho\rangle\downarrow} \qquad \frac{\langle x, \rho\rangle\uparrow}{\langle x /_f H, \rho\rangle\uparrow}$$

The axioms for thread-service composition are given in Table 7. In this table and all subsequent tables with axioms or transition rules in which g occurs, like f, g stands for an arbitrary focus from $\mathcal{F}$.

The structural operational semantics of the basic thread algebra with foci and methods extended with thread-service composition is described by the transition rules given in Tables 3, 6 and 8.

The action tau arises as the residue of processing commands. Therefore, tau is not connected to a particular focus, and is always accepted.

5 Guarding Tests

In this section, we extend the thread algebra developed so far with guarding tests. Guarding tests are basic actions meant to verify whether a service will accept the request to process a certain method now, and if not so whether it will be accepted after some time. Guarding tests allow for dealing with delayed processing and exception handling as will be shown in Section 6.

We extend the set of basic actions. For the set of basic actions, we now take the set $\{f.m, f?m, f??m \mid f \in \mathcal{F}, m \in \mathcal{M}\}$. Basic actions of the forms $f?m$ and $f??m$ will be called *guarding tests*. Performing a basic action $f?m$ is taken as making the request to the service named f to reply whether it will accept the request to process method m now. The reply is positive if the service will accept that request now, and otherwise it is negative. Performing a basic action

Table 9. Additional axioms for cyclic interleaving & deadlock at termination

$\parallel(\langle x \trianglelefteq f?m \trianglerighteq y\rangle \frown \alpha) = \parallel(\langle x\rangle \frown \alpha) \trianglelefteq f?m \trianglerighteq \parallel(\alpha \frown \langle y\rangle)$	CSI6
$\parallel(\langle x \trianglelefteq f??m \trianglerighteq y\rangle \frown \alpha) = \parallel(\langle x\rangle \frown \alpha) \trianglelefteq f??m \trianglerighteq \parallel(\alpha \frown \langle y\rangle)$	CSI7
$\mathsf{S_D}(x \trianglelefteq f?m \trianglerighteq y) = \mathsf{S_D}(x) \trianglelefteq f?m \trianglerighteq \mathsf{S_D}(y)$	S2D5
$\mathsf{S_D}(x \trianglelefteq f??m \trianglerighteq y) = \mathsf{S_D}(x) \trianglelefteq f??m \trianglerighteq \mathsf{S_D}(y)$	S2D6

Table 10. Additional transition rules for cyclic interleaving & deadlock at termination

$$\frac{\langle x_1,\rho\rangle\downarrow,\ldots,\langle x_k,\rho\rangle\downarrow,\langle x_{k+1},\rho\rangle \xrightarrow{\gamma} \langle x'_{k+1},\rho'\rangle}{\langle \parallel(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha),\rho\rangle \xrightarrow{\gamma} \langle \parallel(\langle x'_{k+1}\rangle \frown \alpha),\rho'\rangle} (\alpha,\mathsf{T}) \in \rho(\langle\,\rangle) \quad (k \geq 0)$$

$$\frac{\langle x_1,\rho\rangle\not\rightarrow,\ldots,\langle x_k,\rho\rangle\not\rightarrow,\langle x_l,\rho\rangle\uparrow,\langle x_{k+1},\rho\rangle \xrightarrow{\gamma} \langle x'_{k+1},\rho'\rangle}{\langle \parallel(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha),\rho\rangle \xrightarrow{\gamma} \langle \parallel(\langle x'_{k+1}\rangle \frown \alpha \frown \langle \mathsf{D}\rangle),\rho'\rangle} (\alpha,\mathsf{T}) \in \rho(\langle\,\rangle) \quad (k \geq l > 0)$$

$$\frac{\langle x_1,\rho\rangle\downarrow,\ldots,\langle x_k,\rho\rangle\downarrow,\langle x_{k+1},\rho\rangle \xrightarrow{\gamma} \langle x'_{k+1},\rho'\rangle}{\langle \parallel(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha),\rho\rangle \xrightarrow{\gamma} \langle \parallel(\alpha \frown \langle x'_{k+1}\rangle),\rho'\rangle} (\alpha,\mathsf{F}) \in \rho(\langle\,\rangle) \quad (k \geq 0)$$

$$\frac{\langle x_1,\rho\rangle\not\rightarrow,\ldots,\langle x_k,\rho\rangle\not\rightarrow,\langle x_l,\rho\rangle\uparrow,\langle x_{k+1},\rho\rangle \xrightarrow{\gamma} \langle x'_{k+1},\rho'\rangle}{\langle \parallel(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha),\rho\rangle \xrightarrow{\gamma} \langle \parallel(\alpha \frown \langle \mathsf{D}\rangle \frown \langle x'_{k+1}\rangle),\rho'\rangle} (\alpha,\mathsf{F}) \in \rho(\langle\,\rangle) \quad (k \geq l > 0)$$

$$\frac{\langle x,\rho\rangle \xrightarrow{\gamma} \langle x',\rho'\rangle}{\langle \mathsf{S_D}(x),\rho\rangle \xrightarrow{\gamma} \langle \mathsf{S_D}(x'),\rho'\rangle}$$

$f??m$ is taken as making the request to the service named f to reply whether it will accept the request to process method m now or after some time. The reply is positive if the service will accept that request now or after some time, and otherwise it is negative.

As explained below, it happens that not only thread-service composition but also cyclic interleaving has to be adapted to the presence of guarding tests.

The additional axioms for cyclic interleaving and deadlock at termination in the presence of guarding tests are given in Table 9. Axioms CSI6 and CSI7 state that:

- after a positive reply on $f?m$ or $f??m$, the same thread proceeds with its next basic action; and thus it is prevented that meanwhile other threads can cause a state change to a state in which the processing of m is blocked (and $f?m$ would not reply positively) or the processing of m is refused (and both $f?m$ and $f??m$ would not reply positively);
- after a negative reply on $f?m$ or $f??m$, the same thread does not proceed with it; and thus it is prevented that other threads cannot make progress.

Without this difference, the Simulation Lemma (Section 7) would not go through.

The additional transition rules for cyclic interleaving and deadlock at termination in the presence of guarding tests are given in Table 10, where γ stands for an arbitrary basic action from the set $\{f?m, f??m \mid f \in \mathcal{F}, m \in \mathcal{M}\}$.

Table 11. Additional axioms for thread-service composition

$(x \trianglelefteq g?m \trianglerighteq y) /_f H = (x /_f H) \trianglelefteq g?m \trianglerighteq (y /_f H)$	if $f \neq g$	TSC8
$(x \trianglelefteq f?m \trianglerighteq y) /_f H = \mathsf{tau} \circ (x /_f H)$	if $H(\langle m \rangle) \in \{\mathsf{T}, \mathsf{F}\}$	TSC9
$(x \trianglelefteq f?m \trianglerighteq y) /_f H = \mathsf{tau} \circ (y /_f H)$	if $H(\langle m \rangle) = \mathsf{B} \wedge f \neq \mathsf{t}$	TSC10
$(x \trianglelefteq f?m \trianglerighteq y) /_f H = \mathsf{D}$	if $(H(\langle m \rangle) = \mathsf{B} \wedge f = \mathsf{t}) \vee$ $H(\langle m \rangle) = \mathsf{R}$	TSC11
$(x \trianglelefteq g??m \trianglerighteq y) /_f H = (x /_f H) \trianglelefteq g??m \trianglerighteq (y /_f H)$	if $f \neq g$	TSC12
$(x \trianglelefteq f??m \trianglerighteq y) /_f H = \mathsf{tau} \circ (x /_f H)$	if $H(\langle m \rangle) \in \{\mathsf{T}, \mathsf{F}, \mathsf{B}\}$	TSC13
$(x \trianglelefteq f??m \trianglerighteq y) /_f H = \mathsf{tau} \circ (y /_f H)$	if $H(\langle m \rangle) = \mathsf{R}$	TSC14

Table 12. Additional transition rules for thread-service composition

$$\frac{\langle x, \rho\rangle \xrightarrow{f?m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x' /_f H, \rho'\rangle} \; H(\langle m\rangle) \in \{\mathsf{T}, \mathsf{F}\},\ (f?m, \mathsf{T}) \in \rho(\langle\,\rangle)$$

$$\frac{\langle x, \rho\rangle \xrightarrow{f?m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x' /_f H, \rho'\rangle} \; H(\langle m\rangle) = \mathsf{B},\ f \neq \mathsf{t},\ (f?m, \mathsf{F}) \in \rho(\langle\,\rangle)$$

$$\frac{\langle x, \rho\rangle \xrightarrow{\mathsf{t}?m} \langle x', \rho'\rangle}{\langle x /_{\mathsf{t}} H, \rho\rangle\!\uparrow} \; H(\langle m\rangle) = \mathsf{B} \qquad \frac{\langle x, \rho\rangle \xrightarrow{f?m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle\!\uparrow} \; H(\langle m\rangle) = \mathsf{R}$$

$$\frac{\langle x, \rho\rangle \xrightarrow{f??m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x' /_f H, \rho'\rangle} \; H(\langle m\rangle) \in \{\mathsf{T}, \mathsf{F}, \mathsf{B}\},\ (f?m, \mathsf{T}) \in \rho(\langle\,\rangle)$$

$$\frac{\langle x, \rho\rangle \xrightarrow{f??m} \langle x', \rho'\rangle}{\langle x /_f H, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x' /_f H, \rho'\rangle} \; H(\langle m\rangle) = \mathsf{R},\ (f?m, \mathsf{F}) \in \rho(\langle\,\rangle)$$

A service may be local to a single thread, local to a multi-thread, local to a host, or local to a network. A service local to a multi-thread is shared by all threads from which the multi-thread is composed, etc. Henceforth, to simplify matters, it is assumed that each thread, each multi-thread, each host, and each network has a unique local service. Moreover, it is assumed that $\mathsf{t}, \mathsf{p}, \mathsf{h}, \mathsf{n} \in \mathcal{F}$. Below, the foci t, p, h and n play a special role:

- for each thread, t is the focus of its unique local service;
- for each multi-thread, p is the focus of its unique local service;
- for each host, h is the focus of its unique local service;
- for each network, n is the focus of its unique local service.

The additional axioms for thread-service composition in the presence of guarding tests are given in Table 11. Axioms TSC10 and TSC11 are crucial. If $f = \mathsf{t}$, then f is the focus of the local service of the thread $x \trianglelefteq f?m \trianglerighteq y$. No other thread can raise a state of its local service in which the processing of m is blocked. Hence, if the processing of m is blocked, it is blocked forever.

The additional transition rules for thread-service composition in the presence of guarding tests are given in Table 12.

6 Delays and Exception Handling

We go on to show how guarding tests can used to express postconditional composition with delay and postconditional composition with exception handling.

For postconditional composition with delay, we extend the set of basic actions $\mathcal{A}$ with the set $\{f!m \mid f \in \mathcal{F}, m \in \mathcal{M}\}$. Performing a basic action $f!m$ is like performing $f.m$, but in case processing of the command m is temporarily blocked, it is automatically delayed until the blockade is over.

Postconditional composition with delay is defined by the equation given in Table 13. The equation from this table guarantees that $f.m$ is only performed if $f?m$ yields a positive reply.

Table 13. Defining equation for postconditional composition with delay

$x \trianglelefteq f!m \trianglerighteq y = (x \trianglelefteq f.m \trianglerighteq y) \trianglelefteq f?m \trianglerighteq (x \trianglelefteq f!m \trianglerighteq y)$

For postconditional composition with exception handling, we introduce the following notations: $x \trianglelefteq f.m\,[y] \trianglerighteq z$ and $x \trianglelefteq f!m\,[y] \trianglerighteq z$.

The intuition for $x \trianglelefteq f.m\,[y] \trianglerighteq z$ is that $x \trianglelefteq f.m \trianglerighteq z$ is tried, but y is done instead in the exceptional case that $x \trianglelefteq f.m \trianglerighteq z$ fails because the request to process m is refused. The intuition for $x \trianglelefteq f!m\,[y] \trianglerighteq z$ is that $x \trianglelefteq f!m \trianglerighteq z$ is tried, but y is done instead in the exceptional case that $x \trianglelefteq f!m \trianglerighteq z$ fails because the request to process m is refused. The processing of m may first be blocked and thereafter be refused; in that case, y is done instead as well.

The two forms of postconditional composition with exception handling are defined by the equations given in Table 14. The equations from this table guarantee that $f.m$ is only performed if $f?m$ yields a positive reply.

Table 14. Defining equations for postconditional composition with exception handling

$x \trianglelefteq f.m\,[y] \trianglerighteq z = (x \trianglelefteq f.m \trianglerighteq z) \trianglelefteq f??m \trianglerighteq y$
$x \trianglelefteq f!m\,[y] \trianglerighteq z = ((x \trianglelefteq f.m \trianglerighteq z) \trianglelefteq f?m \trianglerighteq (x \trianglelefteq f!m\,[y] \trianglerighteq z)) \trianglelefteq f??m \trianglerighteq y$

An alternative to the second equation from Table 14 is

$$x \trianglelefteq f!m\,[y] \trianglerighteq z = ((x \trianglelefteq f.m \trianglerighteq z) \trianglelefteq f?m \trianglerighteq (x \trianglelefteq f!m \trianglerighteq z)) \trianglelefteq f??m \trianglerighteq y\ .$$

In that case, y is only done if the processing of m is refused immediately.

7 A Formal Design Prototype

In this section, we show how the thread algebra developed so far can be used to give a simplified, formal representation schema of the design of systems that

consist of several multi-threaded programs on various hosts in different networks. We propose to use the term *formal design prototype* for such a schema. The presented schema can be useful in understanding certain aspects of the system designed.

The set of *basic thread expressions*, with typical element P, is defined by

$$\begin{aligned} P \;::=\; & \mathsf{D} \mid \mathsf{S} \mid P \trianglelefteq f.m \trianglerighteq P \mid P \trianglelefteq f!m \trianglerighteq P \mid \\ & P \trianglelefteq f.m\,[P] \trianglerighteq P \mid P \trianglelefteq f!m\,[P] \trianglerighteq P \mid X_E \;, \end{aligned}$$

where $f \in \mathcal{F}$, $m \in \mathcal{M}$ and X_E is a constant standing for the unique solution for variable X of a guarded system of recursion equations E.

A thread vector in which each thread has its local service is of the form

$$\langle P_1 \between_{\mathsf{t}} TLS\rangle \frown \ldots \frown \langle P_n \between_{\mathsf{t}} TLS\rangle \;,$$

where $P_1, \ldots, P_n$ are basic thread expressions and TLS is a local service for threads. TLS does nothing else but maintaining local data for a thread. A multi-thread vector in which each multi-thread has its local service is of the form

$$\langle \| \, (TV_1) \between_{\mathsf{p}} PLS\rangle \frown \ldots \frown \langle \| \, (TV_m) \between_{\mathsf{p}} PLS\rangle \;,$$

where $TV_1, \ldots, TV_m$ are thread vectors in which each thread has its local service and PLS is a local service for multi-threads. PLS maintains shared data of the threads from which a multi-thread is composed. A typical example of such data are Java pipes. A host behavior vector in which each host has its local service is of the form

$$\langle \| \, (PV_1) \between_{\mathsf{h}} HLS\rangle \frown \ldots \frown \langle \| \, (PV_l) \between_{\mathsf{h}} HLS\rangle \;,$$

where $PV_1, \ldots, PV_l$ are multi-thread vectors in which each multi-thread has its local service and HLS is a local service for hosts. HLS maintains shared data of the multi-threads on a host. A typical example of such data are the files connected with Unix sockets used for data transfer between multi-threads on the same host. A network behavior vector in which each network has its local service is of the form

$$\langle \| \, (HV_1) \between_{\mathsf{n}} NLS\rangle \frown \ldots \frown \langle \| \, (HV_k) \between_{\mathsf{n}} NLS\rangle \;,$$

where $HV_1, \ldots, HV_k$ are host behavior vectors in which each host has its local service and NLS is a local service for networks. NLS maintains shared data of the hosts in a network. A typical example of such data are the files connected with Unix sockets used for data transfer between different hosts in the same network.

The behavior of a system that consist of several multi-threaded programs on various hosts in different networks is described by an expression of the form $\| \, (NV)$, where NV is a network behavior vector in which each network has its local service. A typical example is the case where NV is an expression of the form

$$\begin{aligned} & \| \, (\langle \| \, (\langle \| \, (\langle P_1 \between_{\mathsf{t}} TLS\rangle \frown \langle P_2 \between_{\mathsf{t}} TLS\rangle) \between_{\mathsf{p}} PLS\rangle \frown \\ & \qquad \langle \| \, (\langle P_3 \between_{\mathsf{t}} TLS\rangle \frown \langle P_4 \between_{\mathsf{t}} TLS\rangle \frown \langle P_5 \between_{\mathsf{t}} TLS\rangle) \between_{\mathsf{p}} PLS\rangle) \between_{\mathsf{h}} HLS\rangle \frown \\ & \quad \langle \| \, (\langle \| \, (\langle P_6 \between_{\mathsf{t}} TLS\rangle) \between_{\mathsf{p}} PLS\rangle) \between_{\mathsf{h}} HLS\rangle) \between_{\mathsf{n}} NLS \;, \end{aligned}$$

Table 15. Definition of simulation relation

$\mathsf{S}\ \mathsf{sim}\ x$
$\mathsf{D}\ \mathsf{sim}\ x$
$x\ \mathsf{sim}\ y \wedge x\ \mathsf{sim}\ z \Rightarrow x\ \mathsf{sim}\ y \trianglelefteq a \trianglerighteq z$
$x\ \mathsf{sim}\ y \wedge z\ \mathsf{sim}\ w \Rightarrow x \trianglelefteq a \trianglerighteq z\ \mathsf{sim}\ y \trianglelefteq a \trianglerighteq w$

where $P_1, \ldots, P_6$ are basic thread expressions, and TLS, PLS, HLS and NLS are local services for threads, multi-threads, hosts and networks, respectively. It describes a system that consists of two hosts in one network, where on the first host currently a multi-thread with two threads and a multi-thread with three threads exist concurrently, and on the second host currently a single multi-thread with a single thread exists.

Evidence of correctness of the schema $\parallel(NV)$ is obtained by Lemma 1 given below. This lemma is phrased in terms of a simulation relation sim on the closed terms of the thread algebra developed in the preceding sections. The relation sim (is simulated by) is defined inductively by means of the rules in Table 15.

Lemma 1 (Simulation Lemma). *Let P be a basic thread expression in which all basic actions are from the set $\{f.m \mid f \in \mathcal{F} \setminus \{\mathsf{t}, \mathsf{p}, \mathsf{h}, \mathsf{n}\}, m \in \mathcal{M}\}$ and constants standing for the solutions of guarded systems of recursion equations do not occur. Let $C[P]$ be a context of P of the form $\parallel(NV)$ where NV is a network behavior vector as above. Then $P\ \mathsf{sim}\ C[P]$. This implies that $C[P]$ will perform all steps of P in finite time.*

Proof. First we prove $P\ \mathsf{sim}\ C'[P]$, where C' is a context of P of the form $\parallel(TV)$, by induction on the depth of P, and in both the basis and the inductive step, by induction on the position of P in thread vector TV. Using in each case the preceding result, we prove an analogous result for each higher-level vector in a similar way.

8 Conclusions

We have presented an algebraic theory of threads and multi-threads based on multi-level strategic interleaving for the simple strategy of cyclic interleaving. The other interleaving strategies treated in [4] can be adapted to the setting of multi-level strategic interleaving in a similar way. We have also presented a reasonable though simplified formal representation schema of the design of systems that consist of several multi-threaded programs on various hosts in different networks. By dealing with delays and exceptions, this schema is sufficiently expressive to formalize mechanisms like Java pipes (for communication between threads) and Unix sockets (for communication between multi-threads, called processes in Unix jargon, and communication between hosts). The exception handling notation introduced is only used for single threads and a translation takes care of its meaning.

References

1. Arnold, K., Gosling, J.: The Java Programming Language. Addison-Wesley (1996)
2. Bishop, J., Horspool, N.: C# Concisely. Addison-Wesley (2004)
3. Bergstra, J.A., Klop, J.W.: Process algebra for synchronous communication. Information and Control **60** (1984) 109–137
4. Bergstra, J.A., Middelburg, C.A.: Thread algebra for strategic interleaving. Computer Science Report 04-35, Department of Mathematics and Computer Science, Eindhoven University of Technology (2004)
5. Bergstra, J.A., Loots, M.E.: Program algebra for sequential code. Journal of Logic and Algebraic Programming **51** (2002) 125–156
6. Bergstra, J.A., Bethke, I.: Polarized process algebra and program equivalence. In Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J., eds.: Proceedings 30th ICALP. Volume 2719 of Lecture Notes in Computer Science., Springer-Verlag (2003) 1–21
7. Aceto, L., Fokkink, W.J., Verhoef, C.: Structural operational semantics. In Bergstra, J.A., Ponse, A., Smolka, S.A., eds.: Handbook of Process Algebra. Elsevier, Amsterdam (2001) 197–292
8. Middelburg, C.A.: An alternative formulation of operational conservativity with binding terms. Journal of Logic and Algebraic Programming **55** (2003) 1–19

Membrane Computing — Current Results and Future Problems*

Francesco Bernardini[1], Marian Gheorghe[1], Natalio Krasnogor[2], and German Terrazas[2]

[1] Department of Computer Science, University of Sheffield, UK
{F.Bernardini, M.Gheorghe}@dcs.shef.ac.uk
[2] ASAP Group, School of Computer Science and IT, University of Nottingham, UK
Natalio.Krasnogor@Nottingham.ac.uk
gzt@cs.nott.ac.uk

In the last decade and especially after Adleman's experiment [1] a number of computational paradigms, inspired or gleaned from biochemical phenomena, are becoming of growing interest building a wealth of models, called generically Molecular Computing. New advances in, on the one hand, molecular and theoretical biology, and on the other hand, mathematical and computational sciences promise to make it possible in the near future to have accurate systemic models of complex biological phenomena. Recent advances in cellular Biology led to new models, hierarchically organised, defining a new emergent research area called Cellular Computing.

P-systems represent a class of distributed and parallel computing devices of a biological type that was introduced in [14] which are included in the wider field of cellular computing. Several variants of this model have been investigated and the literature on the subject is now rapidly growing. The main results in this area show that P-systems are a very powerful and efficient computational model [15], [16], [13]. There are variants that might be classified according to different criteria. They may be regarded as language generators or acceptors, working with strings or multisets, developing synchronous or asynchronous computation. Two main classes of P-systems can be identified in the area of membrane computing [15]: *cell-like P-systems* and *tissue-like P-systems*. The former type is inspired by the internal organization of living cells with different compartments and membranes hierarchically arranged; formally this structure is associated with a tree. Tissue P-systems have been motivated by the structure and behaviour of multicellular organisms where they form a multitude of different tissues performing various functions [2]; the structure of the system is instead represented as a graph where nodes are associated with the cells which are allowed to communicate alongside the edges of the graph.

More recently, a notion of population P-systems has been introduced [3], [4] as a model for tissue P-systems where the structure of the underlying graph can be modified during a computation by varying the set of nodes and the

* We are indebted to the anonymous referees for their comments that allowed us improving the readability of the text.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 49–53, 2005.

set of edges in the graph. Specifically, nodes are associated with cells, each of them representing a basic functional unit of the system, and edges model bonds among these cells that are dynamically created and destroyed. Although mainly inspired by the cell behaviour in living tissues, population P-systems may be also regarded as an abstraction of a population of bio-entities aggregated together in a more complex bio-unit (e.g. social insects like ants, bees, wasps etc, organized in colonies or bacteria of different types). This is the main reason why we use the term population instead of tissue albeit the term cell is retained to denoting an individual in the system. The concept also recalls other similar computational models: grammar systems [8], eco-grammar systems [9], or more recently, networks of parallel/evolutionary processors [10].

Universality results have been obtained [4] for a number of variants of population P-systems. The following different rules are considered: transformation rules for modifying the objects that are present inside the cells, communication rules for moving objects from a cell to another one, cell division rules for introducing new cells in the system, cell differentiation rules for changing the types of the cells, and cell death rules for removing cells from the system. As well as this, bond making rules are considered that are used to modify the links between the existing cells (i.e., the set of edges in the graph) at the end of each step of evolution performed by means of the aforementioned rules. In other words, a population P-system in [4] is basically defined as an evolution-communication P-system [7] but with the important difference that the structure of the system is not rigid and it is represented as an arbitrary graph. In particular, bond making rules are able to influence cell capability of moving objects from a place to another one by varying the set of edges in the underlying graph.

Another interesting variant of population P-systems is obtained by considering the general mechanism of cell communication based on signal molecules as a mechanism for triggering particular transformations inside of a cell once a particular signal-object has been received from some other cell in the system [3]. This leads to a notion of population P-systems where the sets of rules associated with the cell can vary according to the presence of particular objects inside and outside the cells. Yet again, the introduction of this mechanism is motivated by the features shared by biological systems at various levels where the behaviour of an individual is affected both by its internal state and by the external stimuli received. Some results concerning the power of population P-systems with a rule activating mechanism have been obtained [5].

Further developments of the area of population P-systems are expected to cover alternative ways of defining the result of a computation and the use of string objects. Population P-systems in fact attempt to model aspects of biological systems formed by many different individual components cooperating in a coherent way for the benefit of the system as a whole; a more appropriate notion of computation is therefore necessary in order to characterise the emergent behaviour of the system. Existing approaches in the area of grammar system such parallel communicating grammar systems [8] or eco-grammar systems [9], rely on the use of a single sentential form that is rewritten in parallel by differ-

ent interacting/cooperating grammar components. In particular, in the case of eco-grammar systems, this sentential form is associated with the environment and it can be rewritten both by rules corresponding to action taken from the individual components in the system and by dedicated rules associated with the environment. In a similar way, we can consider string-processing population P-systems where the result of a computation is given by a string (or a language) produced in the environment at the end of a computation. However, with respect to grammar systems, population P-systems present some other interesting features like the possibility of moving objects from a place to another one, the possibility of forming bonds among the cells, the possibility of introducing new cells in the system by means of cell division, which need to be formalised for the particular case of string objects. In this respect, we aim to present some reasonable variants of population P-systems with string objects.

Apart from being a very interesting research area in theoretical computer science P-systems have been used in modelling different biological systems. One of the most exciting biological system is represented by the quorum sensing phenomenon occurring in bacteria.

Recent advances in analytical biotechnology, computational biology, bioinformatics and computational modeling promise ever deeper understanding of the complexity of biological systems, particularly the computations they perform in order to survive in dynamic and hostile environments. These insights will ultimately enable researchers to harness the living cell as a computational device with its own sensors, internal states, transition functions, actuators, etc, and to program them as "nano-bots" for particular tasks such as targeted drug delivery, chemical factories, nano-structures repairs, bio-film scaffolding and self-assembling, to name but a few.

Quorum sensing (QS) have been described as "the most consequential molecular microbiology story of the last decade" [20, 6]. It relies on the activation of a sensor kinase or response regulator protein by a diffusible, low molecular weight, signal molecule (a "pheromone" or "autoinducer") [18]. In QS, the concentration of the signal molecule reflects the number of bacterial cells in a particular niche and perception of a threshold concentration of that signal molecule indicates that the population is "quorated" i.e. ready to make a behavioral decision [19].

An overview on Quorum Sensing in *P. aeruginosa* with comments on some of the techniques that have been used to model this phenomenom as well as a more "computationally flavoured" approach for QS and some research tracks which could benefit from an in-depth understanding of QS are presented in [11]

This perspective on modelling biological systems at the level mentioned before is investigated by describing various bio-components as agents. An agent is a fairly complex computer system that is situated in some environment and is capable of flexible, autonomous actions in order to meet its design objectives [12]. The extreme complexity of agent systems is due to substantial differences between the attributes of their components, high computational power required by the processes running within these components, huge volume of data manipulated by these processes and finally possibly extensive amount of communication

in order to achieve coordination and collaboration. The use of a computational framework that is capable of modelling both the dynamic aspects (i.e. the continuous change of agents' states together with their communication) and the static aspects (i.e. the amount of knowledge and information available), will facilitate modelling and simulation of such complex systems.

Acknowledgments

The research of FB and MG has been supported by the Engineering and Physical Science Research Council (EPSRC) of United Kingdom, Grant GR/R84221/01. NK acknowledges the EPSRC (GR/T07534/01) and BBSRC (BB/C511764/1) for (partially) funding his research in bioinformatics.

References

1. Adleman, L.M. 1994. Molecular computation of solutions to combinatorial problems. Science, 226, 1021-1024
2. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P. 2002. The Molecular Biology of The Cell. Fourth Edition. Garland Publ. Inc., London
3. Bernardini, F., Gheorghe, M. 2004. Cell Communication in Tissue P-systems and Cell Division in Population P-Systems. In [17], 74-91
4. Bernardini, F., Gheorghe, M. 2004. Population P-Systems. Journal of Universal Computer Science, 10, 509-539
5. Bernardini, F., Gheorghe, M. 2005. Cell Communication in Tissue P-Systems: Universality Results. Soft Computing (to appear)
6. Busby, S., de Lorenzo, V. 2001. Cell regulation - putting together pieces of the big puzzle. Curr. Op. Microbiol, 4, 117-118.
7. Cavaliere, M. 2003. Evolution Communication P-Systems. In [16], 134-145 and in [17], 206-223.
8. Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, Gh. 1997. Grammar Systems. A Grammatical Approach to Distribution and Cooperation. Gordon and Breach, London
9. Csuhaj-Varjú, E., Kelemen, J., Kelemenova, A., Păun, Gh. 1997. Eco-Grammar Systems: A Grammatical Framework for Studying Life-Like Interactions. Artificial Life, 3, 1-28.
10. Csuhaj-Varjú, E., Salomaa, A., 1997. Networks of Parallel Language Processors. In New Trends in Formal Languages. Control, Cooperation, and Combinatorics. In Păun, Gh., Salomaa, A. (eds), Lecture Notes in Computer Science, 1218, Springer-Verlag, Berlin, Heidelberg, New York, 299-318
11. Krasnogor, N., Gheorghe, M., Terrazas, G., Diggle, S., Williams, P., Camara, M. 2005. Bulletin of the EATCS (to appear)
12. Jennings, N.R. 2000. On agent-based software engineering. Artificial Intelligence, 117, 277-296.
13. Martin-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G., Salomaa, A. (eds) 2004. Membrane computing. International workshop, WMC 2003, Tarragona, Spain, July 2003. Revised papers. Lecture Notes in Computer Science, 2933, Springer, Berlin Heidelberg New York

14. Păun, Gh. 2000. Computing with Membranes. Journal of Computer and System Sciences, 61, 108-143
15. Păun, Gh. 2002. Membrane computing. An introduction. Springer, Berlin Heidelberg New York
16. Păun, Gh., Rozenberg, G., Salomaa, A., Zandron, C. (eds) 2003. Membrane computing. International workshop, WMC-CdeA 02, Curtea de Arges, Romania, August 19-23, 2002. Revised papers. Lecture Notes in Computer Science, 2597, Springer, Berlin Heidelberg New York
17. Păun, Gh., Riscos-Nunez, A., Romero-Jimenez, A., Sancho-Caparrini, F. (eds) 2004. Second brainstorming week on membrane computing, Seville, 2-7 February 2004. Technical Report 01/2004, Department of Computer Science and Artificial Intelligence, University of Seville, Spain
18. Swift, S., Downie, J.A., Whitehead, N.A., Barnard, A.M.L., Salmond, G.P.C., Williams, P. 2001. Quorum sensing as a population-density-dependent determinant of bacterial physiology. Adv Micro Physiol, 45, 199-270.
19. Williams, P., Camara, M., Hardman, A., Swift, S., Milton, D., Hope, V.J., Winzer, K., Middleton, B., Pritchard, D.I., Bycroft, B.W. 2000. Quorum sensing and the population dependent control of virulence. Phil. Trans Roy Soc London B, 355(1397), 667-680.
20. Winzer, K., Hardie, K.H., Williams, P. 2002. Bacterial cell-to-cell communication: sorry can't talk now – gone to lunch!. Curr Op. Microbiol, 5, 216-222.

How to Compare the Power of Computational Models*

Udi Boker and Nachum Dershowitz

School of Computer Science, Tel Aviv University,
Ramat Aviv, Tel Aviv 69978, Israel
udiboker@tau.ac.il
nachum.dershowitz@cs.tau.ac.il

Abstract. We argue that there is currently no satisfactory general framework for comparing the extensional computational power of arbitrary computational models operating over arbitrary domains. We propose a conceptual framework for comparison, by linking computational models to hypothetical physical devices. Accordingly, we deduce a mathematical notion of relative computational power, allowing the comparison of arbitrary models over arbitrary domains. In addition, we claim that the method commonly used in the literature for "strictly more powerful" is problematic, as it allows for a model to be more powerful than itself. On the positive side, we prove that Turing machines and the recursive functions are "complete" models, in the sense that they are not susceptible to this anomaly, justifying the standard means of showing that a model is "hypercomputational."

1 Introduction

Our goal is to formalize comparisons of computational models, that is, the determination when one set of partial functions is computationally more powerful than another set. We seek a robust definition of relative power, one that does not depend itself on any notion of computability. It should allow one to compare arbitrary models over arbitrary domains in some quasi-ordering that captures the intuitive concept of computational strength. Such a comparison notion (or notions) should also allow one to prove statements like "analogue machines are strictly more powerful than digital devices," even though the two models operate over domains of different cardinalities.

With a satisfactory comparison notion in place, we look into mathematical relations between computational models, and properties they confer on models. We call a model that is not as powerful as any of its proper expansions "complete." We investigate completeness, and check whether some classical models enjoy this property.

* This work was carried out in partial fulfillment of the requirements for the Ph.D. degree of the first author.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 54–64, 2005.

Extensionality. We are only interested in the computational aspect of computational models (extensionality), that is, which problems can be solved by a model, regardless of the solution's complexity or the model's mechanisms. Hence, a computational model is represented simply by a set of (partial) functions (or multivalued functions) over the domain of its operation.

The Problem. Though model comparison is a common practice in the literature, it is usually done without a formal comparison notion and without justification for the chosen method. To the best of our knowledge, there is currently no satisfactory general means for comparing arbitrary computational models operating over arbitrary domains. A notion is lacking via which one could show, for example, that analogue computers are strictly more powerful than Turing machines, as well as show that finite automata are more powerful than some weak analogue model. In Section 4, we list some of the familiar comparison methods and discuss their ramifications.

The Framework. In Section 2, we propose a general, philosophical, definition of a computational model and of relative computational power. We understand a computational model to be a mathematical modeling and idealization of some hypothetical physical device, from a specific point of view of the world. A model B is at least as powerful as A if it has the potential to do whatever A does, under any possible view of the world. Accordingly, we provide, in Section 3, a method (Definition 3) for comparing arbitrary models over arbitrary domains.

Completeness. In Section 5, we show that the method usually used in the literature for "more powerful" ($\succsim$) is mathematically problematic, as it allows for a model to be more powerful than itself ($A \succsim A$). We define a model that is not as powerful as any of its proper expansions to be *complete*. The standard method of comparison is suitable only for such complete models. On the positive side, we prove in Section 6.1 that Turing machines and the recursive functions are complete with respect to the desired comparison notions.

Computability. In Section 6, we show that some of the models known to be of equivalent power to Turing machines (the recursive functions, random access machines and counter machines) are indeed so by our suggested general notion.

Hypercomputation. In Section 6.1, we prove that Turing machines and the recursive functions are complete models. Accordingly, we provide a simpler comparison notion for showing that a model is hypercomputational. This notion provides a justification for the (otherwise improper) comparison method used in the literature for showing that a model is hypercomputational.

Note. We use the Z-standard [1] for function arrows. For example, $\mapsto$ denotes a partial function, $\twoheadrightarrow$ is used for a total surjective function, and $\rightarrowtail$ is an injection. We use double-arrows for mappings (multi-valued functions). So $\rightrightarrows$ denotes a total surjective mapping.

Proofs are omitted for lack of space.

2 The Conceptual Framework

We first propose a general, philosophical, definition of a computational model, and—in Section 2.2—of relative computational power. In Section 3, we will formalize these definitions for comparing arbitrary models over arbitrary domains.

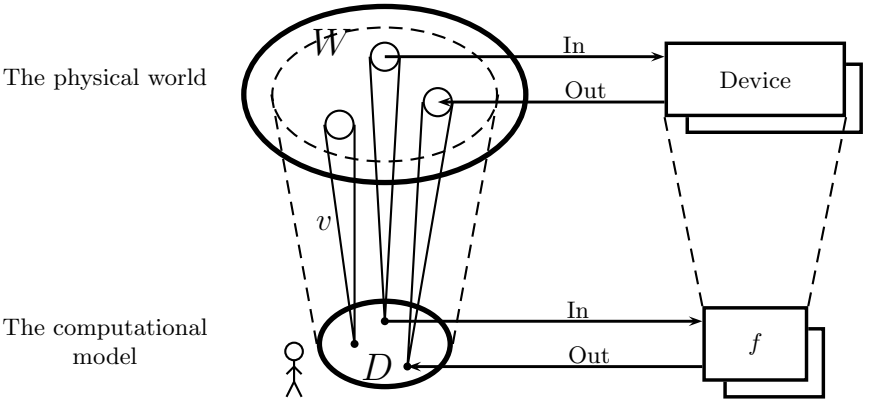


Fig. 1. A computational model is a mathematical modeling of some hypothetical physical devices, from a specific point of view of the world

2.1 What Is a Computational Model?

We can think of a computational model as a mathematical modeling and idealization of some hypothetical physical device, from a specific point of view of the world (see Fig. 1).

- A physical device gets a physical input and returns a physical output. For example, an electric device may take some electric voltage at two of its pins as input, and return a voltage at two other pins as output.
- A corresponding computational model takes a specific point of view of the physical world. For example, a model of a digital computer might view a voltage lower than 0.5v as the binary value $\mathsf{0}$ and of 0.5v or higher as $\mathsf{1}$. That is, the domain of the model, D, is a "view" of the physical world, W. This view is a partial surjective function $v : W \twoheadrightarrow D$.
- The device computes a function on world entities (in our example above, $\xi : \mathbf{R} \to \mathbf{R}$), while from the model's point of view it computes a function on its domain (in our example, $f : \{\mathsf{0}, \mathsf{1}\} \to \{\mathsf{0}, \mathsf{1}\}$).

A computational model, by itself, can be viewed as a "black box," computing a set of partial functions. The domain and range of functions are identical, except that the range is extended with $\perp$, representing "undefined."

The modeling of a hypothetical device from a specific point of view of the world will be at the heart of our method of comparing different models. The world

can be chosen to be any set of cardinality at least as large as the cardinality of the model's domain.

The idea that a model encapsulates a point of view of the world is shared by Minsky [2]:

> We use the term "model" in the following sense: To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A. The model relation is inherently ternary. ... It is understood that B's use of a model entails the use of encodings for input and output, both for A and for A*. If A is the world, questions for A are experiments.

Different Domain and Range. There are models with different domain and range, e.g. numeral input and boolean output. A generalized view is to consider the "actual" model's domain to be the union of the original domain and range.

Uniform Computation. It is common to have models with functions of any fixed arity, like the recursive functions, for example. We consider the "actual" domain (and range) to be the set of all finite tuples of elements of the original domain. This is the view taken for Turing machines, in the BSS model [3–pp. 69–70], and implicitly with recursive functions when comparing them to Turing machines.

Computing over Structures. There are models defined over structures, that is, over sets together with "built-in" functions and relations. See, for example, [4, 5, 3]. We consider the structure's set as the domain, and include the structure's functions and relations in the model.

2.2 Comparing Computational Power

We generally say that a model B is at least as powerful as A, written $B \succsim A$, if it can do whatever A does. When both models have the same domain representation, it means "containment": B is at least as powerful as A if it computes all the functions that A does. The question is how one should compare models operating over different domains, as they compute formally-different functions.

We extend the above characterization as follows: B is at least as powerful as A if it has the potential to do whatever A does for every possible user (an abstract user, not necessarily human). In other words, for every view that A has of the world ($v : W \twoheadrightarrow \mathrm{dom}\ A$), there is a view by B of the world ($u : W \twoheadrightarrow \mathrm{dom}\ B$), such that B has the abstraction capabilities of A, and all the functionality of A from A's point of view (see Fig. 2, Definition 2, and Definition 3).

Assumption. We want to allow the world-domain W to be as big as required, as well as the resolution of its elements to be enlarged as much as required. That is, all elements $x \in W$ may be considered as sets of a fixed cardinality.

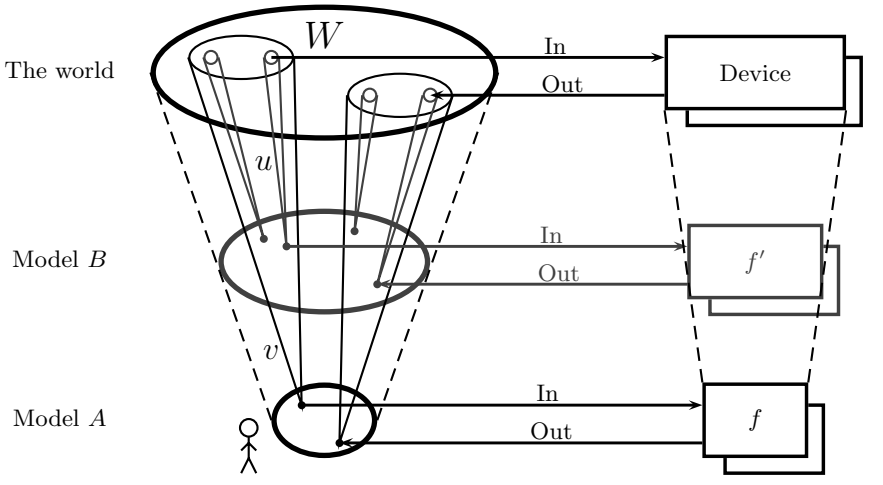


Fig. 2. The "stronger" model, B, should have the potential to provide all the functionality of the "weaker" model, A, from any user point of view

3 The Formal Comparison Notion

We need to formalize the conceptual framework of the previous section.

Definition 1 (Computational Model).

- *A* domain *is a nonempty set of elements.*
- *A* computational model *A over domain D is an object that computes a set of partial functions $f : D \mapsto D$, which may be interpreted as total functions $f : D \to D \cup \{\perp\}$.*
- *We write* dom *A for the domain over which model A operates.*
- *The* extensionality *of a model A, denoted* ext *A, is the set of partial functions that A computes.*
- *For models A and B, and a function f we shall write $f \in A$ as shorthand for $f \in$ ext A, and $A \subseteq B$ as short for* ext $A \subseteq$ ext B.
- *We say that a model B* properly expands *model A if $B \supsetneq A$.*

Some clarifications regarding function notations:

- A (partial) function $f : D \mapsto D'$ can be extended to images of subsets of D, $f : \mathscr{P}(D) \mapsto \mathscr{P}(D')$, in the standard fashion: $f(X) := \{f(x) : x \in X\}$.
- A total surjective mapping $\rho : D \twoheadrightarrow D'$ is a total function, $\rho : D \to \mathscr{P}(D')$, from D to the subsets of D', such that $\bigcup_{x \in D} \rho(x) = D'$.

Directly formalizing the conceptual characterization of "as powerful" (see Fig. 2), we get the following:

Definition 2 (Conceptual Power Comparison). *Model B is* at least as powerful as *model A if for every domain W (the world) and view $v : W \twoheadrightarrow$* dom *A, there are a view $u : W \twoheadrightarrow$* dom *B and abstraction-function $g \in B$, s.t.*

(a) for every function $f \in A$ there is a function $f' \in B$, s.t. $v \circ u^{-1} \circ f' \circ u \circ v^{-1}(x) = \{f(x)\}$ for all $x \in \text{dom } A$,
(b) $g(z) = g(y)$ iff $v \circ u^{-1}(z) = v \circ u^{-1}(y)$ for all $y, z \in \text{dom } B$, and
(c) $v \circ u^{-1} \circ g(y) = v \circ u^{-1}(y)$ for all $y \in \text{dom } B$.

Here "view" is a partial surjective function, and (by the conceptual assumption) all elements $x \in W$ may be considered as sets of a fixed cardinality. The first condition, (a), says that B computes every function of A, up to the mapping between the domains ($v \circ u^{-1}$). Condition (b) says that the function $g \in B$ distinguishes between the equivalence classes generated by the mapping, while (c) says that the distinction is made by choosing a representative element within each class.

Definition 2 may be simplified, omitting the world-domain.

Definition 3 (Power Comparison Notion).

1. *Model B is* (computationally) at least as powerful *as model A, denoted $B \succsim A$, if there are a total surjective mapping $\rho : \text{dom } B \twoheadrightarrow \text{dom } A$ and function $g \in B$, such that:*
   (a) for every function $f \in A$ there is a function $f' \in B$ such that $\rho \circ f' \circ \rho^{-1}(x) = \{f(x)\}$ for all $x \in \text{dom } A$,
   (b) $g(z) = g(y)$ iff $\rho(z) = \rho(y)$ for all $y, z \in \text{dom } B$, and
   (c) $\rho \circ g(y) = \rho(y)$ for all $y \in \text{dom } B$.
2. *Model B is* (computationally) more powerful *than A, denoted $B \succnsim A$, if $B \succsim A$ but $A \not\succsim B$.*
3. *Models A and B are* (computationally) equivalent *if $A \succsim B \succsim A$, in which case we write $A \approx B$.*

Proposition 4. *The computational power relation $\succsim$ between models is a quasi-order. Computational equivalence $\approx$ is an equivalence relation.*

Theorem 5. *Definitions 2 and 3.1 are equivalent. That is $B \succsim A$ by Definition 3.1 iff B is at least as powerful as A by Definition 2.*

Example 6. Consider a modeling of a simple electric-cable by a model EC, providing only the identity function over the reals. Then TM $\not\succsim EC$ and $EC \not\succsim$ TM.

Inclusion of the Identity Function. When the "weak" model includes the identity function ($\lambda x.x$), the general comparison notion may be simplified, replacing the surjective mapping (ρ) by a surjective function. If the "stronger" model is closed under functional composition, it may be further simplified, replacing the surjective function with an opposite injection ($\psi : \text{dom } A \rightarrowtail \text{dom } B$). This is similar to the embedding notion (Definition 9 below) with the additional requirement for an abstraction function (g). Comparison via a surjective function resembles the "representation" of [6–p. 33], just that here we insist on a total function.

Theorem 7. *Let A be a computational model with the identity function $(\lambda x.x \in A)$. Then a model B, closed under functional composition, is at least as powerful as A ($B \succsim A$) iff there exist an injection $\psi : \text{dom } A \rightarrowtail \text{dom } B$ and a total function $g \in B$ onto $\text{rng } \psi$ ($g : \text{dom } B \twoheadrightarrow \text{rng } \psi$), such that for every function $f \in A$ there is a function $f' \in B$ such that $\psi \circ f(x) = f' \circ \psi(x)$ for all $x \in \text{dom } A$.*

Example 8. Real recursive functions (Rrec) [7], are more powerful than Turing machines (TM). That is Rrec $\succsim$ TM. The comparison is done via the injection $\psi : \mathbf{N} \rightarrowtail \mathbf{R}$, where $\psi(n) = n$ [7–p. 18], and the floor function $(\lambda x.\, \lfloor x \rfloor)$ to provide the abstraction capabilities of Rec (the above function g) [7–p. 10].

4 Ramifications of Familiar Notions

Various methods have been used to compare the computational power of competing models.

Extended Domains. It is common to claim that a function is incorporated in any of its extensions. That is, a function $f : D \to D$ is incorporated in $f' : D' \to D'$ if $D \subseteq D'$ and $f = f' \restriction_D$. See, for example, [8–p. 654]: "Here we adopt the convention that a function on $\mathbf{N}$ is in an analog class $\mathcal{C}$ if some extension of it to $\mathbf{R}$ is, i.e. if there is some function $\widetilde{f} \in \mathcal{C}$ that matches f on inputs in $\mathbf{N}$."

By the conceptual framework, "B extends A" can be interpreted as "B having the potential to be at least as powerful as A for a user who has both domain views." For example, one can consider a user who views the world as real numbers, but can identify the natural numbers among them.

This approach is not appropriate as a general power comparison notion, since the extended model B doesn't necessarily have the abstraction capabilities of A. For example, a mathematician working with paper and pencil may consider various physical entities to "be" the symbol 'a' (e.g. a, a, a, a, a). A model that lacks the abstraction of the various 'a's, treating each of them totally differently, is not as powerful.

Embedding. Extending the domain is a special case of embedding. A model B embeds A, if there is an injection from the domain of A to the domain of B, via which B has all the functionality of A *over the range of the injection.*

Definition 9 (Embedding). *A computational model B* embeds *a model A, denoted $B \succsim_{\text{E}} A$, if there is an injection $\psi : \text{dom } A \rightarrowtail \text{dom } B$, s.t. for every function $f \in A$ there is $f' \in B$ such that $f' \circ \psi(x) = \psi \circ f(x)$ for all $x \in \text{dom } A$.*

For example, Turing machines and the (untyped) λ-calculus were shown by Church [9], Kleene [10], and Turing [11] to embed the partial recursive functions.

The reasons for the inadequacy of embedding as a generic power comparison notion are analogous to that of domain-extending.

Example 10. Let RE be the recursively enumerable predicates over $\mathbf{N}$. RE may embed an expansion with infinitely many non-r.e. partial predicates $\{h_i\}$. Let

$$h(n) = \begin{cases} 0 & \text{program } n \text{ halts uniformly} \\ 1 & \text{otherwise} \end{cases} \qquad h_i(n) = \begin{cases} 0 & n < i \vee h(n) = 0 \\ \perp & \text{otherwise .} \end{cases}$$

We have that $\mathrm{RE} \gtrsim_{\mathrm{E}} \mathrm{RE} \cup \{h_i\}$, by an injection $\psi(n) = 2n + h(n)$, as

$$h_i'(n) = \begin{cases} 0 & \lfloor n/2 \rfloor < i \text{ or } n \bmod 2 = 0 \\ \perp & \text{otherwise} \end{cases} \qquad f' = \begin{cases} f(\lfloor n/2 \rfloor) & f \in RE \\ h_i'(n) & f = h_i \text{ .} \end{cases}$$

where $f' \in \mathrm{RE}$ and $f' = \psi \circ f \circ \psi^{-1}$ for every $f \in \mathrm{RE} \cup \{h_i\}$. (Without loss of generality, we are supposing that $\psi(0) = h(0) = 0$.)

Effective Encoding. A common approach for comparing models over different domains is to require some manner of effectiveness of the encoding; see [12–p. 21] and [13–p. 290], for example. There are basically two approaches:

1. One can demand informal effectiveness: "The coding is chosen so that it is itself given by an informal algorithm in the unrestricted sense" [14–p. 27].
2. Or one can require encoding effectiveness via a specific model, say, Turing machines: "The Turing-machine characterization is especially convenient for this purpose. It requires only that the expressions of the wider classes be expressible as finite strings in a fixed finite alphabet of basic symbols" [14–p. 28].

By the conceptual framework, an "effective comparison" means that B is at least as powerful as A for a human user, assuming humans are capable of "effective" representations.

Effectivity is a useful notion; however, it is unsuitable as a general power comparison notion. The first, informal approach is too vague, while the second can add computational power when dealing with subrecursive models and is inappropriate when dealing with non-recursive models.

5 When Is a Model More Powerful?

In general, the *strict part* $\gtrsim^*$ of a quasi-order $\gtrsim^*$ is $\gtrsim^* \cap \not\lesssim^*$. That is, $B \gtrsim^* A$ if $B \gtrsim^* A$ but not $A \gtrsim^* B$.

The Common Method. Intuitively, one would expect that a proper expansion of a model (additional functions) is also more powerful, that is, for $B \supsetneq A$ to imply $B \gtrsim A$. For example, a model that computes more than Turing machines is considered more powerful (see, e.g., [15]). Hence, the common method of showing that a model B is more powerful than model A, for some comparison notion $\gtrsim^*$, is to show that $B \gtrsim^* C \supsetneq A$.

The Problem. Unfortunately, a proper expansion of a model is not necessarily more powerful. That is, $B \supsetneq A$ does not imply $B \succsim^* A$, where $\succsim^*$ may be embedding, our suggested notion, or "containment up to isomorphism" (Theorem 12).

Example 11. Define the set R_2 of "even" recursive functions (Rec):

$$R_2 = \left\{ \lambda n. \begin{cases} 2f(n/2) & n \text{ is even} \\ n & \text{otherwise} \end{cases} : f \in \text{Rec} \right\}$$

R_2 embeds all the recursive functions via the injection $\lambda n.2n$, though $R_2 \subsetneq \text{Rec}$.

See also Example 10, for the embedding of non-r.e. predicates in RE.

Note that the common comparison method (see above) permits a model to be more powerful than itself! For example, one might say that "$R_2 \succsim_{\text{E}} R_2$," since $R_2 \succsim_{\text{E}} \text{Rec} \supsetneq R_2$.

Theorem 12. *There are models isomorphic to proper expansions of themselves. That is, there is a set of functions M over a domain D, and a bijection $\pi : D \rightarrowtail D$, s.t. $\{\pi \circ f \circ \pi^{-1} : f \in M\} \supsetneq M$.*

The Solution. The general solution is to use the strict part of the quasi-order. For example, with embedding one should show that "B may embed A, while there is no injection via which A may embed B."

In addition, one can check whether a specific model is "complete" in the sense that it is not equivalent (with respect to the relevant notion) to any of its proper expansions. For complete models, the common (generally improper) method is suitable, saving the necessity of precluding all possible mappings.

Definition 13 (Complete Models). *Let $\succsim^*$ be a quasi-order (comparison notion). A computational model A is* complete*, with respect to $\succsim^*$, if $A \succsim^* B \supseteq A$ implies $A = B$ for all B.*

Proposition 14. *Let $\succsim^*$ be a quasi-order (comparison notion), and A a complete model w.r.t. $\succsim^*$. Then $B \succsim^* A$ iff there is a model C, such that $B \succsim^* C \supsetneq A$.*

Theorem 15. *Let A be a model with the identity function, closed under function composition, and complete w.r.t. to embedding ($\succsim_{\text{E}}$), then A is complete w.r.t. power-comparison ($\succsim$).*

Corollary 16. *Let A be a model with the identity function, closed under function composition, and complete w.r.t. to embedding. Then a model B is more powerful than A iff B is at least as powerful as A and embeds some proper expansion C of A. That is, $B \succsim A$ iff there is a model C s.t. $B \succsim A \subsetneq C \precsim_E B$.*

6 Computability

Some computational models considered to be of equivalent power to Turing machines are still so according to our suggested comparison notion (Definition 3).

Theorem 17. *Turing machines* (TM)*, the recursive functions* (Rec)*, counter machines* (CM)*, and random access machines* (RAM) *are all of the same computational power. That is,* $\mathrm{TM} \approx \mathrm{Rec} \approx \mathrm{CM} \approx \mathrm{RAM}$*.*

6.1 Hypercomputation

In Section 5, we saw that a proper expansion of a computational model is not necessarily more powerful (by any of the common comparison methods). What does this mean for hypercomputation? Can it be that Turing machines are as powerful as a model that computes additional functions?

We prove that Turing machines and the recursive functions are complete models, thus are not susceptible to such an anomaly. Accordingly, we provide some means to show that a model is hypercomputational.

Definition 18 (Hypercomputation). *A model* A *is* hypercomputational *if it is more powerful than Turing machines, that is, if* $A \succnsim \mathrm{TM}$*.*

Theorem 19. *The recursive functions* (Rec) *and the partial recursive functions* (PR) *are complete w.r.t. embedding.*

Theorem 20. *Turing machines* (TM) *are complete w.r.t. embedding.*

Corollary 21. *Model* A *is hypercomputational if any one of the following conditions is satisfied:*

1. $A \succnsim \mathrm{TM}$*.*
2. $A \supsetneq \mathrm{TM}$*.*
3. *There is a model* C*, such that* $A \succsim C \supsetneq \mathrm{TM}$*.*
4. *There is a model* C*, such that* $A \succsim_{\mathrm{E}} C \supsetneq \mathrm{TM}$ *and also* $A \succsim \mathrm{TM}$*.*

References

1. Bowen, J.P.: Glossary of Z notation. Information and Software Technology **37** (1995) 333–334 Available at: `http://staff.washington.edu/~jon/z/glossary.html`.
2. Minsky, M.L.: Matter, mind and models. Proc. International Federation of Information Processing Congress **1** (1965) 45–49 Available at `http://web.media.mit.edu/~minsky/papers/MatterMindModels.html`.
3. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer-Verlag, New York (1998)
4. Bournez, O., Cucker, F., de Naurois, P.J., Marion, J.Y.: Computability over an arbitrary structure. sequential and parallel polynomial time. FoSSaCS (2003) 185–199
5. Tucker, J.V., Zucker, J.I.: Abstract versus concrete computation on metric partial algebras. ACM Transactions on Computational Logic **5** (2004) 611–668

6. Weihrauch, K.: Computable Analysis — An introduction. Springer-Verlag, Berlin (2000)
7. Mycka, J., Costa, J.F.: Real recursive functions and their hierarchy. Journal of Complexity (2004) In print.
8. Campagnolo, M.L., Moore, C., Costa, J.F.: Iteration, inequalities, and differentiability in analog computers. Journal of Complexity **16** (2000) 642–660
9. Church, A.: An unsolvable problem of elementary number theory. American Journal of Mathematics **58** (1936) 345–363
10. Kleene, S.C.: Lambda-definability and recursiveness. Duke Mathematical Journal **2** (1936) 340–353
11. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society **42** (1936–37) 230–265
12. Engeler, E.: Formal Languages: Automata and Structures. Lectures in Advanced Mathematics. Markham Publishing Company, Chicago, IL (1968)
13. Hennie, F.: Introduction to Computability. Addison-Wesley, Reading, MA (1977)
14. Rogers, Jr., H.: Theory of Recursive Functions and Effective Computability. McGraw-Hill, New York (1966)
15. Siegelmann, H.T.: Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhäuser, Boston (1998)

Recombinant DNA, Gene Splicing as Generative Devices of Formal Languages*

Paola Bonizzoni[1], Clelia De Felice[2], and Giancarlo Mauri[1]

[1] Dipartimento di Informatica Sistemistica e Comunicazione,
Università degli Studi di Milano Bicocca,
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
{bonizzoni, mauri}@disco.unimib.it
[2] Dipartimento di Informatica ed Applicazioni,
Università di Salerno, 84081 Baronissi (SA), Italy
defelice@dia.unisa.it

Recombinant DNA (rDNA) is a general term which refers to the DNA resulting from the process of combining a piece of one DNA with another strand of DNA. More precisely recombinant DNA is obtained from two or more different sources that have been cleaved by restriction enzymes and joined by ligases: this biological mechanism is known as *gene splicing*. Recently, recombinant DNA technology has received a rapid increase in interest in Molecular Biology, aimed at the development of biotechnologies. On the other hand, the surprising power of this process are stimulating interest towards the design of computational models inspired by biological phenomena. In this perspective, formal language theory appears to be a natural framework for formalizing and investigating DNA computing models. In 1987 Tom Head pioneered a language-theoretic approach for studying splicing and recombinant DNA. He introduced the *splicing systems*, abstract models which are a formal counterpart of the DNA recombination under the action of restriction and ligase enzymes. A *splicing operation*, as a model of gene splicing, is introduced as an operator on strings. There are at least three different definitions of this operation, given by Head, Paun and Pixton respectively. A *splicing system* is defined by giving an initial language I (simulators of the initial set of DNA molecules) and a set of special words or rules R (simulating the enzymatic action). The set I is then transformed by repeated applications of the splicing operation. In Nature DNA occurs in both linear and circular form and circular splicing occurs in a recombinant mechanism (transposition) between bacteria and plasmids, that are circular DNA molecules. Depending on the DNA form and on the definition of the splicing operation we refer to, we have different definitions of *linear splicing systems* and of *circular splicing systems*.

Here, we take into account the definition of the splicing operation which is usually adopted, given by Paun. Furthermore, we focus on *finite splicing systems*

* Partially supported by MIUR Project *"Linguaggi Formali e Automi: Metodi, Modelli e Applicazioni"* (2003), by the contribution of EU Commission under The Fifth Framework Programme (*project MolCoNet* IST-2001-32008) and by 60% Project *"Linguaggi formali e codici: modelli e caratterizzazioni strutturali"* (University of Salerno, 2004).

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 65–67, 2005.

that are closest to the biological process. Indeed, they are defined by a finite set of rules and a finite set of initial strings. We discuss results concerning both circular and linear finite splicing systems.

Intensive studies on linear splicing systems and variants have been carried out, as part of a formal language framework, aimed at proving the universality of splicing systems or closure properties of splicing languages generated under different restrictions on I and/or R. In particular, finite splicing systems generate a proper subclass $\mathcal{H}$ of the class of regular languages. The characterization of $\mathcal{H}$ is still an open problem. We do not even know whether it is decidable if a regular language is in $\mathcal{H}$. We survey the partial known results on these two interesting problems and their relationships with classical notions in formal language theory.

Unlike the linear case, relatively few works on circular splicing systems and their languages have been published. Once again classical results in formal language theory and combinatorial tools have been of great use. Indeed, circular splicing systems deal with *circular languages* the elements of which are equivalence classes under the conjugacy relation (*circular words*). A main result about the power of circular splicing systems is due to Pixton who proved that a circular regular language and a finite set of rules generate a regular language if the set of rules satisfies some additional hypotheses. It is already known that in contrast with the linear case, the family of languages generated by finite circular splicing systems is not intermediate between two classes of (circular) languages in the Chomsky hierarchy. Indeed, regular circular languages exist which cannot be generated by any finite circular splicing system whereas context-free circular languages exist which are generated by such systems. As in the linear case, a characterization of regular circular languages which are generated by finite circular splicing is still lacking and, in addition, the computational power of these systems is still unknown. Once again, we survey the known results on these problems. A partial list of references follows below.

References

1. J. Berstel, D. Perrin (1985), *Theory of Codes*, Academic Press, New York.
2. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2004), Linear splicing and syntactic monoid, *submitted*.
3. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2003), Decision problems on linear and circular splicing, *in* "Proc. DLT 2002" (M. Ito, M. Toyama, eds.), Lecture Notes in Computer Science, Vol. 2450, pp. 78 − 92, Springer-Verlag.
4. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2003), Regular languages generated by reflexive finite linear splicing systems, *in* "Proc. DLT 2003" (Z. Èsik, Z. Fülöp, eds.), Lecture Notes in Computer Science, Vol. 2710, pp. 134−145, Springer-Verlag.
5. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2001), DNA and circular splicing, *in*: "Proc. DNA 2000" (A. Condon, G. Rozenberg, eds.), Lecture Notes in Computer Science, Vol. 2054, 117 − 129.
6. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2004), Circular splicing and regularity, *Theoretical Informatics and Appl.* 38, 189 − 228.

7. P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2004), On the power of circular splicing, *submitted*.
8. P. Bonizzoni, C. De Felice, R. Zizza (2004), The structure of reflexive regular splicing languages via Schützenberger constants, *Theoretical Computer Science*, to appear.
9. P. Bonizzoni, C. Ferretti, G. Mauri, R. Zizza (2001), Separating some splicing models, *Information Processing Letters* **76 (6)**, pp. 255 – 259.
10. P. Bonizzoni, G. Mauri (2004), Regular splicing languages and subclasses, *Theoretical Computer Science*, to appear.
11. K. Culik, T. Harju (1991), Splicing semigroups of dominoes and DNA, *Discrete Applied Math.* **31**, pp. 261 – 277.
12. I. Fagnot (2004), Splicing Systems and Chomsky hierarchy, *in* "Proc. of Journees Montoises 2004" (Lieges, Belgium).
13. E. Goode Laun (1999), *Constants and splicing systems*, PHD Thesis, Binghamton University.
14. E. Goode, D. Pixton (2004), Recognizing splicing languages: syntactic monoids and simultaneous pumping, *submitted* (available from http:// www.math.binghamton.edu/dennis/Papers/index.html).
15. T. Head (1987), Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours, *Bull. Math. Biol.* **49**, pp. 737 – 759.
16. T. Head (1998), Splicing languages generated with one sided context, *in* "Computing with bio-molecules: theory and experiments" (Gh. Paun, Ed.), Springer-Verlag.
17. T. Head, Gh. Paun, D. Pixton (1996), Language theory and molecular genetics: generative mechanisms suggested by DNA recombination, *in* "Handbook of Formal Languages" (G. Rozenberg, A. Salomaa, eds.), Vol. 2, pp. 295 – 360, Springer-Verlag.
18. M. A. Harrison (1978), *Introduction to Formal Language Theory*, Addison-Wesley, Reading, Mass.
19. J. E. Hopcroft, R. Motwani, J.D. Ullman (2001), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass.
20. S.M. Kim (1997), An algorithm for identifying spliced languages, *in* "Proc. of Cocoon 97", Lecture Notes in Computer Science **1276**, pp. 403 – 411.
21. S.M. Kim (1997), Computational modeling for genetic splicing systems, *SIAM Journal of Computing* **26**, pp. 1284 – 1309.
22. R. McNaughton, S. Papert (1971), *Counter-Free Automata*, MIT Press, Cambridge, Mass.
23. Gh. Paun (1996), On the splicing operation, *Discrete Applied Math.* **70**, pp. 57–79.
24. Gh. Paun, G. Rozenberg, A. Salomaa (1998), *DNA computing, New Computing Paradigms*, Springer-Verlag.
25. D. Perrin (1990), Finite Automata, *in* "Handbook of Theoretical Computer Science" (J. Van Leeuwen, Ed.), Vol. B, pp. 1 – 57, Elsevier.
26. D. Pixton (1996), Linear and circular splicing systems, *in* "Proc. of 1st Int. Symp. on Intelligence in Neural and Biological Systems", IEEE Computer Society Press, pp. 181 – 188, Silver Spring.
27. D. Pixton (1996), Regularity of splicing languages, *Discrete Applied Math.* **69**, pp. 101 – 124.
28. M. P. Schützenberger (1975), Sur certaines opérations de fermeture dans les langages rationnels, *Symposia Mathematica* **15**, pp. 245 – 253.
29. S. Verlan, R. Zizza (2003), 1–splicing vs. 2–splicing: separating results, *in* "Proc. of Words 03" (Turku, Finland, 2003), pp. 320 – 331.

Quantum Computing

Harry Buhrman

Centrum voor Wiskunde en Informatica,
Kruislaan 413,
1090 GB Amsterdam, The Netherlands
and
Institute for Logic, Language and Computation,
Plantage Muidergracht 24,
1018 TV Amsterdam, The Netherlands

Many computational problems that turn up in industry, operations research, network design, artificial intelligence, simulation of physical systems, logic, number theory, combinatorics, algebra, and computational biology lack a fast or feasible algorithmic solution. The best known algorithms for these problems are horrendously slow. One of the central open problems in computer science is the question of whether this slowness is inherent in these problems or that we simply lack good programming techniques. This question is known as the **P** versus **NP** question. The hardest computational problems of the above type are called **NP**-complete problems. It is widely believed that there does not exist a feasible algorithmic solution for these **NP**-complete problems.

Quantum computing devices are computing devices that take advantage of the laws of quantum mechanics, whereas classical computers only use classical mechanics. Quantum computers can outperform our current, classical, technology of computing, due to the possibility of massive exponential parallelism (the superposition principle) and interference.

Quantum computing gained a lot of momentum after the breakthrough result of Peter Shor who demonstrated that the factoring problem can be efficiently solved on a quantum computer, whereas no classical algorithm is known that solves this problem quickly. His results give some evidence that quantum computers can solve certain computational problems more efficiently than classical computers. Since most of modern cryptography is based on our inability to efficiently factor large numbers, Shor's algorithm breaks all these cryptographic protocols.

In this course, we will introduce the mathematical framework of quantum mechanics and show how it can be used to define a quantum computer. We will then treat some of the known quantum algorithms including Shor's factorization algorithm, entanglement and the Einstein-Podolsky-Rozen paradox, quantum communication complexity, and quantum cryptography. If time permits we will mention some of the recent developments.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, p. 68, 2005.

Symbol Grounding in Connectionist and Adaptive Agent Models

Angelo Cangelosi

Adaptive Behaviour and Cognition Research Group,
School of Computing, Communications and Electronics,
University of Plymouth, UK
acangelosi@plymouth.ac.uk

Abstract. This paper presents the Cognitive Symbol Grounding framework for modeling language in neural networks and adaptive agent simulations. This approach is characterized by the hypothesis that symbols are directly grounded into the agents' own categorical representations, whilst at the same time having syntactic relationships with other symbols. The mechanism of grounding transfer is also introduced. This is the process by which the grounding of basic words, acquired via direct sensorimotor experience, is transferred to higher-order words via linguistic descriptions. Various simulations are briefly reviewed to demonstrate the use of the Cognitive Symbol Grounding approach.

1 The Cognitive Symbol Grounding Framework

The grounding of symbols in computational cognitive systems requires that the simulated cognitive agent is able to access the meaning of its symbols (words) directly, without the intervention of an external viewer such as the experimenter. This has been a significant shortcoming of cognitive models only based on symbolic architecture. Such a limit is commonly referred to as the Symbol Grounding Problem [6].

Recent cognitive models based on connectionist agents and robots use the Cognitive Symbol Grounding framework to intrinsically link symbols to the agents' own cognitive system [1]. This approach is characterized by the hypothesis that symbols are directly grounded into the agents' own categorical representations, whilst at the same time having logical/syntactic relationships with other symbols. First, each symbol is directly grounded into internal categorical representations. These representations include perceptual categories (e.g. the concept of blue color, square shape, and male face), sensorimotor categories (the action concept of grasping, pushing, carrying), social representations (individuals, groups and relationships) and other categorizations of the agent's own internal states (emotions, motivations). Secondly, these categories are connected to the external world through our perceptual, motor and cognitive interaction with the environment.

Two main modeling approaches to the symbol grounding are presented here: (1) the connectionist approach, based on artificial neural network simulations of

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 69–74, 2005.

categorization and linguistic tasks, and (2) the embodied agent modeling method based on multi-agent simulations and robotic studies. Both approaches share an integrative view of the cognitive systems, in which vision, action and language are intrinsically linked with one another. This permits the design of cognitive systems where language is directly grounded in the agent's own sensorimotor and cognitive abilities.

The use of an integrated language/cognition/action system is particularly important for the mechanism of grounding transfer. This is the process by which the grounding of basic words, acquired via direct sensorimotor experience, is transferred to higher-order words via linguistic descriptions (i.e. indirect grounding). For example, I can learn by direct experience that the word `horse` is grounded on my sensorimotor experience of seeing (and/or riding) a horses and that the word `horn` is grounded on the vision of the horn of an animal. Subsequently, via the linguistic description "`The unicorn is an animal similar to a horse with a horn`" I can learn the new word, and concept, `unicorn` and indirectly ground it in my experience of horses and horns.

The following sections will briefly review some modeling work on the Cognitive Symbol Grounding hypothesis developed by the author and collaborators. These examples will demonstrate the use of connectionist and adaptive agent models for the direct grounding of symbols and the transfer of grounding from low level words to higher-level symbols.

2 Connectionist Simulations

The connectionist approach to symbol grounding is based on simulations of artificial neural networks for category learning and naming tasks. In particular, this has been possible through the use of dual-route neural networks architectures [8] that permit the link (a) between perceptual and sensorimotor representations and (b) between these sensorimotor representations and symbolic knowledge. Typically, a neural network will have visual and linguistic input units indirectly link, via hidden units, to motor and linguistic output units. The process of language understanding can be simulated with the link from linguistic input to motor outputs, while language production links visual input to linguistic output units.

A seminal paper on categorization and symbol grounding with neural networks is that proposed by Harnad et al. [7]. This specifically focused on grounding symbols in categorical perception. The authors used a multi-layer perceptron to categorizing lines according to their length. Training consisted of two sequential backpropagation learning tasks. The first was an autoassociation task to train networks to discriminate between different stimuli. In the second task, the networks were trained to categorize stimuli by sorting lines into three categories: short, middle, long. The comparison of the pre- and post-categorization hidden activations showed the well known categorical perception effects, i.e. within-category compression and between-category expansion of category members. The hidden categorical representations constituted the grounding of categorization names (labels).

Subsequent symbol grounding models have focused on the mechanisms of grounding transfer. For example, Riga et al. [9] proposed a connectionist models in which basic symbols, such as names of shapes and colors, are first intrinsically connected to the categories being acquired through direct interaction with the environment. These basic symbols are successively used to construct descriptions of new categories of stimuli consisting of individual objects made by specific combinations of a shape and color). New categories (and their symbols) are in this way defined without the need of a direct experience of their referents. This process of grounding transfer enables the system to express meanings that go beyond immediate experience. New symbols, acquired exclusively from symbolic descriptions, are ultimately grounded in the interaction of the system with its environment.

The simulation consisted of three sequential training stages and a grounding transfer test phase. During the first training stage, an unsupervised network learns to discriminate between different stimulus categories by constructing a feature self-organizing map of different shape and color categories. The network acquires analogue sensorial representations of their environment that enables it to categorize the stimuli along the dimensions of shapes and colors. In the second training phase, symbolic stimuli (the names of the colors and shapes) are presented to the network, together with the images. These symbols are directly grounded in the sensorial representations acquired during the first phase. In the third training phase, the training input is exclusively symbolic. Linguistic descriptions of new higher-order categories are presented. These contain the previously acquired symbols in combination with a new symbol that denotes a new category (e.g. `Red + Square = DAX`). Finally, in the test phase, the network is presented with images of the previously unseen objects, such as `DAX`, to check if these are recognized and named. The successful naming of these previously-unseen images would demonstrate that the grounding transfer has occurred. Simulation results consistently showed that networks are able to recognize and name the images of new objects, therefore demonstrating that the grounding has been transferred from basic order categories to higher order concepts. Thus, the proposed connectionist simulation provides the basis of a working model for the implementation of an autonomous cognitive system able to use combinations of previously-grounded symbols to expand its knowledge of the world.

Other neural network models of language have focused on the grounding of special types of symbol, that is function words. These includes linguistic terms such as spatial prepositions (e.g. in, on, over, under) and quantifiers (e.g. few, some, many). Recently, Coventry et al. [5] have developed a neural network model of the spatial prepositions over, under, above, below. The model addresses the integration of functional and object knowledge factors ("what") with geometric factors such as the relative position of objects ("where"). The model processes movies of a located object (teapot) pouring a liquid (water) into a reference object (cup). The task of the network is to name the objects and to select the most appropriate spatial preposition describing the scene. The model consists of three modules: (1) a neurally-inspired vision module to process the visual scenes,

(2) a simple recurrent neural network to learn compressed representations of the dynamics of interacting objects, and (3) a dual-route network for producing the names of objects and the spatial prepositions. The dual-route network plays the core function of the grounding process by integrating visual and linguistic knowledge. The activation values of the linguistic output nodes correspond to rating values given by subjects in language comprehension experiments. The multilayer perceptron is trained via error backpropagation, by converting the rating data into stimulus presentation frequencies. Simulation results consistently show that the networks produce rating values similar to that of experimental subjects. It can also accurately predicts new experimental data on the ratings of scenes where only the initial frames are shown and the subjects must "mentally replay" the scene and predict its end frame (i.e. where the liquid ends). Such a model is currently being extended to deal with further linguistic terms, such as the vague quantifiers few, a few, some, many, a lot of. The underlying hypothesis is that this grounded connectionist approach will permit the identification of the main mechanisms responsible for quantification judgment and their linguistic expression.

3 Adaptive Agent Simulations

The adaptive agent approach includes multi-agent simulations of the evolution of language and cognitive robotics experiment on communication and language learning. The multi-agent approach uses populations of simulated agents that interact with each other to develop a shared set of symbols (lexicon) to describe their interaction with the world. The robotic studies uses embodied evolutionary and/or epigenetic robotic agents that interact in a simulated (or real) physical environment and build a linguistic representation of this interaction.

In a grounded multi-agent model of language evolution [2], neural networks were used to simulate learning and the genetic algorithm to simulate evolution. The model considers two ways of acquiring categories and language which are in direct competition with one another: In "sensorimotor toil," new categories are acquired through feedback-corrected, trial and error experience in sorting input stimuli. In "symbolic theft," new categories are acquired by hearsay from propositions (i.e. language) based on boolean combinations of symbols. In competition, symbolic theft always beats sensorimotor toil. This is hypothesized to be the basis of the adaptive advantage of language, after basic categories are learned by toil, to avoid an infinite regress (the symbol grounding problem). Changes in the internal representations of categories must take place during the course of learning by toil. These changes were analyzed in terms of the compression of within-category similarities and the expansion of between-category differences. Such compression/expansion effects, called Categorical Perception, have previously been reported with categories acquired by sensorimotor toil. This simulation also shows that they can also arise from symbolic theft alone.

Studies with adaptive robotic agents include simulations of robots that learn to imitate actions and to communicate linguistically about such motor abilities. For example, in an epigenetic robotic model [4], agents learn to perform actions

on objects using imitation and grounding transfer mechanisms. The model is based on a simulation of two robots embedded in a virtual environment that accurately models physical constraints present in real-world situations (using the physics software engine Open Dynamics Engine). Each robot has twelve Degrees of Freedom (DoF) and consists of two 3-segment arms attached to a torso and a base with 4 wheels. The teacher robot has preprogrammed behavior to manipulate objects. The imitator agent learns to perform the same actions by observing the teacher executing them and using an on-line backpropagation algorithm. This effectively enables the agent to mimic a movement in real-time and provides the agents with a mechanism to approximate movements without need for prior learning. The robot's neural network memorizes action patterns related to objects and enables the autonomous execution of the movement associated with an object in absence of the teacher agent. The neural controller receives in input sensorial data on the object's visual properties and the proprioceptive information on the imitator's joint angles. In output it produces the motor force applied to each joint. Overall, the simulation results show that it takes just few online training epochs to obtain a satisfactory performance. Typically, the agents produce a movement that is very similar to the original after having mimicked it once, and optimize it during successive training cycles.

Agents simultaneously learn the words corresponding to actions, whist they are are taught to perform the basic actions by mimicking them. Robot learn the basic actions of opening and closing their left and right arms (upper arms and elbows), lifting them (shoulders), and moving forward and backward (wheels), together with the corresponding words. They also learn higher-level composite behaviors by receiving linguistic descriptions containing these previously acquired words (grounding transfer). After basic grounding, the robot receives 1st level linguistic descriptions of combined actions, consisting in a new word and two known words referring to basic actions. For example, the action of grabbing the object in front of them was described as: "`CloseLeft + CloseRight = Grab`". Grounding is successfully transferred from the basic words `CloseLeft` and `CloseRight` to the higher order symbol `Grab`. In a test phase, when the agent is given the command `Grab` it successfully executes the combined action of pushing its arms towards the object and grabbing it. Robots can also receive further higher-level descriptions, in which a defining word is itself learned exclusively from a linguistic description. For example, the grabbing and moving forward actions were combined into the higher-order action of carrying: "`MoveForward + Grab = Carry`". Grounding transfer was successfully transferred to the new word, enabling the agent to correctly perform the action of carrying on hearing the word `Carry`. The system learned several of these combined actions simultaneously, and also four-word definitions and grounding transfers of up to three levels have been realized. In addition to demonstrating the grounding transfer mechanism in robotic agents, this model also highlights the role of language as a cognitive enhancer tool, i.e. a means through which new behaviors can be acquired quickly and effortlessly, building on experience accumulated by previous generations of agents.

4 Conclusions

Overall, these studies demonstrate the feasibility of the Cognitive Symbol Grounding approach in which neural networks are used to deal with the symbol grounding problems and the grounding transfer. The "embodiment" of such connectionist architectures in either simulated agents or robots permits a deeper understanding of the relationship between linguistic/symbolic abilities and other sensorimotor and cognitive capabilities. For example, adaptive agent models of verb and noun learning have shown that linguistic and sensorimotor representations share common neural structures. In these simulations, the same hidden units are involved or the processing of nouns and sensorimotor representations, whilst separate hidden units specialize for verb and motor processing [3]. This approach also has important practical and technological implications. For example, in robotics and artificial intelligence, language grounding models can provide novel algorithms and methodologies for the development of effective interaction between humans and autonomous computer and robotic systems. As demonstrated in the epigenetic robotic model of the symbol grounding transfer, the imitation and language instruction modalities can be integrated to form a situated learning process in which higher-order linguistic representations can be autonomously grounded into the agents' own sensorimotor and cognitive abilities.

References

1. Cangelosi, A.: Grounding symbols in perceptual and sensorimotor categories: Connectionist and embodied approaches. In: Cohen, H., Lefebvre, C. (Eds), Handbook of Categorization in Cognitive Science, Elsevier (2005)
2. Cangelosi, A., Harnad, S.: The Adaptive Advantage of Symbolic Theft over Sensorimotor Toil: Grounding Language in Perceptual Categories. Evolution of Communication **4** (2000) 117–142
3. Cangelosi, A., Parisi, D.: The processing of verbs and nouns in neural networks: Insights from synthetic brain imaging. Brain and Language, **89** (2004) 401–408
4. Cangelosi, A., Riga, T.: Adaptive agent and robotic approaches to the sensorimotor ground-ing of language Artificial Intelligence Journal (in submission)
5. Coventry, K.R.,Cangelosi, A., Rajapakse, R., Bacon, A., Newstead, S., Joyce, D., Richards, L.V.: Spatial prepositions and vague quantifiers: Implementing the functional geometric framework. Spatial Cognition Conference, Germany, 11–13 October (2004)
6. Harnad, S.: The Symbol Grounding Problem. Physica D, **42** (1990) 335–346
7. Harnad, S., Hanson, S. J. and Lubin, J.: Categorical perception and the evolution of super-vised learning in neural nets. In D.W. Powers and L. Reeker (Eds.), Proceedings of Proceedings of the AAAI Spring Symposium on Machine Learning of Natural Language and Ontology (1991)
8. Plunkett, K., Sinha, C., Mller, M.F., Strandsby, O.: Symbol grounding or the emergence of symbols? Vocabulary growth in children and a connectionist net. Connection Science, **4** (1992) 293–312
9. Riga, T., Cangelosi, A., Greco, A.: Symbol grounding transfer with hybrid self-organizing/supervised neural networks. Proceedings of IJCNN04 International Joint Conference on Neural Networks. Budapest, July (2004)

The Complexity of Inductive Definability

Douglas Cenzer[1] and Jeffrey B. Remmel[2,⋆]

[1] Department of Mathematics, University of Florida,
P.O. Box 118105, Gainesville, FL, USA 32611
cenzer@math.ufl.edu
[2] Department of Mathematics,
University of California at San Diego,
La Jolla, CA 92093, USA
jremmel@ucsd.edu

Abstract. We study the complexity of computable and Σ^0_1 inductive definitions of sets of natural numbers. For we example, we show how to assign natural indices to monotone Σ^0_1-definitions and we use these to calculate the complexity of the set of all indices of monotone Σ^0_1-definitions which are computable. We also examine the complexity of new type of inductive definition which we call *weakly finitary* monotone inductive definitions. Applications are given in proof theory and in logic programming.

1 Introduction

Inductive definitions play a central role in mathematical logic. For example, the set of formulas in a first order language is given by an inductive definition. Given a set A of axioms for a mathematical theory T and a set of logical axioms and rules, the theory T is obtained by an inductive definition. The set of computable functions can be realized by an inductive definition. Similarly, for any Horn logic program P, the unique stable model of P is obtained by an inductive definition.

It is well-known that any computable or Σ^0_1 monotone inductive definition Γ, one can construct the closure of Γ, $Cl(\Gamma)$, in at most ω steps and $Cl(\Gamma)$ is always a Σ^0_1 set. In some situations, it is important that $Cl(\Gamma)$ is computable. For example, it is important that the set of formulas in a typical first order theory is recursive. In other situations, we know that $Cl(\Gamma)$ is Σ^0_1 but not computable. For example, even a finitely axiomatized theory T may be Σ^0_1 but not decidable (computable). In this paper, we shall explore the complexity of when a Σ^0_1 monotone inductive definition Γ has a computable closure. We shall do this by assigning indices to Σ^0_1 monotone inductive operators. In particular, this means that we can effectively enumerate the family of all Σ^0_1 monotone inductive operators as $\Gamma_0, \Gamma_1, \ldots$. We shall then show that the set C of indices e such that the closure, or *least fixed point* $\mathrm{lfp}(\Gamma_e)$ is computable is Σ^0_3 complete.

⋆ The second author was partially supported by NSF grant DMS 0400507.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 75–85, 2005.

We will also define a new class of inductive operators called *weakly finitary* monotone inductive operators. The basic idea is that for a weakly finitary operator there may exists a finite set of elements x such that x is forced into $\Gamma(A)$ only if A contains one of a collection of possibly infinite sets. We will show that if Γ is a weakly finitary monotone inductive operator, then it will still be the case that $\mathrm{lfp}(\Gamma)$ will be Σ^0_1 but that it can take more than ω steps to construct $\mathrm{lfp}(\Gamma)$. An example of such an operator is when we allow finitely many instances of the ω-rule to generate a partial theory of arithmetic. We also assign indices to the family of weakly finite Σ^0_1 monotone inductive operators. We shall show that the set of indices of weakly finitary Σ^0_1 monotone inductive operators Γ such that $\mathrm{lfp}(\Gamma)$ is computable is also Σ^0_3 complete. However, for certain computable sets R, the set of indices of weakly finitary Σ^0_1 monotone inductive operators Γ such that $\mathrm{lfp}(\Gamma) \cap R$ is computable lies in the difference hierarchy over the Σ^0_3 sets.

We will use standard notation from computability theory [6]. In particular, we let ϕ_e (ϕ^A_e) denote the e-th partial computable function (e-th A-partial computable function) from $\mathbb{N}$ to $\mathbb{N}$ and let $W_e = Dom(\phi_e)$ ($W^A_e = Dom(\phi^A_e)$) be the e-th computably enumerable (c.e.) (e-th A-computably enumerable) subset of $\mathbb{N}$. We let $W_{e,s}$ ($W^A_{e,s}$) denote the set of numbers $m \leq s$ such that $\phi_e(m)$ ($\phi^A_e(m)$) converges in s or fewer steps. Given a finite set $S = \{a_1 < \ldots a_n\}$, the canonical index of S is $\sum_{i=1}^{n} 2^{a_i}$. The canonical index of the empty set is 0. We let D_n denote the finite set whose canonical index is n. We fix some primitive recursive pairing function $\langle , \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We then code k-tuples for $k \geq 3$, inductively by $\langle n_1, \ldots, n_k \rangle = \langle n_1, \langle n_2, \ldots, n_k \rangle \rangle$. We then let the canonical index $[a_1, \ldots, a_k]$ of a k-tuple $(a_1, \ldots, a_k) \in \mathbb{N}^k$ be given by $[a_1, \ldots, a_k] = \langle k, \langle a_1, \ldots, a_k \rangle \rangle$. It easily follows that for each k, the coding function $[,] : \mathbb{N}^k \to \mathbb{N}$ and the projection functions given by $[a_1, \ldots, a_k]_i = a_i$ are primitive recursive.

2 Inductive Definitions

In this paper, we are going to consider inductive operators $\Gamma : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ which inductively define subsets of $\mathbb{N}$. We begin with a review of basic definitions and results which can be found, for example, in Hinman [5].

Definition 1. *Let $\Gamma : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$.*

1. *Γ is said to be* monotone *if $A \subset B$ implies $\Gamma(A) \subseteq \Gamma(B)$ for all A, B.*
2. *Γ is said to be* inclusive *if $A \subseteq \Gamma(A)$ for all A.*
3. *Γ is said to be* inductive *if it is either monotone or inclusive.*

An inductive operator Γ recursively defines a sequence $\{\Gamma^\alpha : \alpha \text{ an ordinal}\}$ by setting $\Gamma^0 = \emptyset$, $\Gamma^{\alpha+1} = \Gamma(\Gamma^\alpha)$ for all α and $\Gamma^\lambda = \bigcup_{\alpha<\lambda} \Gamma^\alpha$. It is easy to see that $\Gamma^\alpha \subseteq \Gamma^\beta$ whenever $\alpha < \beta$. By cardinality considerations, there exists a countable ordinal α such that $\Gamma^\alpha = \Gamma^\beta$ for all $\beta > \alpha$. The least such α is called the *closure ordinal* of Γ and will be denoted by $|\Gamma|$. The set $\Gamma^{|\Gamma|}$ is called the closure of Γ or the set inductively defined by Γ and will be denoted by $Cl(\Gamma)$.

For a monotone operator, the closure is also the least fixed point $\mathrm{lfp}(\Gamma)$ as indicated by the following.

Lemma 2. *If Γ is a monotone operator, then $Cl(\Gamma)$ is the unique least set C such that $\Gamma(C) = C$. That is, for any set A such that $\Gamma(A) = A$, $\Gamma(C) \subseteq A$.*

For any operator $\Gamma : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$, let $R_\Gamma \subseteq \mathbb{N} \times \mathcal{P}(\mathbb{N})$ be given by $R_\Gamma(m, A) \iff m \in \Gamma(A)$. In general, we say that a predicate $R(x_1, \ldots, x_k, A) \subseteq \mathbb{N}^k \times \mathcal{P}(\mathbb{N})$ is computable if there is an oracle Turing machine M_e such that for any $A \in \mathcal{P}(\mathbb{N})$, M_e with oracle A and input $(x_1, \ldots, x_n)$ outputs 1 if $R(x_1, \ldots, x_n, A)$ holds and outputs 0, otherwise. The notation of a predicate being $\Sigma^0_n, \Pi^0_n, \Sigma^1_1, \Pi^1_1$, etc. can then be defined as usual over the class of computable predicates. We then say that an operator Γ is computable (respectively Σ^0_1, arithmetical, etc.) if the relation R_Γ is recursive (respectively Σ^0_1, arithmetical, etc.). The following results are well-known.

Theorem 3. *1. If Γ is a computable inductive operator, then the sequence $\{\Gamma^n : n \in \omega\}$ is uniformly computable, $|\Gamma| \leq \omega$ and $Cl(\Gamma)$ is Σ^0_1.*
2. If Γ is a Σ^0_1 inductive operator, then $|\Gamma| \leq \omega$ and if Γ is monotone, then $Cl(\Gamma)$ is Σ^0_1.
3. Any Σ^0_1 set is 1-1 reducible to the closure of some computable monotone inductive operator.
4. If Γ is a monotone arithmetical operator, then $|\Gamma| \leq \omega_1^{CK}$ (the least non-recursive ordinal) and $\mathrm{lfp}(\Gamma)$ *is Π^1_1.*
5. Any Π^1_1 set is 1-1 reducible to the closure of a monotone Π^0_1 inductive operator.

Example 4. The classic example of a computable monotone operator is given by the definition of the set of sentences of a propositional logic over an infinite set $a_0, a_1, \ldots$ of propositional variables. Identifying sentences p, q with their Gödel number $gn(p), gn(q)$, we have for any i, p, q,and A:

(0) $a_i \in \Gamma(A)$,
(1) $\neg p \in \Gamma(A)$ if $p \in A$, and
(2) $p \wedge q \in \Gamma(A)$ if $p \in A$ and $q \in A$, and
(3) $p \in \Gamma(A)$ if $p \in A$.

Other clauses could be added to include conjunction, implication or other binary connectives. This operator is computable because for any sentence p, we can compute the (at most two) other sentences which need to be in A for p to get into $\Gamma(A)$. Similar computable inductive definitions can be given for the set of terms in a first order language and the set of formulas in predicate logic. In each case, the closure ordinal of such a Γ is ω and the set of sentences (respectively, terms, formulas) is is computable since for any sentence (term, formula) p of length n, $p \in \mathrm{lfp}(\Gamma) \iff p \in \Gamma^n$.

Example 5. Suppose we are given a computable or Σ^0_1 set A_0 of axioms for propositional logic together with the logical axioms $\neg p \vee p$ for each p and a finite set of rules as indicated below. Then the set of consequences of A_0 is generated by the operator Γ where, for all sentences p, q, r and all A:

(0) $p \in \Gamma(A)$ if p is an axiom,
(1) $p \vee q \in \Gamma(A)$ if $p \in A$ or $q \in A$,
(2) $p \in \Gamma(A)$ if $p \vee p \in A$,
(3) $(p \vee q) \vee r \in \Gamma(A)$ if $p \vee (q \vee r) \in A$, and
(4) $q \vee r \in \Gamma(A)$ if $p \vee q \in A$ and $\neg p \vee r \in A$.

In this case, Γ is a Σ^0_1 operator but is not computable since, for example, the Cut Rule (4) asks for the existence of a p such that $p \vee q$ and $\neg p \vee r$ are in A.

Now in fact the consequences of a recursive set A_0 will be a recursive set but a similar example can be given for first order logic where the consequences of a finite set of axioms for arithmetic is Σ^0_1 but not recursive.

Example 6. The one-step provability operator for a computable Horn logic program is a Σ^0_1 monotone operator. That is, suppose A is a computable set of propositional letters or atoms. We assume that $A = \mathbb{N}$. A logic programming clause is a construct of the form

$$C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_n$$

where $p, q_1, \ldots, q_m, r_1, \ldots, r_n$ are atoms. Given a clause C, we let

$$[C] = \langle p, [q_1, \ldots, q_n], [r_1, \ldots, r_m] \rangle$$

where by convention, we let $[q_1, \ldots, q_n] = 0$ if $n = 0$ and $[r_1, \ldots, r_m] = 0$ if $m = 0$. The atoms $q_1, \ldots, q_m, \neg r_1, \ldots, \neg r_n$ form the *body* of C and the atom p is its *head*. Given a set of atoms $M \subseteq A$, we say M is a model of C either (i) there is an q_i such that $q_i \notin M$ or there is an r_j such that $r_j \in M$ (M does not satisfy the body of C) or (ii) $p \in M$ (M satisfies the head of C). The clauses C where $n = 0$ are called *Horn* clauses.

A program P is a set of clauses. We say that P is computable (Σ^0_1, arithmetical, etc.) if $\{[C] : C \in P\}$ is computable (Σ^0_1, arithmetical, etc.). A program entirely composed of Horn clauses is called a Horn program. If P is a Horn program, then there is a one step provability operator associated with P, $T_P : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$, which is defined by

$T_P(A)$ equals the set of all p such that there exists a clause $C = p \leftarrow q_1, \ldots, q_n$ in P such that $q_1, \ldots, q_n \in A$.

A Horn program always has a least model which is the closure of T_p. It is the intended semantics of such a program.

For programs with bodies containing the negation operator *not*, we will use the stable model semantics. Following [4], we define a *stable model* of the program as follows. Assume M is a collection of atoms. The *Gelfond-Lifschitz reduct* of P by M is a Horn program arising from P by first eliminating those clauses in P which contain $\neg r$ with $r \in M$. In the remaining clauses, we drop all negative literals from the body. The resulting program $GL_M(P)$ is a Horn program. We call M a stable model of P if M is the least model of $GL_M(P)$. In

the case of a Horn program, there is a unique stable model, namely, the least model of P.

It should be pointed out that both Example 1 and Example 2 can reformulated in the framework of logic programming as computable Horn programs. That is, the set of rules is a computable set, even though the corresponding inductive operator need not be computable.

3 Index Sets for Σ_1^0 Monotone Operators

An important property of Σ_1^0 monotone operators Γ is that the relation $m \in \Gamma(A)$ depends only on positive information about A. That is, we have the following lemma, see [5], pg. 92.

Lemma 7. *For any Σ_1^0 monotone operator Γ, there is a recursive relation R such that for all $m \in \mathbb{N}$ and $A \in \mathcal{P}(\mathbb{N})$,*

$$m \in \Gamma(A) \iff (\exists n)(D_n \subseteq A \ \&\ R(m,n)) \tag{1}$$

It follows from Lemma 7 that the Σ_1^0 monotone inductive operators may be effectively enumerated as $\Gamma_0, \Gamma_1, \ldots$ in the following manner. For all $e, m \in \mathbb{N}$ and all $A \in \mathcal{P}(\mathbb{N})$, let

$$m \in \Gamma_e(A) \iff (\exists n)[D_n \subseteq A \text{ and } \langle m,n\rangle \in W_e].$$

Lemma 8. *1. There is a primitive recursive function f such that for all m, e, a:*

$$\Gamma_e(W_a) = W_{f(e,a)}.$$

2. The relation $m \in \Gamma_e^t$ is Σ_1^0.
3. The relation $m \in \mathrm{lfp}(\Gamma_e)$ is Σ_1^0.

Proof. (1) We have

$$m \in \Gamma_e(W_a) \iff (\exists n)[D_n \subseteq W_a \text{ and } \langle m,n\rangle \in W_e].$$

Thus we may define a partial computable function $\phi_c(e,a,m)$ which will search for the least pair $\langle n,s\rangle$ such that $D_n \subseteq W_{a,s}$ and $\langle m,n\rangle \in W_{e,s}$. Then

$$m \in \Gamma_e(W_a) \iff \langle e,a,m\rangle \in Dom(\phi_c).$$

Now the *s-m-n* theorem will provide a primitive recursive f such $\phi_{f(e,a)}(m) = \phi_c(e,a,m)$.

(2) Let $W_0 = \emptyset$ and let f be given by (1). For any fixed e, let g_e be the partial recursive function defined by $g_e(a) = f(e,a)$. Then clearly, $\Gamma_e^t = W_{g_e^t(0)}$.

(3) This follows from the fact that $m \in \mathrm{lfp}(\Gamma_e) \iff (\exists t)(m \in \Gamma_e^t)$. In fact, it is easy to see that there is computable function h such that $\mathrm{lfp}(\Gamma_e) = W_{h(e)}$. □

Theorem 9. $\{e : \mathrm{lfp}(\Gamma_e)$ *is computable*$\}$ *is* Σ^0_3 *complete.*

Proof. Let $C = \{e : \mathrm{lfp}(\Gamma_e)$ is computable$\}$. Let h be the computable function defined in the proof of part (3) of Lemma 8. Then $C = \{e : W_{h(e)}$ is computable$\}$. The predicate "W_a is computable" is Σ^0_3 so that C is a Σ^0_3 set.

For completeness, we use the well-known fact [6] that $Rec = \{e : W_e$ is recursive$\}$ is Σ^0_3 complete. We can use the s-m-n theorem to define the operator $\Gamma_{g(e)}$ by

$$\langle m, s\rangle \in \Gamma_{g(e)}(A) \iff m \in W_{e,s} \text{ or } \langle m, s+1\rangle \in A.$$

It is easy to see that $\mathrm{lfp}(\Gamma_{g(e)}) = W_e \times \mathbb{N}$. Thus W_e is recursive if and only if $\mathrm{lfp}(\Gamma_{g(e)})$ is recursive. Hence $e \in Rec \iff g(e) \in C$ so that C is Σ^0_3 complete. □

Computable operators are continuous and we can use the indexing of [3] to define the eth computable monotone operator Δ_e for e in the Π^0_2 set of indices such that ϕ_e is a total function. Then a similar result to Theorem 9 holds for the family of *computable* monotone operators since the operator $\Gamma_{g(e)}$ is computable for all e.

4 Weakly Finitary Monotone Operators

The idea of a weakly finitary operator is to have a finite set $m_1, \ldots, m_k$ of *exceptional* numbers which may be put into $\Gamma(A)$ when an *infinite* set is included in A. If there are exactly k exceptional numbers, then the operator Γ will be called k-weakly finitary. For example, we might allow some finite number of consequences of the ω-rule in a subsystem of Peano arithmetic and still obtain a c.e. theory.

Definition 10. *1. We say that an inductive operator* $\Gamma : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ *is* weakly finitary *if there is a finite set* S_Γ *such that for all* A,

(a) $x \notin S_\Gamma$ *and* $x \in \Gamma(A)$ *implies there exists a finite set* $F \subseteq A$ *such that* $x \in \Gamma(F)$ *and*

(b) $x \in S_\Gamma$ *and there is a family* $\mathcal{F}_{\Gamma,s,A}$ *of subsets of* Γ *which includes at least one infinite subset of* Γ *such that* $x \in \Gamma(A)$ *implies there exists an* $F \subseteq A$ *such that* $x \in \Gamma(F)$ *for some* $F \in \mathcal{F}_{\Gamma,x,A}$.

If $|S_\Gamma| = k$, *then we say that* Γ *is* k-weakly finitary.

2. We say $\Gamma = \Lambda_{k,e}$ *is a* k*-weakly finitary* Σ^0_1 *monotone inductive operator with index* $\langle k, e\rangle = \langle k, \langle d, \langle m_1, e_1, \ldots, m_k, e_k\rangle\rangle$ *if* Γ *is a weakly finitary monotone operator with* $S_\Gamma = \{m_1 < \cdots < m_k\}$ *and for all* A *and* m_i, $\mathcal{F}_{\Gamma,s,A} = \{W_a : a \in W_{e_i}\}$. *Thus for all* $A \in \mathcal{P}(\mathbb{N})$ *and* $m \in \mathbb{N}$, $m \in \Lambda_{k,e}(A)$ *if and only if either*

(i) $m \in \Gamma_d(A)$ *or*

(ii) *for some* i, $m = m_i$ *and* $(\exists a \in W_{e_i})(W_a \subseteq A)$.

Example 11. One example of this type of operator comes from attempts to extend logic programming to be able to reason about infinite sets described in [1, 2]. That is, suppose that we consider a Horn program P and we construct an extended program by adding extended clauses of the form

$$C = p \leftarrow a_1, \ldots, a_n, W_{e_1}, \ldots, W_{e_m}. \tag{2}$$

We call p the *head* of C and $a_1, \ldots, a_n, W_{e_1}, \ldots, W_{e_m}$ the body of C. We say a set of atoms M satisfies the body of C if $\{a_1, \ldots, a_n\} \cup \bigcup_{i=1}^{m} W_{e_i}$ is contained in M. We say that M satisfies C if either M does not satisfy the body of C or M satisfies the head of C, i.e. $p \in M$. In such a case, there is still a natural one-step provability operator associated to the extended program P', namely, $T_{P'}(A)$ equals the set of p such that there exists a clause or an extended clause $C \in P'$ which M satisfies of the body of C and p equals the head of C. If the sets of heads of the extended clauses in P' is finite, then $T_{P'}$ is a weakly finitary monotone operator.

It is no longer the case that one can find the least fixed point of $T_{P'}$ in ω steps. For example, consider the following extended program P where W_e is the set of even natural numbers. Consider the following program.

$$\begin{aligned} 0 &\leftarrow \\ 2x+2 &\leftarrow 2x \quad \text{(for every number } x\text{)} \\ 1 &\leftarrow W_e \quad \text{for all } n \in \omega \\ 2x+3 &\leftarrow 2x+1 \quad \text{(for every number } x\text{)} \end{aligned}$$

Clearly ω is the least model of P but it takes 2ω steps to reach the fixed point. That is, it is easy to check that $T_P \uparrow^{\omega} = W_e$ and that $T_P \uparrow^{\omega+\omega} = \omega$.

We note in [1, 2], we consider a much richer class of programs where the one-step provability operator is a weakly finitary monotone operator.

Theorem 12. *Let Λ be a k-weakly Σ_1^0 monotone operator with index $\langle k, e\rangle = \langle k, \langle d, \langle m_1, e_1, \ldots, m_k, e_k\rangle\rangle$. Then*

1. $|\Lambda| \leq \omega \cdot (k+1)$.
2. $\mathrm{lfp}(\Lambda)$ *is* Σ_1^0.

Proof. We will present an informal procedure which constructs the closure in $\leq k+1$ rounds where each round may consist of as many ω steps.

Round (1). First let $U_0 = \mathrm{lfp}(\Gamma_d)$. Since Γ_d is a Σ_1^0 monotone inductive operator, U_0 is c.e. by Theorem 3. Next consider the finite set

$$F_0 = \{m_i : (\exists a \in W_{e_i})(W_a \subseteq U_0)\}.$$

We can not necessarily find F_0 effectively, but, nevertheless, F_0 is a finite set so that $A_1 = U_0 \cup F_0$ will be a c.e. set. If $F_0 = \emptyset$, then $\mathrm{lfp}(\Lambda) = U_0$ and $|\Lambda| \leq \omega$. Otherwise go on to Round 2.

We now present the description of Round $n+1$, for $n \geq 1$. Assume that A_n is the result of step n.

Round $(n+1)$**.** Consider the set $U_n = \Gamma_d^\omega(A_n)$. It is easy to see that that since A_n is c.e., U_n is also c.e.. Next consider the finite set

$$F_n = \{m_i : (\exists a \in W_{e_i})(W_a \subseteq U_n)\}.$$

Again we can not necessarily find F_n effectively, but, nevertheless, $A_{n+1} = U_n \cup F_n$ is a c.e. set. Now if $F_n = \emptyset$, then $\mathrm{lfp}(\Lambda) = U_n$ and $|\Lambda| \leq \omega \cdot (n+1)$. Otherwise go on to Round $(n+2)$.

It is clear that this process must be completed after at most $k+1$ rounds, so that $|\Lambda| \leq \omega \cdot (k+1)$ and $\mathrm{lfp}(\Lambda)$ is always a c.e. set. □

The following lemma gives an alternate approach to proving part (2) of Theorem 12 and will be needed below.

Lemma 13. *Let Λ be a k-weakly finitary Σ_1^0 monotone operator with index*

$$\langle k, e\rangle = \langle k, \langle d, \langle m_1, e_1, \ldots, m_k, e_k\rangle\rangle.$$

Then for some finite subset F of $\{m_1, \ldots, m_k\}$, $\mathrm{lfp}(\Lambda) = \Gamma_d^\omega(F)$.

Proof. Let $F = \{m_i : m_i \in \mathrm{lfp}(\Lambda)\}$. Then certainly $\Gamma_d^\omega(F) \subseteq \Lambda^\omega(F) \subseteq \mathrm{lfp}(\Lambda)$. For the reverse inclusion, it suffices to show that $C = \Gamma_d^\omega(F)$ is closed under Λ. If $\Lambda(C) - C \neq \emptyset$, then either (i) there is some $y \notin S_\Gamma = \{m_1, \ldots, m_k\}$ such that $y \in \Gamma_d(C)$ or (ii) there is some m_i such $W_a \subseteq C$ for some $a \in W_{e_i}$. Note that (i) is not possible since $\Gamma_d(C) \subseteq C$ because Γ_d is a Σ_1^0 monotone operator, so $\Gamma_d(\Gamma^\omega(F)) = \Gamma^\omega(F)$. But (ii) is not possible since otherwise $m_i \in F$ and $F \subseteq C$. Thus it must be the case that $\Lambda(C) = \Gamma_d(C)$. □

Next we need to define the family of difference sets of Σ_3^0 sets. For two Σ_3^0 sets A and B , the difference $A - B$ is the intersection of Σ_3^0 set and a Π_3^0 set and is said to be a 2-Σ_3^0 set. For $n > 0$, we say that a set C is $2n$-Σ_3^0 if and only if A is the union of n 2-Σ_3^0 sets and is $2n+1$-Σ_3^0 if and only if A is the union of a Σ_3^0 set with a $2n$-Σ_3^0 set. We say that A is n-Π_3^0 set if the complement of A is n-Σ_3^0 set.

We can then prove the following.

Theorem 14. *Fix an infinite coinfinite recursive set R_t. Then for each k, $\{e : \mathrm{lfp}(\Lambda_{k,e}) \cap R_t$ is computable$\}$ is a $(2^{k+1}-1)$-Σ_3^0 set.*

Proof. Fix a set $F \subseteq \{1, \ldots, k\}$. For each index $\langle k, e\rangle = \langle k, \langle d, \langle m_1, e_1, \ldots, m_k, e_k\rangle\rangle$, let $M_{F,k,e} = \Gamma_d^\omega(\{m_i : i \in F\})$. We are interested in analyzing the predicate that

$$P(F,k,e) : \ M_{F,k,e} = \mathrm{lfp}(\Gamma_{k,e}) \ \&\ R_t \cap M_{F,k,e} \text{ is computable.} \tag{3}$$

It follows from Lemma 13 that $\mathrm{lfp}(\Lambda_{k,e}) = M_{F,k,e}$ if and only if

1. $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq M_{F,k,e})\} \subseteq \{m_i : i \in F\}$ and
2. for all $G \subsetneq F$, $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq M_{G,k,e})\} \not\subseteq \{m_i : i \in G\}$.

The predicate that $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq M_{G,k,e})\} \not\subseteq \{m_i : i \in G\}$ is Σ^0_3 since it holds iff there is an $i \in \{1, \ldots, k\} - G$ such that $(\exists a)(a \in W_{e_i}\ \&\ W_a \subseteq M_{G,k,e})$. Since $M_{G,k,e}$ is uniformly c.e., the predicate $W_a \subseteq M_{G,k,e}$ is Π^0_2 and hence the predicate $(\exists a)(a \in W_{e_i}\ \&\ W_a \subseteq M_{G,k,e})$ is Σ^0_3. It follows that the predicate $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq M_{F,k,e})\} \subseteq \{m_i : i \in F\}$ is Π^0_3 if $F \neq \{1, \ldots, k\}$. Finally, the predicate "$M_{F,k,e} \cap R_t$ is computable" is Σ^0_3. Thus if $F \neq \{1, \ldots, k\}$, the predicate $P(F, k, e)$ is the conjunction of a Σ^0_3 and Π^0_3 predicate and hence is 2-Σ^0_3 predicate. If $F = \{1, \ldots, k\}$, then, we may omit the Π^0_3 predicate so that $P(F, k, e)$ is a Σ^0_3 predicate.

It follows that the predicate that $\{e : \mathrm{lfp}(\Gamma_{k,e}) \cap R_t$ is computable$\}$ is a disjunction of $2^k - 1$ 2-Σ^0_3 sets and one Σ^0_3 set and hence a $2^{k+1} - 1$ set. □

It is important to note that the set of all $\langle k, e\rangle$ such that $\mathrm{lfp}(\Lambda_{k,e})$ *itself* is computable is just Σ^0_3. That is, for each finite $F \subseteq \{1, \ldots, k\}$ and each computable set R, the question of whether $R = M_{F,k,e}$ is a Π^0_2 question since $M_{F,k,e}$ is uniformly c.e.. If there is an F such that $R = M_{F,k,e}$, then the question of whether $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq R)\} \subseteq \{m_i : i \in F\}$ is a Π^0_2 question. That is, the question whether $W_a \subseteq R$ is a Π^0_1 question so that the question of whether $(\exists i \in \{1, \ldots, k\} - F)(\exists a)(a \in W_{e_i}\ \&\ W_a \subseteq R)$ is a Σ^0_2 question. Thus $\mathrm{lfp}(\Lambda_{k,e})$ is computable if and only if there is an s and there exists an $F \subseteq \{1, \ldots, k\}$ such that W_s is computable, $M_{F,k,e} = W_s$, $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq W_s)\} \subseteq \{m_i : i \in F\}$, and for all $G \subsetneq F$, $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq M_{G,k,e})\} \not\subseteq \{m_i : i \in G\}$. Since the predicates W_s is computable, $M_{F,k,e} = W_s$, and $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq W_s)\} \subseteq \{m_i : i \in F\}$ are all Π^0_2 and the predicates $\{m_i : (\exists a \in W_{e_i})(W_a \subseteq M_{G,k,e})\} \not\subseteq \{m_i : i \in G\}$ are Σ^0_3, the predicate $\mathrm{lfp}(\Lambda_{k,e})$ is computable is Σ^0_3. We can then proceed as in proof of Theorem 12 to prove $\{\langle k, e\rangle : \mathrm{lfp}(\Lambda_{k,e})$ is computable$\}$ is Σ^0_3-complete. Thus we have the following.

Theorem 15. $\{\langle k, e\rangle : \mathrm{lfp}(\Lambda_{k,e})$ *is computable*$\}$ *is* Σ^0_3*-complete.*

Finally, we give a completeness result for Theorem 14 in the case where $k = 1$.

Theorem 16. *Let* R_t *be a fixed infinite coinfinite recursive set.* $\{e : \mathrm{lfp}(\Lambda_{1,e}) \cap R_t$ *is computable*$\}$ *is* 3-Σ^0_3*-complete.*

Proof. The upper bound on the complexity is given by the proof of Theorem 14. For the other direction, fix $R_t = \{2n : n \in \mathbb{N}\}$ without loss of generality. Let $C = \{e : \mathrm{lfp}(\Lambda_{1,e}) \cap R_t$ is computable$\}$. Note that it is proved in [6] that $Rec = \{e : W_e$ is computable$\}$ and $Cof = \{e : W_e$ is cofinite$\}$ are Σ^0_3 complete.

For the completeness, first we claim that

$$D = \{\langle a, b, c\rangle : (W_a \text{ is not cofinite } \&\ W_b \text{ is recursive}) \ \vee\ W_c \text{ is recursive}\}$$

is 3-Σ^0_3 complete. That is, let $S = (B - A) \cup C$, where A,B,C are Σ^0_3. Then there are functions f, g, h such that $a \in A \iff f(a) \in Cof$, $b \in B \iff g(b) \in Rec$, and $c \in C \iff h(c) \in Rec$. Thus $< a, b, c >\in S$ iff $[(f(a) \notin Cof)$ and $g(b) \in Rec)$ or $h(c) \in Rec]$ iff $\phi(s) =< f(a), g(b), h(c) >\in D$. Thus it suffices to reduce

D to C. That is, we will define a 1-weakly finitary Σ^0_1 monotone operator $\Lambda_{f(a,b,c)}$ such that $\{\text{lfp}(\Lambda_{f(a,b,c)}) \cap R_t \text{ is computable}\}$ if and only if $\langle a,b,c\rangle \in D$. Since Rec and Cof are Σ^0_3 complete, it follows that there exists a computable function g such that W_c is recursive or W_a is cofinite if and only if $W_{g(a,c)}$ is cofinite. Let $\omega = W_m$ and let h be a computable function such that for each n, $W_{h(n)} = \{8i+3 : i > n\}$. The inductive operator $\Lambda = \Lambda_{f(a,b,c)}$ is defined by the following clauses.

(1) $0 \in \Lambda(A)$ if $W_{h(n)} \subseteq A$ for some n.

(2) $8\langle i,s\rangle + 1 \in \Lambda(A)$ if $i \in W_{g(a,c),s}$ or $8\langle i,s+1\rangle + 1 \in A$.

(3) $8i+3 \in \Lambda(A)$ if $8\langle i,0\rangle + 1 \in A$.

(4) $8\langle i,s\rangle + 5 \in \Lambda(A)$ if $i \in W_{b,s}$ or $8\langle i,s+1\rangle + 5 \in A$.

(5) $8i+2 \in \Lambda(A)$ if $8\langle i,0\rangle + 5 \in A$.

(6) $8\langle i,s\rangle + 7 \in \Lambda(A)$ if $0 \in A$ and either $i \in W_{c,s}$ or $8\langle i,s+1\rangle + 7 \in A$.

(7) $8i+4 \in \Lambda(A)$ if $8\langle i,0\rangle + 7 \in A$.

(8) $8i+2 \in \Lambda(A)$ if $0 \in A$.

It is easy to see that clauses (2)-(8) define a computable monotone inductive operator so that Λ is a 1-weakly finitary Σ^0_1 operator where $S_\Lambda = \{0\}$.

Clauses of type (2) and (3) ensure that $\text{lfp}(\Lambda)$ must include $\{8i+3 : i \in W_{g(a,c)}\}$ and clauses of type (4) and (5) ensure that $\text{lfp}(\Lambda)$ must include $\{8i+2 : i \in W_b\}$.

Let $M = \text{lfp}(\Lambda)$. If $W_{g(a,c)}$ is cofinite, then one of the clauses of type (1) will apply and then the clauses of type (6), (7), and (8) will ensure that $M \cap R_t$ equals $\{0\} \cup \{8i+2 : i < \omega\} \cup \{8i+4 : i \in W_c\}$ and, hence, $M \cap R_t$ will be computable if and only if W_c is computable. If $W_{g(a,c)}$ is not cofinite, then $M \cap R_t$ will consist of $\{8i+2 : i \in W_b\}$ and, hence, $M \cap R_t$ will be computable if and only if W_b is computable.

If $\langle a,b,c\rangle \in D$, then there are two cases. First suppose that W_c is computable. Then $W_{g(a,c)}$ is cofinite so that $M \cap R_t$ is computable as desired. Next suppose that W_c is not computable. Then we must have W_a not cofinite and W_b computable. In this case, $W_{g(a,c)}$ is not cofinite and $M \cap R_t$ is again computable.

If $\langle a,b,c\rangle \notin D$, then W_c is not computable and either W_a is cofinite or W_b is not computable. Again there are two cases. First suppose that W_a is cofinite. Then $W_{g(a,c)}$ is cofinite, so that $M \cap R_t$ is not computable, as desired. If W_a is not cofinite, then $W_{g(a,c)}$ is not cofinite and W_b is not computable. Thus again $M \cap R_t$ is not computable. □

We conjecture that a similar completeness result will hold for k-weakly Σ^0_1 operators. Finally, we remark that k-weakly computable monotone operators may be defined and corresponding versions of Theorems 14, 15 and 16 can be shown.

References

1. Cenzer, D., Marek, W., Remmel, J.B.: Using logic programs to reason about infinite sets. Proceedings of the Symposium on Mathematics and Artificial Intelligence (AIM 2004). http:rutcor.rutgers.edu/~amai/aimath04
2. Cenzer, D., Marek, W., Remmel, J.B.: Logic programming with snfinite sets. Annals of Mathematics and Artificial Intelligence (to appear)
3. Cenzer, D., Remmel, J.B.: Index sets in computable analysis. Theoretical Computer Science 219 (1999) 111-150
4. M. Gelfond, M., Lifschitz, V.: The stable semantics for logic programs. ICLP88 (Eds. Kowalski, R., Bowen). (1988) 1070–1080.
5. Hinman, P.G.: Recursion-Theoretic Hierarchies. Springer-Verlag (1978)
6. Soare, R.E.: Recursively Enumerable Sets and Degrees. Springer-Verlag (1987)

A Logical Approach to Abstract Algebra

Thierry Coquand

Institutionen för Datavetenskap,
Chalmers Tekniska Högskola,
Göteborg, Sweden
coquand@cs.chalmers.se

Abstract. Recent work in constructive mathematics show that Hilbert's program works for a large part of abstract algebra. Furthermore the arguments we get are not only elementary but also mathematically simpler. We present an example where the simplification was significant enough to suggest an improved version of a classical theorem. For this we use a general method to transform some logically complex first-order formulae in a geometrical form which may be interesting in itself.

1 Introduction

The purpose of this extended abstract is to survey recent works in constructive algebra [4, 5, 6, 7, 8, 9] from the point of view of mathematical logic. The goal is to illustrate the relevance of simple logical considerations in the development of constructive algebra.

We analyse the logical complexity of statements and proofs in abstract algebra. Two notions of formulae, being geometrical and being first-order, will play an important role. The two notions are in general incomparable. Both notions have a fundamental "analytical" property: if a statement is formulated in first-order logic and has a proof, then we know that it can be proved in a first-order way. Similarly, if a geometrical statement holds, it has a constructive proof which has a particularly simple tree form [2, 7, 9].

We present first some basic examples in algebra which are directly formulated with the required logical complexity: the first one is an implication between equational statements, and the second one is geometric and first-order. We end with a more elaborate example, that was a mathematical conjecture and where a first-order formulation is not obvious. We can transform further it to a geometrical problem applying a general method which may be interesting in itself. Knowing a priori that we had to look for an "analytical" proof involving only simple algebraic manipulations helps then in finding a proof. One main theme, which is also present in the work [10] is the elimination of Noetherianity hypotheses to get simple first-order statements.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 86–95, 2005.

2 Logical Complexity

The theory of commutative rings is a first-order theory, and actually even equational. We need 3 symbols of functions $+, \times, -$, we often write ab for $a \times b$, two constants $0, 1$ and the axioms are

$$x + (-x) = 0, \quad x + (y + z) = (x + y) + z, \quad x + y = y + x, \quad x + 0 = x$$

$$x1 = x, \quad xy = yx, \quad x(yz) = (xy)z, \quad x(y + z) = xy + xz$$

Some elementary concepts and theorems of commutative abstract algebra can be formulated in this language. For instance the notion of *integral* ring can be represented by the formula

$$xy = 0 \rightarrow [x = 0 \vee y = 0]$$

By the completness theorem of first-order logic, we know that if a theorem can be formulated as a first-order theorem, it has a proof in first-order logic. If it is furthermore formulated equationally, we even know, by Birkhoff's completness theorem, that there is a purely equational proof. This can be seen as a partial realisation of Hilbert's program.

If we take however a basic book in abstract algebra such as Atiyah-Macdonald or Matsumura [1, 18] we discover that even basic theorems are not formulated in a first-order way because of the introduction of abstract notions. Such abstract notions are

1. arbitrary ideals of the rings, that are defined as subsets, which is not a first-order notion
2. *prime* or *maximal* ideals, whose existence relies usually on Zorn's lemma,
3. Noetherianity

These notions have different levels of non effectivity. The notion of Noetherianity can be captured by a generalised inductive definition [14], but we leave then first-order logic. The notion of prime ideals seems even more ineffective: it is rather simple to build effective rings, that is with computable operations and decidable equality, with an empty set of prime ideals constructively.

Furthermore a notion such as "being nilpotent" cannot be expressed in a first-order way since it involves an infinite disjunction (or alternatively, an existential quantification over natural numbers).

G. Wraith [21] points us the relevance of the notion of *geometric formula* for constructive algebra. One defines first the notion of *positive formulae*: a positive formula is one formula of the language of rings built using positive atomic formula (equality between two terms) and the connectives $\vee, \wedge$. Special cases are the empty disjunction which is the false formula $\perp$, and the empty conjunction which is the true formula T. We allow also existential quantification and existential quantification over natural numbers[1]. A geometrical formula is an implication

[1] Sometimes, the notion of "arbitrary" infinite disjunction is allowed, but we shall not need this generality here.

between two positive formulae. Notice that, as special cases, any positive formula is geometrical, and the negation of a positive formula is geometrical.

The notion for $a \in R$ to be nilpotent is not first-order but it can be expressed as a positive formula: a is nilpotent if and only if $a^n = 0$ for some $n \in \mathbb{N}$. Another typical example [21] of notion expressed by a geometrical formula if the notion of *flat* module M over a ring R. It says that if we have a relation $PX = 0$ where P is a line vector in coefficient in R and X a column vector with elements in M then we can find a rectangular matrix Q and a vector Y such that $QY = X$ and $PQ = 0$. Since we don't say anything about the dimension of Q this statement involves implicitely an infinite disjunction over natural numbers. Thus the notion of flat module is not first-order but geometric.

As stressed by G. Wraith the importance of geometric formula comes from *Barr's theorem*: if a geometric formula is provable classically from axioms that are geometric, then there is a constructive proof. Furthermore this proof has a simple branching tree form, of a *dynamical* proof [7, 2, 9].

Both the completeness theorem and Barr's theorem however are purely heuristic results from a constructive point of view. Indeed, they are both proved using non constructive means, and do not give algorithms to transform a non effective proof to an effective one. In practice however, in all examples analysed so far, it has been possible to extract effective arguments from the ideas present in the non effective proofs.

3 Some Basic Examples

In this section, we provide two elementary examples where Barr's theorem can be invoked. They are directly expressed with the appropriate logical complexity. In the next section, we present more elaborate examples where some works has to be done in order to get the right logical complexity. For the first example of this section, Birkhoff's completness theorem for equational logic is enough. Both examples appear at the beginning of [18].

3.1 Dimension over Rings

The following result is usually proved using maximal ideals [18].

Theorem 1. *If $n < m$ and $f : R^n \to R^m$ is surjective then $1 = 0 \in R$.*

What is the logical complexity of this statement? If we fix n and m, let say $n = 2$ and $m = 3$ the statement becomes an implication from a conjunction of equalities to $1 = 0$. More precisely, the hypothesis is that we have a 2×3 matrix P and a 3×2 matrix Q such that $PQ = I$. That is we have 9 equations of the form

$$p_{i1}q_{1j} + p_{i2}q_{2j} = \delta_{ij}$$

with $i, j = 1, 2, 3$.

The classical proof uses existence of maximal ideals: if R is not trivial it has a maximal ideal $\mathfrak{m}$. If $k = R/\mathfrak{m}$ we have a surjective map from k^n to k^m and this is a contradiction.

It is possible to transform this argument in a direct equational reasoning. Here we simply remark that the concrete statement means that 1 belongs to the ideal generated by $p_{i1}q_{1j} + p_{i2}q_{2j} - \delta_{ij}$, seeing p_{ik}, q_{kj} as indeterminates, and this can be certified with a simple algebraic identity.

3.2 Projective Modules over Local Rings

We analyse the following elementary theorem.

Theorem 2. *If M is a finitely generated projective module over a local ring R then M is free.*

The concrete formulation of this theorem [17] is the following.

Theorem 3. *If F is an idempotent square matrix over a local ring R then F is similar to a matrix of the form*

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}$$

The statement of this theorem, for a fixed size of F is first-order and geometric.

We have a first-order classical derivation, that we can transform by proof-theoretic methods to a constructive first-order derivation.

Proof. (Classical) The fact that R is local is expressed by the geometrical formula

$$\mathsf{Inv}(x) \vee \mathsf{Inv}(1 - x)$$

where $\mathsf{Inv}(a)$ means $\exists y.ay = 1$. We have

$$\mathsf{Inv}(xy) \leftrightarrow (\mathsf{Inv}(x) \wedge \mathsf{Inv}(y))$$

It follows that we have, for all x and y

$$\mathsf{Inv}(x) \vee \mathsf{Inv}(1 - xy)$$

Classically it is possible to derive from this

$$\mathsf{Inv}(x) \vee \forall y.\mathsf{Inv}(1 - xy)$$

that is any element x is invertible or belongs to the Jacobson radical of R, which is the only maximal ideal of R[2]. Constructively, this inference is not justified.

[2] Classically the *Jacobson radical* of R is the intersection of all maximal ideals of R. It is easy to see that this is the same as the set of elements x such that all $1 - xy$ are invertible, and this is a first-order characterisation of the Jacobson radical.

If J is the Jacobson radical of R, we have proved classically

$$\mathsf{Inv}(x) \vee x \in J$$

We can use this to prove the theorem: we get that the matrix F is similar to a matrix of the form

$$\begin{pmatrix} I & L \\ 0 & G \end{pmatrix}$$

where all the elements of G are in J. Since F is idempotent, so is G and we get $G(I - G) = 0$. Since all elements of G are in J, the determinant of $I - G$ is invertible and hence $G = 0$. The matrix

$$\begin{pmatrix} I & L \\ 0 & 0 \end{pmatrix}$$

is then similar to the matrix

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}$$

since it is PFP^{-1}, where P is the matrix

$$\begin{pmatrix} I & L \\ 0 & I \end{pmatrix}$$

This reasoning is (for a fixed size matrix) done in first-order logic. But it uses the classical validity of $\mathsf{Inv}(x) \vee x \in J$. It is known from proof theory that we can transform this argument to a constructive one, using for instance negative translations or cut-elimination.

It is interesting that the argument we get in this way, when simplified gives an argument which is mathematically simpler than the classical argument.

Proof. (Constructive) For any square matrix H either the determinant of $I - H$ is invertible or there is an element in H which is invertible. If H is idempotent this implies, since $H(I - H) = 0$, that $H = 0$ or there is an element in H which is invertible. Using this remark, we see directly that the matrix F is similar to a matrix of the form

$$\begin{pmatrix} I & L \\ 0 & 0 \end{pmatrix}$$

and we can conclude as above.

Notice that this last argument can be read as an algorithm: given the matrix F, and a procedure to decide whether x or $1 - x$ is invertible, it computes an invertible matrix Q such that QFQ^{-1} has the required form.

4 Serre's Splitting-Off Theorem

4.1 Classical Formulation

The example we are going to present has his origin in a paper of Serre [19] from 1958. It is a purely algebraic theorem, but it has a geometrical intuition. (The

geometrical statement is roughly that if we have a vector fibre bundle over a space of finite dimension, and each fiber have a large enough dimension, then we can find a non vanishing section.) We give first the classical formulation, where both hypotheses and conclusions have a non elementary form, and then a version where the conclusion is first-order.

We assume R to be a noetherian ring, and we let $\mathsf{Max}(R)$ to be the space of maximal ideals with the topology induced from the Zariski topology. We assume that the dimension of $\mathsf{Max}(R)$ is finite and $< n$ (that is there is no proper chains of irreducible closed sets of length n). For instance, if R is a local ring, then $\mathsf{Max}(R)$ is a singleton and we can take $n = 1$.

If M is a finitely generated module over R and x a maximal ideal of R, then M/xM is a finite dimensional vector space over $k = R/x$ and we let $r_x(M)$ be its dimension. (Intuitively, M represents the module of global section of a vector bundle over the space $\mathsf{Max}(R)$ and $r_x(M)$ is the dimension of the fiber at the point $x \in \mathsf{Max}(R)$.) If $s \in M$ it is suggestive to write $s(x)$ the equivalence class of s in $M(x) = M/xM$. Intuitively $s(x)$ is a continuous family of sections in the fibers $M(x)$.

Theorem 4 (Serre, 1958). *If M is a finitely generated projective module over R such that $n \leq r_x$ for all maximal ideals x of R. There exists $s \in M$ such that $s(x) \neq 0$ for all $x \in \mathsf{Max}(R)$.*

The first step is to give a more concrete formulation of this result. We give only the end result [8, 17]. If F is a matrix over R we let $\Delta_k(F)$ be the ideal generated by all minors of F of order k. We say that a vector of elements of R is *unimodular* if and only if 1 belongs to the ideal generated by these elements.

Theorem 5 (Serre, 1958, concrete version). *If F is an idempotent matrix over R and $\Delta_n(F) = 1$ then there exists a linear combination of the columns of F which is unimodular.*

Interestingly, in this form, the theorem can then be seen as a special case of Swan's theorem [20], theorem conjectured by Serre. The generalisation of these theorems to the non Noetherian case has been first established in [8], by analysing the paper [13] using the techniques that are presented in this note.

Notice that the conclusion of this theorem is expressed in first-order logic, and even in a positive way. The hypothesis however is non elementary: we suppose both that R is Noetherian and we have an hypothesis on the dimension of $\mathsf{Max}(R)$. It was conjectured that the theorem holds without the hypothesis that R is Noetherian, and this is the statement that we want to analyse. It is left to express the hypothesis of the theorem $\mathsf{dim}\ (\mathsf{Max}(R)) < n$ in a first-order way.

4.2 Geometric Formulation of Krull Dimension

The first step is to give an elementary formulation of the notion of Krull dimension. It is not so easy a priori since the usual definition is in term of chain of prime ideals: a ring R is of Krull dimension $< n$ if and only if there is no proper

chain of prime ideals of length n. An elementary definition is presented in [5]. We introduce first the notion of *boundary* of an element of a ring: the boundary N_a of a is the ideal generated by a and the elements x such that ax is nilpotent, that is belong to the nilradical of R. Classically the nilradical is the intersection of all prime ideals of R. We define then inductively $\mathsf{Kdim}\ R < n$: for $n = 0$ it means that $1 = 0 \in R$ and for $n > 0$ it means that we have $\mathsf{Kdim}\ (R/N_a) < n-1$ for all $a \in R$.

For each n, we get a formulation of $\mathsf{Kdim}\ R < n$ which is positive, but *not* first-order. For instance $\mathsf{Kdim}\ R < 1$ is expressed by the formula

$$\forall x.\exists a.\ \bigvee_{k\in\mathbb{N}} x^k(1-ax) = 0$$

while $\mathsf{Kdim}\ R < 2$ is expressed by

$$\forall x, y.\exists a, b.\ \bigvee_{k\in\mathbb{N}} y^k(x^k(1-ax) - by)) = 0$$

We can now expressed the concrete form of the non Noetherian version of Forster's theorem (that motivated Swan's theorem in the Noetherian case).

Theorem 6 (Heitmann, 1984, concrete version). *If* $\mathsf{Kdim}\ R < n$ *and if* F *is a rectangular matrix over* R *such that* $\Delta_n(F) = 1$ *then there exists a linear combination of the columns of* F *which is unimodular.*

The formulation is now geometric (but not first-order). The hypothesis is a positive statement (of the form $\forall\exists$ but the existential quantification is over natural numbers) and the conclusion is purely existential. We expect it to have a constructive proof, of a very simple nature furthermore. In this case, it is enough to extract this direct proof from the argument in [13]. This is carried out in [8].

4.3 A New Notion of Dimension

We present now a notion of dimension, introduced in [8] and which appears implicitely in [13]. This notion is finer than the notion of Krull dimension: we always have $\mathsf{Hdim}(R) \leq \mathsf{Kdim}\ R$. Interestingly $\mathsf{Hdim}(R) \leq n$ can be expressed by a first-order formula, but the logical complexity of this formula increases with n, contrary to $\mathsf{Kdim}\ R \leq n$ which stays a positive formula (and thus, proof-theoretically can be considered to be Π^0_2) for all n.

We get this definition by changing the nilradical in the definition of Krull dimension by the Jacobson radical J which is classically the intersection of all maximal ideals, but can be defined in a first-order way as the set of elements a such that $1 - ax$ is invertible for all $x \in R$. We introduce then a new notion of *boundary* of an element of a ring: the boundary J_a of a is the ideal generated by a and the elements x such that ax is in the Jacobson radical of R. We define then inductively $\mathsf{Hdim}(R) < n$: for $n = 0$ it means that $1 = 0 \in R$ and for $n > 0$ it means that we have $\mathsf{Hdim}(R/N_a) < n - 1$ for all $a \in R$.

What is the logical complexity of $\mathsf{Hdim}(R) < n$? We introduce the predicate $\mathsf{Inv}(x)$ which means $\exists y.xy = 1$ and the predicate $J(x)$ which means $\forall y.\mathsf{Inv}(1-xy)$. For $n = 1$ we get that $\mathsf{Hdim}(R) < n$ means

$$\forall x.\exists a.J(x(1-ax))$$

which is a prenex formula with two alternations of quantifiers. For $n = 2$ we get an even more complex formula, and the logical complexity increases with n.

In this way we get a way to state a plausible non Noetherian version of Swan's theorem in a purely first-order way, as an implication

$$\mathsf{Hdim}(R) < n \rightarrow \Delta_n(F) = 1 \rightarrow \exists X, Y.1 = XFY$$

where X is a vector line and Y a vector column. For a given n and a given size of F this is a first-order statement.

The form of the statement for $\mathsf{Hdim}(R) < n$ is particular since it is a purely *prenex* formula. It is then possible to conclude, by using general proof-theoretic arguments that, if we have a first-order classical proof, then we also have an intuitionistic proof. From proof theory, one can use sharpened Gentzen Hauptsatz [12], or a negative translation.

Alternatively, we can find another statement equivalent to this which can be expressed in a geometrical and first-order way. We illustrate the idea only for $n = 1$. We have see that $\mathsf{Hdim}(R) < 1$ is equivalent to

$$\forall x.\exists a.\forall y.\exists b.1 = b(1 - yx(1 - ax))$$

We replace this hypothesis by the following geometrical theory $\phi(I, J)$

$$\forall x.\exists a.J(x(1-ax)), \quad \forall x, y.J(x) \rightarrow I(1-xy), \quad \forall x.I(x) \rightarrow \exists y.1 = xy$$

where J, I are new predicate symbols.

It is quite easy to see that we have, for any statement ψ which does not contain I, J

$$(\mathsf{Hdim}(R) < 1) \rightarrow \psi$$

if and only if

$$\phi(I, J) \rightarrow \psi$$

Indeed, we have clearly $\phi(I, J) \rightarrow \mathsf{Hdim}(R) < 1$, so that $(\mathsf{Hdim}(R) < 1) \rightarrow \psi$ implies $\phi(I, J) \rightarrow \psi$. Conversely if $\phi(I, J) \rightarrow \psi$, since ψ does not contain I, J we can instantiate $I(x)$ to the predicate $\exists y.1 = xy$ and $J(x)$ to the predicate $\forall y.I(1 - xy)$ and we get $\mathsf{Hdim}(R) < 1 \rightarrow \psi$.

It can be checked that if R is Noetherian then $\mathsf{Hdim}(R) < n$ if and only if $\mathsf{dim}\ (\mathsf{Max}(R)) < n$. A possible generalisation of Serre's theorem can thus be formulated as follows.

Theorem 7 ([8], 2004). *If* $\mathsf{Hdim}(R) < n$ *and if* F *is a rectangular matrix over* R *such that* $\Delta_n(F) = 1$ *then there exists a linear combination of the columns of* F *which is unimodular.*

The formulation of this theorem is now purely first-order and geometric, in a geometrical theory which has a specially simple form (no branching). If it holds, it has a purely elementary proof, and knowing this helps in finding a proof [8].

We can read the proof presented in [8] as an algorithm which produces an unimodular column. To analyse the complexity of this algorithm seems to be an interesting problem.

It is remarkable that the Noetherian hypothesis could be avoided in this case. There are examples in algebra, like Krull's Principal Ideal theorem, or the regular element property where the Noetherian hypothesis is necessary.

References

1. M.F. Atiyah and I.G. Macdonald. *Introduction to Commutative Algebra.* Addison Wesley, 1969.
2. M. Bezem and Th. Coquand. Newman's lemma—a case study in proof automation and geometric logic. Bull. Eur. Assoc. Theor. Comput. Sci. EATCS No. 79 (2003), 86–100.
3. A. Blass. Topoi and computation, *Bulletin of the EATCS* **36**, October 1988, pp. 57–65.
4. Th. Coquand and H. Lombardi. Hidden constructions in abstract algebra (3) Krull dimension. in Commutative ring theory and applicatoins. Eds: Fontana, M, LNPAM 131.
5. Th. Coquand, H. Lombardi and M.F. Roy. Une caractérisation élémentaire de la dimension de Krull. Preprint 2003.
6. Th. Coquand. Sur un théorème de Kronecker concernant les variétés algébriques C. R. Acad. Sci. Paris, Ser. I 338 (2004), Pages 291-294
7. Th. Coquand. A Completness Proof for Geometrical Logic to appear, 2005.
8. Th. Coquand, H. Lombardi, C. Quitte. Generating non-Noetherian modules constructively. Manuscripta mathematica, 115, 513-520 (2004)
9. M. Coste, H. Lombardi, and M.F. Roy. Dynamical methods in algebra: effective Nullstellensätze, *Annals of Pure and Applied Logic* **111**(3):203–256, 2001.
10. L. Ducos, H. Lombardi, C. Quitté and M. Salou. Théorie algorithmique des anneaux arithmétiques, de Prüfer et de Dedekind. Journal of Algebra 281, (2004), 604-650.
11. O. Forster. Über die Anzahl der Erzeugenden eines Ideals in einem Noetherschen Ring. Math. Z. 84 1964 80–87.
12. G. Gentzen. *Collected Works.* Edited by Szabo, North-Holland, 1969.
13. R. Heitmann. Generating non-Noetherian modules efficiently. Michigan Math. J. 31 (1984), no. 2, 167–180.
14. C. Jacobsson and C. Löfwall. Standard bases for general coefficient rings and a new constructive proof of Hilbert's basis theorem. J. Symbolic Comput. 12 (1991), no. 3, 337–371.
15. A. Joyal. Le théorème de Chevalley-Tarski. *Cahiers de Topologie et Géométrie Différentielle* 16, 256–258 (1975).
16. L. Kronecker. Grundzüge einer arithmetischen Theorie der algebraischen Grössen. *J. reine angew. Math.* 92, 1-123 (1882). Reprinted in *Leopold Kronecker's Werke*, II, 237–387.
17. H. Lombardi and C. Quitté. *Modules projectifs de type fini.* To appear.

18. H. Matsumura. *Commutative ring theory.* Translated from the Japanese by M. Reid. Cambridge Studies in Advanced Mathematics, 8. Cambridge University Press, Cambridge, 1986.
19. J.P. Serre. Modules projectifs et espaces fibrés à fibre vectorielle. Séminaire P. Dubreil, Année 1957/1958.
20. R.G. Swan. The Number of Generators of a Module. Math. Z. 102 (1967), 318-322.
21. G. Wraith. Intuitionistic algebra: some recent developments in topos theory. Proceedings of the International Congress of Mathematicians (Helsinki, 1978), pp. 331–337, Acad. Sci. Fennica, Helsinki, 1980.

Schnorr Dimension

Rodney Downey[1], Wolfgang Merkle[2], and Jan Reimann[2]

[1] School of Mathematics, Statistics and Computer Science,
Victoria University P. O. Box 600,
Wellington, New Zealand
[2] Arbeitsgruppe Mathematische Logik und Theoretische Informatik,
Institut für Informatik,
Fakultät für Mathematik und Informatik,
Ruprecht-Karls-Universität Heidelberg,
Im Neuenheimer Feld 294,
D-69120 Heidelberg, Germany

Abstract. Following Lutz's approach to effective (constructive) dimension, we define a notion of dimension for individual sequences based on Schnorr's concept(s) of randomness. In contrast to computable randomness and Schnorr randomness, the dimension concepts defined via computable martingales and Schnorr tests coincide. Furthermore, we give a machine characterization of Schnorr dimension, based on prefix free machines whose domain has computable measure. Finally, we show that there exist computably enumerable sets which are Schnorr irregular: while every c.e. set has Schnorr Hausdorff dimension 0 there are c.e. sets of Schnorr packing dimension 1, a property impossible in the case of effective (constructive) dimension, due to Barzdin's Theorem.

1 Introduction

Martin-Löf's concept of individual random sequences was recently generalized by Lutz [10, 11], who introduced an effective notion of Hausdorff dimension. As (classical) Hausdorff dimension can be seen as a refinement of Lebesgue measure on $2^{\mathbb{N}}$, in the sense that it further distinguishes between classes of measure 0, the effective Hausdorff dimension of an individual sequence can be interpreted as a *degree of randomness* of the sequence. This viewpoint is supported by a series of results due to Ryabko [15, 16], Staiger [21, 20], Cai and Hartmanis [3], and Mayordomo [12], which establish that the effective Hausdorff dimension of a sequence equals its lower asymptotic Kolmogorov complexity (plain or prefix-free).

Criticizing Martin-Löf's approach to randomness as not being truly algorithmic, Schnorr [18] presented two alternative randomness concepts, one based one computable martingales, nowadays referred to as *computable randomness*, the other based on stricter effectivity requirements for Martin-Löf tests. This concept is known as *Schnorr randomness*.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 96–105, 2005.

In this paper we will generalize and extend Schnorr's randomness concepts to Hausdorff dimension. Like in the case of randomness, the approach suffers from some technical difficulties like the absence of a universal test/martingale. We will see that for dimension, Schnorr's two approaches coincide, in contrast to Schnorr randomness and computable randomness. Furthermore, it turns out that, with respect to Schnorr dimension, computably enumerable sets can expose a complex behavior, to some extent. Namely, we will show that there are c.e. sets of high Schnorr packing dimension, which is impossible in the effective case, due to a result by Barzdin' [2]. On the other hand, we prove that the Schnorr Hausdorff dimension of the characteristic sequence of a c.e. set is 0. Thus, the class of computably enumerable sets contains *irregular* sequences – sequences for which Hausdorff and packing dimension do not coincide.

The paper is structured is as follows. In Section 2 we give a short introduction to the classical theory of Hausdorff measures and dimension, as well as packing dimension. In Section 3 we will define algorithmic variants of these concepts based on Schnorr's approach to randomness. In Section 4 we prove that the approach to Schnorr dimension via coverings and via computable martingales coincide, in contrast to Schnorr randomness and computable randomness. In Section 5 we derive a machine characterization of Schnorr Hausdorff and packing dimension. Finally, in Section 6, we study the Schnorr dimension of computably enumerable sets. The main result here will be that on those sets Schnorr Hausdorff dimension and Schnorr packing dimension can differ as largely as possible.

We will use fairly standard notation. $2^{\mathbb{N}}$ will denote the set of infinite binary sequences. Sequences will be denoted by upper case letters like A, B, C, or X, Y, Z. We will refer to the nth bit ($n \geq 0$) in a sequence B by either B_n or $B(n)$, i.e. $B = B_0B_1B_2 \ldots = B(0)\,B(1)\,B(2)\ldots$.

Strings, i.e. finite sequences of 0s and 1s will be denoted by lower case letters from the end of the alphabet, u, v, w, x, y, z along with some lower case Greek letters like σ and τ. $\{0,1\}^*$ will denote the set of all strings. The *initial segment of length* n, $A\upharpoonright_n$, of a sequence A is the string of length n corresponding to the first n bits of A.

Given two strings v, w, v is called a *prefix* of w, $v \sqsubseteq w$ for short, if there exists a string x such that $vx = w$, where vx is the concatenation of v and x. Obviously, this relation can be extended to hold between strings and sequences as well. A set of strings is called *prefix free* if all its elements are pairwise incomparable.

Initial segments induce a standard topology on $2^{\mathbb{N}}$. The basis of the topology is formed by the *basic open cylinders* (or just *cylinders*, for short). Given a string $w = w_0 \ldots w_{n-1}$ of length n, these are defined as $[w] = \{A \in 2^{\mathbb{N}} : A\upharpoonright_n = w\}$. For $C \subseteq \{0,1\}^*$, we define $[C] = \bigcup_{w \in C}[w]$.

Throughout the paper we assume familiarity with the basic concepts of computability theory such as Turing machines, computably enumerable sets, computable and left-computable (c.e.) reals. Due to space consideration, formal proofs of the results are omitted. (Some ideas are sketched.)

2 Hausdorff Measures and Dimension

The basic idea behind Hausdorff dimension is to generalize the process of measuring a set by approximating (covering) it with sets whose measure is already known. Especially, the size of the sets used in the measurement process will be manipulated by certain transformations, thus making it harder (or easier) to approximate a set with a covering of small accumulated measure. This gives rise to the notion of *Hausdorff measures*.

Definition 1. *Let $\mathcal{X} \subseteq 2^{\mathbb{N}}$. Given $\delta > 0$ and a real number $s \geq 0$, define*

$$\mathcal{H}^s_\delta(\mathcal{X}) = \inf \left\{ \sum_{w \in C} 2^{-|w|s} : \ (\forall w \in C)[2^{-|w|} \leq \delta] \ \wedge \ \mathcal{X} \subseteq [C] \right\}$$

*The s-*dimensional Hausdorff measure *of $\mathcal{X}$ is defined as*

$$\mathcal{H}^s(\mathcal{X}) = \lim_{\delta \to 0} \mathcal{H}^s_\delta(\mathcal{X}).$$

Note that $\mathcal{H}^s(\mathcal{X})$ is well defined, since, as δ decreases, there are fewer δ-covers available, hence $\mathcal{H}^s_\delta$ is non-decreasing. However, the value may be infinite. For $s = 1$, one obtains Lebesgue measure on $2^{\mathbb{N}}$.

The outer measures $\mathcal{H}^s$ have an important property.

Proposition 2. *Let $\mathcal{X} \subseteq 2^{\mathbb{N}}$. If, for some $s \geq 0$, $\mathcal{H}^s(\mathcal{X}) < \infty$, then $\mathcal{H}^t(\mathcal{X}) = 0$ for all $t > s$.*

This means that there can exist only one point $s \geq 0$ where a given class might have finite positive s-dimensional Hausdorff measure. This point is the *Hausdorff dimension* of the class.

Definition 3. *For a class $\mathcal{X} \subseteq 2^{\mathbb{N}}$, define the* Hausdorff dimension *of $\mathcal{X}$ as*

$$\dim_{\mathrm{H}}(\mathcal{X}) = \inf\{s \geq 0 : \ \mathcal{H}^s(\mathcal{X}) = 0\}.$$

For more on Hausdorff measures and dimension refer to the book by Falconer [7]. A presentation of Hausdorff measures and dimension in $2^{\mathbb{N}}$ can be found in [13].

2.1 Packing Dimension

We say that a prefix free set $P \subseteq \{0,1\}^*$ is a *packing* for $\mathcal{X} \subseteq 2^{\mathbb{N}}$, if for every $\sigma \in P$, there is some $A \in \mathcal{X}$ such that $\sigma \sqsubset A$.

Given $s \geq 0$, $\delta > 0$, let

$$\mathcal{P}^s_\delta(\mathcal{X}) = \sup \left\{ \sum_{w \in P} 2^{-|w|s} : \ P \text{ packing for } \mathcal{X} \text{ and } (\forall w \in P)[2^{-|w|} \leq \delta \right\}. \quad (1)$$

$\mathcal{P}^s_\delta(\mathcal{X})$ decreases with δ, so the limit $\mathcal{P}^s_0(\mathcal{X}) = \lim_{\delta\to 0} \mathcal{P}^s_\delta(\mathcal{X})$ exists. Finally, define

$$\mathcal{P}^s(\mathcal{X}) = \inf\left\{ \sum \mathcal{P}^s_0(\mathcal{X}_i) : \ \mathcal{X} \subseteq \bigcup_{i\in\mathbb{N}} \mathcal{X}_i \right\}. \tag{2}$$

(The infimum is taken over arbitrary countable covers of $\mathcal{X}$.) $\mathcal{P}^s$ is called, in correspondence to Hausdorff measures, the *s-dimensional packing measure* on $2^{\mathbb{N}}$. Packing measures were introduced by Tricot [24] and Taylor and Tricot [23]. They can be seen as a dual concept to Hausdorff measures, and behave in many ways similar to them. In particular, one may define *packing dimension* in the same way as Hausdorff dimension.

Definition 4. *The* packing dimension *of a class $\mathcal{X} \subseteq 2^{\mathbb{N}}$ is defined as*

$$\dim_{\mathrm{P}} \mathcal{X} = \inf\{s : \ \mathcal{P}^s(\mathcal{X}) = 0\} = \sup\{s : \ \mathcal{P}^s(\mathcal{X}) = \infty\}. \tag{3}$$

Again, we refer to Falconer's book [7] for details on packing measures and dimension.

2.2 Martingales

It is possible do characterize Hausdorff and packing dimension via *martingales*, too. Martingales have become a fundamental tool in probability theory. In Cantor space $2^{\mathbb{N}}$, they can be described very conveniently.

Definition 5. *A* martingale *on $2^{\mathbb{N}}$ is a function $d : \{0,1\}^* \to [0,\infty)$ which satisfies*

$$d(w) = \frac{d(w01) + d(w1)}{2} \quad \textit{for all } w \in \{0,1\}^*.$$

Martingales can be interpreted as capital functions of an accordant betting strategy, when applied to a binary sequence. The value $d(w)$ reflects the player's capital after bits $w(0), \ldots, w(|w|-1)$ have been revealed to him.

Definition 6. *Let $g : \mathbb{N} \to [0,\infty)$ be a positive, unbounded function. A martingale is g-successful (or* g-succeeds*) on a sequence $B \in 2^{\mathbb{N}}$ if*

$$d(B\upharpoonright_n) \geq g(n) \ \textit{ for infinitely many } n.$$

d is strongly g-successful *(or* g-succeeds strongly*) on a sequence $B \in 2^{\mathbb{N}}$ if*

$$d(B\upharpoonright_n) \geq g(n) \ \textit{ for all but finitely many } n. \tag{4}$$

It turns out that, in terms of Hausdorff dimension, the relation between $\mathcal{H}^s$-nullsets and $2^{(1-s)n}$-successful martingales is very close.

Theorem 7. *Let $\mathcal{X} \subseteq 2^{\mathbb{N}}$. Then it holds that*

$$\dim_{\mathrm{H}} \mathcal{X} = \inf\{s : \exists \textit{ martingale } d \ 2^{(1-s)n}\textit{-successful on all } B \in \mathcal{X}\}. \tag{5}$$

$$\dim_{\mathrm{P}} \mathcal{X} = \inf\{s : \exists \textit{ martingale } d \textit{ strongly } 2^{(1-s)n}\textit{-successful on all } B \in \mathcal{X}\} \tag{6}$$

In the form presented here, equation (5) was first proven by Lutz [9]. However, close connections between Hausdorff dimension and winning conditions on martingales have been observed by Ryabko [17] and Staiger [22]. Equation (6) is due to Athreya, Lutz, Hitchcock, and Mayordomo [1].

Note that, if a martingale $2^{(1-s)n}$-succeeds on a sequence A, for any $t > s$ it will hold that

$$\limsup_{n\to\infty} \frac{d(A\upharpoonright_n)}{2^{(1-t)n}} = \infty. \tag{7}$$

So, when it comes to dimension, we will, if convenient, use (7) and the original definition interchangeably. Furthermore, a martingale which satisfies (7) for $t = 1$ is simply called *successful on* A.

3 Schnorr Null Sets and Schnorr Dimension

We now define a notion of dimension based on Schnorr's approach to randomness.

Definition 8. *Let $s \in [0,1]$ be a rational number.*

(a) A Schnorr s-test *is a computable sequence $(S_n)_{n\in\mathbb{N}}$ of c.e. sets of finite strings such that, for all n, $\sum_{w\in S_n} 2^{-|w|s} \le 2^{-n}$, and $\sum_{w\in S_n} 2^{-|w|s}$ is a uniformly computable real number.*

(b) A class $\mathcal{A} \subseteq 2^{\mathbb{N}}$ is Schnorr s-null *if there exists a Schnorr s-test (S_n) such that $\mathcal{A} \subseteq \bigcap_{n\in\mathbb{N}}[S_n]$.*

The *Schnorr random sequences* are those which are (as a singleton class in $2^{\mathbb{N}}$) not Schnorr 1-null.

Downey and Griffiths [4] observe that, by adding elements, one can replace any Schnorr 1-test by an equivalent one (i.e., one defining the same Schnorr nullsets) where each level of the test has measure exactly 2^{-n}. We can apply the same argument in the case of arbitrary rational s, and hence we may, if appropriate, assume that $\sum_{w\in S_n} 2^{-|w|s} = 2^{-n}$, for all n.

Note further that, for rational s, each set S_n in a Schnorr s-test is actually computable, since to determine whether $w \in S_n$ it suffices to enumerate S_n until the accumulated sum given by $\sum 2^{-|v|s}$ exceeds $2^{-n} - 2^{|w|s}$ (assuming the measure of the n-th level of the test is in fact 2^{-n}). If w has not been enumerated so far, it cannot be in S_n. (Observe, too, that the converse does not hold.)

One can describe Schnorr s-nullsets also in terms of *Solovay tests*. Solovay tests were introduced by Solovay [19] and allowed for a characterization of Martin-Löf nullsets via a single test set, instead of a uniformly computable sequence of test sets.

Definition 9. *Let $s \in [0,1]$ be rational.*

(a) An Solovay s-test *is a c.e. set $C \subseteq \{0,1\}^*$ such that $\sum_{w\in C} 2^{-|w|s} \le 1$.*

(b) An Solovay s-test is total *if $\sum_{w\in C} 2^{-|w|s}$ is a computable real number.*

(c) An Solovay s-test C covers *a sequence $A \in 2^{\mathbb{N}}$ if it contains infinitely many initial segments of A. In this case we also say that A* fails *the test C.*

Theorem 10. *For any rational $s \in [0,1]$, a class $\mathcal{X} \subseteq 2^{\mathbb{N}}$ is Schnorr s-null if and only if there is a total Solovay s-test which covers every sequence $A \in \mathcal{X}$.*

3.1 Schnorr Dimension

Like in the classical case, each class has a critical value a critical value with respect to Schnorr s-measure.

Proposition 11. *Let $\mathcal{X} \subseteq 2^{\mathbb{N}}$. For any rational $s \geq 0$, if $\mathcal{X}$ is Schnorr s-null then it is also Schnorr t-null for any rational $t \geq s$.*

This follows from the fact that every Schnorr s-test is also a t-test. The definition of Schnorr Hausdorff dimension can now be given in a straightforward way.

Definition 12. *The* Schnorr Hausdorff dimension *of a class $\mathcal{X} \subseteq 2^{\mathbb{N}}$ is defined as*

$$\dim_{\mathrm{H}}^{\mathrm{S}}(\mathcal{X}) = \inf\{s \geq 0 : \mathcal{X} \text{ is Schnorr } s\text{-null}\}.$$

For a sequence $A \in 2^{\mathbb{N}}$, we write $\dim_{\mathrm{H}}^{\mathrm{S}} A$ for $\dim_{\mathrm{H}}^{\mathrm{S}}\{A\}$ and refer to $\dim_{\mathrm{H}}^{\mathrm{S}} A$ as the Schnorr Hausdorff dimension of A.

3.2 Schnorr Packing Dimension

Due to the more involved definition of packing dimension, it is not immediately clear how to define a Schnorr-type version of packing dimension. However, we will see in the next section that Schnorr dimension allows an elegant characterization in terms of martingales, building on Theorem 7. This will also make it possible to define a Schnorr version of packing dimension.

4 Schnorr Dimension and Martingales

In view of his unpredictability paradigm for algorithmic randomness, Schnorr [18] suggested a notion of randomness based on *computable* martingales. According to this notion, nowadays referred to as *computable randomness*, a sequence is computably random if no computable martingale succeeds on it.

Schnorr [18] himself proved that a sequence is Martin-Löf random if and only if some *left-computable* martingale succeeds on it. Therefore, one might be tempted to derive a similar relation between Schnorr null sets and *computable* martingales. However, Schnorr [18] pointed out that the increase in capital of a successful computable martingale can be so slow it cannot be computably detected. Therefore, he introduced *orders* ("Ordnungsfunktionen"), which allow to ensure an effective control over the capital.

In general, any positive, real, unbounded function g is called an *order*. (It should be remarked that, in Schnorr's terminology, an "Ordnungsfunktion" is always computable.)

Schnorr showed that Schnorr nullsets can be characterized via computable martingales successful against computable orders.

Theorem 13 (Schnorr). *A set $\mathcal{X} \subseteq 2^{\mathbb{N}}$ is Schnorr 1-null if and only if there exists a computable martingale d and a computable order g such that d is g-successful on all $B \in \mathcal{X}$.*

Schnorr calls the functions $g(n) = 2^{(1-s)n}$ *exponential orders*, so much of the theory of effective dimension is already, though apparently without explicit reference, present in Schnorr's treatment of algorithmic randomness [18].

If one drops the requirement of being g-successful for some computable g, one obtains the concept of *computable randomness*. Wang [26] showed that the concepts of computable randomness and Schnorr randomness do not coincide. There are Schnorr random sequences on which some computable martingale succeeds. However, the differences vanish if it comes to dimension.

Theorem 14. *For any sequence $B \in 2^{\mathbb{N}}$,*

$$\dim_{\mathrm{H}}^{\mathrm{S}} B = \inf\{s \in \mathbb{Q} : \textit{ some computable martingale } d \textit{ is } s\textit{-successful on } B\}.$$

So, in contrast to randomness, the approach via Schnorr tests and the approach via computable martingales to dimension yield the same concept.

Besides, we can build on Theorem 14 to introduce *Schnorr packing dimension*.

Definition 15. *Given a sequence $A \in 2^{\mathbb{N}}$, we define the* Schnorr packing dimension *of A, $\dim_{\mathrm{P}}^{\mathrm{S}} A$, as*

$$\dim_{\mathrm{P}}^{\mathrm{S}} A = \inf\{s \in \mathbb{Q} : \textit{ some comp. martingale is strongly } s\textit{-successful on } A\}$$

Schnorr packing dimension is implicitly introduced as a computable version of *strong dimension* in [1]. It follows from the definitions that for any sequence $A \in 2^{\mathbb{N}}$, $\dim_{\mathrm{H}}^{\mathrm{S}} A \leq \dim_{\mathrm{P}}^{\mathrm{S}} A$. We call sequences for which Schnorr Hausdorff and Schnorr packing dimension coincide *Schnorr regular* (see [24] and [1]). It is easy to construct a non-Schnorr regular sequence, however, in Section 6 we will see that such sequences already occur within the class of c.e. sets.

5 A Machine Characterization of Schnorr Dimension

One of the most powerful arguments in favor of Martin-Löf's approach to randomness is the coincidence of the Martin-Löf random sequences with the sequences that are incompressible in terms of (prefix free) Kolmogorov complexity.

Such an elegant characterization via machine compressibility is possible neither for Schnorr randomness nor Schnorr dimension. To obtain a machine characterization of Schnorr dimension, we have to restrict the admissible machines to those with domains having computable measure.

Definition 16. *A prefix free machine M is* computable *if $\sum_{w \in \mathrm{dom}(M)} 2^{-|w|}$ is a computable real number.*

Note that, as in the case of Schnorr tests, if a machine is computable, then its domain is computable (but not vice versa). To determine whether $M(w) \downarrow$,

enumerate $\mathrm{dom}(M)$ until $\sum_{w\in\mathrm{dom}(M)} 2^{-|w|}$ is approximated by a precision of 2^{-N}, where $N > |w|$. If $M(w)\downarrow$, w must have been enumerated up to this point.

Furthermore, we sometimes assume that the measure of the domain of a computable machine is 1. This can be justified, as in the case of Schnorr tests, by adding superfluous strings to the domain.

The definition of machine complexity follows the standard scheme. We restrict ourselves to prefix free machines.

Definition 17. *Given a Turing machine M with prefix free domain, the M-*complexity *of a string x is defined as*

$$\mathrm{K}_M(x) = \min\{|p| : M(p) = x\},$$

where $\mathrm{K}_M(x) = \infty$ if there does not exist a $p \in \{0,1\}^*$ such that $M(p) = x$.

We refer to the books by Li and Vitanyi [8] and Downey and Hirschfeldt [5] for comprehensive treatments on machine (Kolmogorov) complexity.

Downey and Griffiths [4] show that a sequence A is Schnorr random if and only if for every computable machine M, there exists a constant c such that $\mathrm{K}_M(A\upharpoonright_n) \geq n - c$. Building on this characterization, we can go on to describe Schnorr dimension as asymptotic entropy with respect to computable machines.

Theorem 18. *For any sequence A it holds that*

$$\dim^{\mathrm{S}}_{\mathrm{H}} A = \inf_M \underline{\mathrm{K}}_M(A) \stackrel{\mathrm{def}}{=} \inf_M \left\{ \liminf_{n\to\infty} \frac{\mathrm{K}_M(A\upharpoonright_n)}{n} \right\},$$

where the infimum is taken over all computable prefix free machines M.

One can use an analogous argument to obtain a machine characterization of Schnorr packing dimension.

Theorem 19. *For any sequence A it holds that*

$$\dim^{\mathrm{S}}_{\mathrm{P}} A = \inf_M \overline{\mathrm{K}}_M(A) \stackrel{\mathrm{def}}{=} \inf_M \left\{ \limsup_{n\to\infty} \frac{\mathrm{K}_M(A\upharpoonright_n)}{n} \right\},$$

where the infimum is taken over all computable prefix free machines M.

6 Schnorr Dimension and Computable Enumerability

Usually, when studying algorithmic randomness, interest focuses on *c.e. reals* (i.e. left-computable real numbers) rather than on *c.e. sets*. The reason is that c.e. sets exhibit a trivial behavior with respect to most randomness notions, while there are c.e. reals which are Martin-Löf random, such as Chaitin's Ω.

As regards c.e. reals, we can extend the result of Downey and Griffiths [4] that every Schnorr random c.e. real is of high Turing degree.

Theorem 20. *Every sequence of positive Schnorr Hausdorff dimension has high Turing degree. That is, if* $\dim_{\mathrm{H}}^{\mathrm{S}} A > 0$*, then* $S' \equiv_{\mathrm{T}} 0''$*.*

Using the fact that every noncomputable c.e. set contains an infinite computable subset, and the fact that, for all n, $\lambda\{A \in 2^{\mathbb{N}} : A(n) = 1\} = 1/2$, it is not hard to show that no c.e. set can be Schnorr random.

It does not seem immediately clear how to improve this to Schnorr dimension zero. Indeed, defining coverings from the enumeration of a set directly might not work, because due to the dimension factor in Hausdorff measures, longer strings will be weighted higher. Depending on how the enumeration is distributed, this might not lead to a Schnorr s-covering at all.

However, one might exploit the somewhat predictable nature of a c.e. set to define a computable martingale which is, for any $s > 0$, s-successful on the characteristic sequence of the enumerable set, thereby ensuring that each c.e. set has Schnorr Hausdorff dimension 0.

Theorem 21. *Every computably enumerable set* $A \subseteq \mathbb{N}$ *has Schnorr Hausdorff dimension zero.*

On the other hand, concerning upper entropy, c.e. sets may exhibit a rather complicated structure, in sharp contrast to the case of effective (constructive) dimension, where Barzdin's Theorem [2] ensures that all c.e. sets have effective packing dimension 0. As the proof of the following theorem shows, this is due to the requirement that all machines involved in the determination of Schnorr dimension are total.

Theorem 22. *There exists a computably enumerable set* $A \subseteq \mathbb{N}$ *such that*

$$\dim_{\mathrm{P}}^{\mathrm{S}} A = 1.$$

References

1. K. B. Athreya, J. M. Hitchcock, J. H. Lutz, and E. Mayordomo. Effective strong dimension in algorithmic information and computational complexity. In *Proceedings of the Twenty-First Symposium on Theoretical Aspects of Computer Science (Montpellier, France, March 25–27, 2004)*, pages 632–643. Springer-Verlag, 2004.
2. Ja. M. Barzdin'. Complexity of programs which recognize whether natural numbers not exceeding n belong to a recursively enumerable set. *Sov. Math., Dokl.*, 9:1251–1254, 1968.
3. J.-Y. Cai and J. Hartmanis. On Hausdorff and topological dimensions of the Kolmogorov complexity of the real line. *J. Comput. System Sci.*, 49(3):605–619, 1994.
4. R. G. Downey and E. J. Griffiths. Schnorr randomness. *J. Symbolic Logic*, 69(2): 533–554, 2004.
5. R. G. Downey and D. R. Hirschfeldt. Algorithmic randomness and complexity. book, in preparation, 2004.
6. H. G. Eggleston. The fractional dimension of a set defined by decimal properties. *Quart. J. Math., Oxford Ser.*, 20:31–36, 1949.

7. K. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, 1990.
8. M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Graduate Texts in Computer Science. Springer-Verlag, New York, 1997.
9. J. H. Lutz. Dimension in complexity classes. In *Proceedings of the Fifteenth Annual IEEE Conference on Computational Complexity*, pages 158–169. IEEE Computer Society, 2000.
10. J. H. Lutz. Gales and the constructive dimension of individual sequences. In *Automata, languages and programming (Geneva, 2000)*, volume 1853 of *Lecture Notes in Comput. Sci.*, pages 902–913. Springer, Berlin, 2000.
11. J. H. Lutz. The dimensions of individual strings and sequences. *Inform. and Comput.*, 187(1):49–79, 2003.
12. E. Mayordomo. A Kolmogorov complexity characterization of constructive Hausdorff dimension. *Inform. Process. Lett.*, 84(1):1–3, 2002.
13. J. Reimann. *Computability and fractal dimension*. Doctoral dissertation, Universität Heidelberg, 2004.
14. J. Reimann and F. Stephan. Effective Hausdorff dimension. In *Logic Colloquium '01 (Vienna)*. Assoc. Symbol. Logic. to appear.
15. B. Y. Ryabko. Coding of combinatorial sources and Hausdorff dimension. *Dokl. Akad. Nauk SSSR*, 277(5):1066–1070, 1984.
16. B. Y. Ryabko. Noise-free coding of combinatorial sources, Hausdorff dimension and Kolmogorov complexity. *Problemy Peredachi Informatsii*, 22(3):16–26, 1986.
17. B. Y. Ryabko. An algorithmic approach to the prediction problem. *Problemy Peredachi Informatsii*, 29(2):96–103, 1993.
18. C.-P. Schnorr. *Zufälligkeit und Wahrscheinlichkeit. Eine algorithmische Begründung der Wahrscheinlichkeitstheorie*. Springer-Verlag, Berlin, 1971.
19. R. M. Solovay. Lecture notes on algorithmic randomness. unpublished manuscript, UCLA, 1975.
20. L. Staiger. Constructive dimension equals Kolmogorov complexity. Inform. Proc. Letters, to appear.
21. L. Staiger. Kolmogorov complexity and Hausdorff dimension. *Inform. and Comput.*, 103(2):159–194, 1993.
22. L. Staiger. A tight upper bound on Kolmogorov complexity and uniformly optimal prediction. *Theory of Computing Systems*, 31(3):215–229, 1998.
23. S. J. Taylor and C. Tricot. Packing measure, and its evaluation for a Brownian path. *Trans. Amer. Math. Soc.*, 288(2):679–699, 1985.
24. C. Tricot, Jr. Two definitions of fractional dimension. *Math. Proc. Cambridge Philos. Soc.*, 91(1):57–74, 1982.
25. J. Ville. *Etude critique de la notion de collectif*. Gauthier-Villars, 1939.
26. Y. Wang. A separation of two randomness concepts. *Inform. Process. Lett.*, 69(3): 115–118, 1999.

Abstract Geometrical Computation: Turing-Computing Ability and Undecidability

Jérôme Durand-Lose*

Laboratoire d'Informatique Fondamentale d'Orléans,
Université d'Orléans, B.P. 6759, F-45067 Orléans Cedex 2, France
Jerome.Durand-Lose@lifo.univ-orleans.fr

Abstract. In the Cellular Automata (CA) literature, discrete lines inside (discrete) space-time diagrams are often idealized as Euclidean lines in order to analyze a dynamics or to design CA for special purposes. In this article, we present a parallel analog model of computation corresponding to this idealization: dimensionless signals are moving on a continuous space in continuous time generating Euclidean lines on (continuous) space-time diagrams. Like CA, this model is parallel, synchronous, uniform in space and time, and uses local updating. The main difference is that space and time are continuous and not discrete (*i.e.* $\mathbb{R}$ instead of $\mathbb{Z}$). In this article, the model is restricted to $\mathbb{Q}$ in order to remain inside Turing-computation theory. We prove that our model can carry out any Turing-computation through two-counter automata simulation and provide some undecidability results.

Keywords: Abstract geometrical computation, Analog model of computation, Cellular automata, Geometry, Turing universality.

1 Introduction

Cellular automata (CA) form a well known and studied model of computation and simulation. Configurations are $\mathbb{Z}$-arrays of cells, the states of which belong to a finite set. Each cell can only access the states of its neighboring cells. All cells are updated iteratively and simultaneously. The main characteristics of CA are: parallelism, synchrony, uniformity and locality of updating. The trace of a computation, or the orbit starting from an initial configuration, is represented as a *space-time diagram*: a coloring of $\mathbb{Z} \times \mathbb{N}$ with states.

Discrete lines are often observed on these diagrams. They can be the key to understanding the dynamics and correspond to so-called *particles* or *signals* as in, *e.g.*, [1–pp. 87–94] or [2, 3, 4, 5]. They can also be the tool to design CA for precise purposes and then named *signals* and used for, *e.g.*, prime number generation [6], firing squad synchronization [7, 8, 9, 10] or reversible simulation [11, 12].

* This research was done while the author was at the LIP, ÉNS Lyon, France.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 106–116, 2005.

These discrete lines systems have also been studied on their own [13, 14]. All the figures in cited papers exhibit discrete lines which are explicitly refereed to –and are often idealized as Euclidean lines– for describing or designing. Many more articles could have been cited, the cited ones were randomly chosen in order to show the variety.

We want to consider this idealization: Euclidean lines on their own –not as a passing point for analysis or conception– while remaining with the main characteristics of CA. As Euclidean lines belong to Euclidean spaces and not $\mathbb{Z} \times \mathbb{N}$, the support of space and time is now $\mathbb{R} \times \mathbb{R}^+$. Configurations (at a given time or the restriction of the space-time diagram to a given time) are not mappings from $\mathbb{Z}$ to a finite set of states but partial mappings from $\mathbb{R}$ to the union of a finite set of *meta-signals* and a finite set of *collision rules*, defined for finitely many positions. The time scale is $\mathbb{R}^+$ (not $\mathbb{N}$), so that there is no such thing as a "next configuration". The following configurations are defined by the uniform movement of each signal, the speed of which is defined by its associated meta-signal. When two or more signals meet, a *collision* happens. Each collision follows a collision rule, each of which is defined by a pair of sets of meta-signals: *(incoming meta-signals, outgoing meta-signals)*. There must be at least two incoming meta-signals and all sets of incoming meta-signals must differ (which means determinism). In the configurations following a collision, incoming signals are removed and outgoing signals corresponding to the outgoing meta-signals are added.

Due to the continuous nature of space and time and the uniformity of movements, the traces of signals form Euclidean lines on the space-time diagrams, which we freely call *signals*. Each signal corresponds to a meta-signal which indicates its slope. Since there are finitely many meta-signals, there are finitely many slopes. This limitation may seem restrictive and unrealistic, even awkward as a finite quantification inside an analog model of computation. Let us notice that, first, it comes from CA: once a discrete line is identified, wherever (and whenever) the same pattern appears, the same line is expected, thus with the same slope. Second, we give two pragmatic arguments: (1) laws to define the new slopes from the previous ones in collisions are not so easy to design and pretty cumbersome to manipulate; (2) there is already much computing power (as presented in this paper and addressed in the conclusion).

Before presenting the results in this paper, we want to convince the reader that it is not just "one more analog model of computation". First, it does not come "out of the blue" because of its CA origin (where it is often implicitly used). Second, let's do a brief tour of analog/super-Turing models of computation[1]. To our knowledge, the closest model is the Mondrian automata of Jacopini and Sontacchi [16]. They also define dynamics and work on space-time diagrams which are mappings from $\mathbb{R}^n \times \mathbb{R}$ to a finite set of colors. Their diagrams should be composed of bounded finite polyhedra; we are only addressing lines –(hyper-)faces

[1] A wider survey can be found in [15–Chap. 2], unfortunately, there is no room here for more on the subject.

are not considered– and our diagrams may be unbounded and accumulation points may appear (they just forbid them; we address them in [17]). Another close model is the piecewise-constant derivative system [18, 19, 20]. Continuous space is partitioned into finitely many polygonal regions. The trajectory from a point is defined by a constant derivative on each region, thus an orbit is a sequence of (Euclidean) line segments. This model is very different from ours: it is sequential –there is only one "signal"– and the hyper-faces that delimit the regions are artifacts that do not exist in our model.

All the other models are based on totally different approaches. Some only define mapping over $\mathbb{R}$ like recursive analysis (type 2 Turing machines) [21] or analog recursive functions [22]. Many use a discrete iterative time like the BSS model [23] or analog recurrent neural networks [24]. The models with continuous time mostly use differential equations, finite support (variables) with uncountably many possible values [25, 26, 27, 28]. To our knowledge, our model is the only one that is a dynamical system with continuous time and space but finitely many local values; this is also one of the few parallel ones.

In this paper, space and time are restricted to rationals (this is possible since all the operations used preserve rationality); this allows manipulation by, *e.g.*, Turing machines, thus remaining in the classical discrete computation theory, and is enough for Turing-computing capability. All intervals should be understood over $\mathbb{Q}$, not $\mathbb{R}$. Extension to $\mathbb{R}$ is automatic but only the rational case is addressed here[2].

After formally defining our model in Sect. 2, we show that any Turing-computation can be carried out through the simulation of two-counter automata in Sect. 3. The counters are encoded in unary and the instructions are going forth and back between the two groups of signals. The nature of space is used here: any finite number of signals can be enclosed in a bounded interval of $\mathbb{Q}$.

In Sect. 4, with simple reductions from the Halting problem, we prove that the following problems are undecidable: finite number of collisions, collision with a given signal, appearance of a given signal, and disappearance of all signals.

Conclusion, remarks and perspectives are gathered in Sect. 5.

2 Definitions

Our abstract geometrical computations are defined by the following machines:

Definition 1. *A* signal machine *is defined by* (M, S, R) *where* M *is a finite set,* S *is a mapping from* M *to* $\mathbb{Q}$*, and* R *is a subset of* $\mathcal{P}(M)\times\mathcal{P}(M)$ *that corresponds to a partial mapping of the subsets of* M *of cardinality at least 2 to the subsets of* M *(both domain and range are restricted to elements of different* S*-images).*

[2] Our personal opinion is that $\mathbb{Q}$ is indeed continuous –if not analog– even if it is countable because (1) there are no next nor previous element, (2) any interval is either empty, a singleton or infinite, and (3) its usual topology is not the discrete one.

The elements of M are called *meta-signals*. Each instance of a meta-signal is a *signal* which corresponds to a line segment in the space-time diagram. The mapping S assigns rational *speeds* to meta-signals, *i.e.* the slopes of the segments. The set R defines the *collision rules*, *i.e.* what happens when two or more signals meet. It also defines the intersections / extremities of the segments. The signal machines are deterministic because R must correspond to a mapping; if it were just a relation, then the machine would be non-deterministic.

Definition 2. *A* configuration*, c, is a partial mapping from $\mathbb{Q}$ to the union of M and R such that the set $\{\, q \in \mathbb{Q} \mid c(q)$ is defined$\,\}$ is finite.*

Let us define the dynamics:

A signal corresponding to a meta-signal μ at a position q, *i.e.* $c(q) = \mu$, is moving uniformly with constant speed $S(\mu)$. A signal must start in the initial configuration or be generated by a collision. It must end in a collision or in the last configuration. This corresponds to condition 1. in Def. 2.

A collision corresponds to a collision rule $\rho = (\rho^-, \rho^+)$, also noted as $\rho^- \to \rho^+$. All, and only, signals corresponding to the meta-signals in ρ^- (resp. ρ^+) must end (resp. start) in this collision. No other signal should be present. This corresponds to condition 2. in Def. 2.

Definition 3. *The* space-time diagram*, or orbit, issued from an initial configuration c_0 and lasting for T[3], is a mapping c from $[0,T]$ to configurations (i.e. a partial mapping from $\mathbb{Q} \times [0,T]$ to $M \cup R$) such that, $\forall (q,t) \in \mathbb{Q} \times [0,T]$:*

1. *if $c_t(q) = \mu$ then $\exists t_i, t_f \in [0,T]$ with $t_i < t < t_f$ or $0 = t_i = t < t_f$ or $t_i < t = t_f = T$ s.t.:*
   - *$\forall t' \in (t_i, t_f)$, $c_{t'}(q + S(\mu)(t' - t)) = \mu$,*
   - *$t_i = 0$ or $c_{t_i}(q_i) \in R$ and $\mu \in (c_{t_i}(q_i))^+$ where $q_i = q + S(\mu)(t_i - t)$,*
   - *$t_f = T$ or $c_{t_f}(q_f) \in R$ and $\mu \in (c_{t_f}(q_f))^-$ where $q_f = q + S(\mu)(t_f - t)$;*
2. *if $c_t(q) = \rho^- \to \rho^+ \in R$ then $\exists \varepsilon$, $0 < \varepsilon$, $\forall t' \in [0,T]$,*
   - *if $t - \varepsilon < t' < t$ then $\forall \mu \in \rho^-$, $c_t(q + S(\mu)(t' - t)) = \mu$,*
   - *if $t < t' < t + \varepsilon$ then $\forall \mu \in \rho^+$, $c_t(q + S(\mu)(t' - t)) = \mu$,*
   - *$\exists \alpha$, $0 < \alpha$, $c_{[t-\varepsilon, t)}([q - \alpha, q + \alpha]) = \rho^-$ and $c_{(t, t+\varepsilon]}([q - \alpha, q + \alpha]) = \rho^+$.*

The traces of signals represent line segments whose directions are defined by $(S(.), 1)$ (1 is the temporal coordinate). Collisions correspond to the extremities of these segments. Examples of space-time diagrams are provided by the various figures. Time is always increasing upwards.

3 Turing-Computation Capability

We prove the Turing-computation power of our model by simulating any two-counter automaton (a finite automaton couple with two counters, A and B). Each automaton can only *add/subtract* 1 to a counter and *branch if* a counter

[3] This definition can easily be extended to the case where $T = \infty$.

is *non-zero*. It can be described with a six-operations (the three aforementioned ones for each of the two counters) assembly language with branching labels as on the left part of Fig. 6 (see [29] for more on two-counter automata).

Definition 4. *A* two-counter automaton *has two non-negative integer counters (*`A` *and* `B`*). Its action is defined by a sequence of labeled instructions; the only instructions available are:*

- `A++` *: add* 1 *to the value of* `A`*,*
- `B++` *: add* 1 *to the value of* `B`*,*
- `A--` *: if* `A` *is not at* 0 *then subtract* 1 *from its value,*
- `B--` *: if* `B` *is not at* 0 *then subtract* 1 *from its value,*
- `A !=0` m *: if* `A` *is not at* 0 *then the next instruction is the one labeled* m*,*
- `B !=0` m *: if* `B` *is not at* 0 *then the next instruction is the one labeled* m*.*

The simulation is carried out with both counters unary encoded. Configurations are formed by, left to right: a left end-marking signal, a signals amounting for the value of A, one signal encoding the current instruction, b signals for B and a right end-marking signal. This is depicted on Fig. 1.

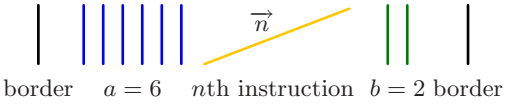


Fig. 1. Configuration encoding

The only active signal (no collision ever happens without this signal being present) is the middle one; this signal both carries out the operation and encodes the line number. It goes forth and back between the signals encoding A and B. The end-markers are needed when the value is zero; in such a case, there would be no other signal on the side and the active signal would drift away infinitely. These end-markers also provide a simple way to test for non-zero: an end marker is met if and only if the value of the corresponding counter is zero.

They are two kinds of meta-signals: five for the counters and end-markers and the ones generated for the code as depicted on Fig. 2. The meta-signals of the first kind are: border, a and b of speed 0 used to mark the borders and to encode respectively A and B in unary, and aMv and bMv of speed $1/2$ and $-1/2$ used to increment A and B. The use of bMv is explained in the presentation of `B++` (Fig. 5). For the second kind, each line n of the program is converted to $\overrightarrow{n}$ and $\overleftarrow{n}$ of speed 1 and -1 (except for the test B $\neq 0$ which generates $\overrightarrow{n}$, $\overleftarrow{n}_Y$, and $\overleftarrow{n}_N$). The program is encoded in the collision rules.

The full transformation of a program into a signal machine is given in Fig. 3. We only detail the action of the collision rules generated for `B !=0` and `B++` instructions (at line n). All other instructions are carried out in a similar way.

In the space-time diagram of figures 4 and 5, we suppose that instructions at lines $n-1$ and $n+1$ are not doing anything, except in the right cases where the previous one is `B++`, so that there is some bMv to set in position.

Fig. 2. The two kinds of meta-signals

Instruction	Rightwards	Leftwards
n `A++`	$\{\overrightarrow{n}, \mathsf{border}\} \to \{\overleftarrow{n}, \mathsf{border}\}$ $\{\overrightarrow{n}, \mathsf{b}\} \to \{\overleftarrow{n}, \mathsf{b}\}$ $\{\overrightarrow{n}, \mathsf{bMv}\} \to \{\overleftarrow{n}, \mathsf{b}\}$	$\{\mathsf{border}, \overleftarrow{n}\} \to \{\mathsf{border}, \mathsf{aMv}, \overrightarrow{n+1}\}$ $\{\mathsf{a}, \overleftarrow{n}\} \to \{\mathsf{a}, \mathsf{aMv}, \overrightarrow{n+1}\}$ $\{\mathsf{aMv}, \overleftarrow{n}\} \to \{\mathsf{a}, \mathsf{aMv}, \overrightarrow{n+1}\}$
n `B++`	$\{\overrightarrow{n}, \mathsf{border}\} \to \{\overleftarrow{n}, \mathsf{bMv}, \mathsf{border}\}$ $\{\overrightarrow{n}, \mathsf{b}\} \to \{\overleftarrow{n}, \mathsf{bMv}, \mathsf{b}\}$ $\{\overrightarrow{n}, \mathsf{bMv}\} \to \{\overleftarrow{n}, \mathsf{bMv}, \mathsf{b}\}$	$\{\mathsf{border}, \overleftarrow{n}\} \to \{\mathsf{border}, \overrightarrow{n+1}\}$ $\{\mathsf{a}, \overleftarrow{n}\} \to \{\mathsf{a}, \overrightarrow{n+1}\}$ $\{\mathsf{aMv}, \overleftarrow{n}\} \to \{\mathsf{a}, \overrightarrow{n+1}\}$
n `A--`	$\{\overrightarrow{n}, \mathsf{border}\} \to \{\overleftarrow{n}, \mathsf{border}\}$ $\{\overrightarrow{n}, \mathsf{b}\} \to \{\overleftarrow{n}, \mathsf{b}\}$ $\{\overrightarrow{n}, \mathsf{bMv}\} \to \{\overleftarrow{n}, \mathsf{b}\}$	$\{\mathsf{border}, \overleftarrow{n}\} \to \{\mathsf{border}, \overrightarrow{n+1}\}$ $\{\mathsf{a}, \overleftarrow{n}\} \to \{\overrightarrow{n+1}\}$ $\{\mathsf{aMv}, \overleftarrow{n}\} \to \{\overrightarrow{n+1}\}$
n `B--`	$\{\overrightarrow{n}, \mathsf{border}\} \to \{\overleftarrow{n}, \mathsf{border}\}$ $\{\overrightarrow{n}, \mathsf{b}\} \to \{\overleftarrow{n}\}$ $\{\overrightarrow{n}, \mathsf{bMv}\} \to \{\overleftarrow{n}\}$	$\{\mathsf{border}, \overleftarrow{n}\} \to \{\mathsf{border}, \overrightarrow{n+1}\}$ $\{\mathsf{a}, \overleftarrow{n}\} \to \{\mathsf{a}, \overrightarrow{n+1}\}$ $\{\mathsf{aMv}, \overleftarrow{n}\} \to \{\mathsf{a}, \overrightarrow{n+1}\}$
n `A !=0` m	$\{\overrightarrow{n}, \mathsf{border}\} \to \{\overleftarrow{n}, \mathsf{border}\}$ $\{\overrightarrow{n}, \mathsf{b}\} \to \{\overleftarrow{n}, \mathsf{b}\}$ $\{\overrightarrow{n}, \mathsf{bMv}\} \to \{\overleftarrow{n}, \mathsf{b}\}$	$\{\mathsf{border}, \overleftarrow{n}\} \to \{\mathsf{border}, \overrightarrow{n+1}\}$ $\{\mathsf{a}, \overleftarrow{n}\} \to \{\mathsf{a}, \overrightarrow{m}\}$ $\{\mathsf{aMv}, \overleftarrow{n}\} \to \{\mathsf{a}, \overrightarrow{m}\}$
n `B !=0` m	$\{\overrightarrow{n}, \mathsf{border}\} \to \{\overleftarrow{n}_N, \mathsf{border}\}$ $\{\overrightarrow{n}, \mathsf{b}\} \to \{\overleftarrow{n}_Y, \mathsf{b}\}$ $\{\overrightarrow{n}, \mathsf{bMv}\} \to \{\overleftarrow{n}_Y, \mathsf{b}\}$	$\{\mathsf{border}, \overleftarrow{n}_N\} \to \{\mathsf{border}, \overrightarrow{n+1}\}$ $\{\mathsf{a}, \overleftarrow{n}_N\} \to \{\mathsf{a}, \overrightarrow{n+1}\}$ $\{\mathsf{aMv}, \overleftarrow{n}_N\} \to \{\mathsf{a}, \overrightarrow{n+1}\}$ $\{\mathsf{border}, \overleftarrow{n}_Y\} \to \{\mathsf{border}, \overrightarrow{m}\}$ $\{\mathsf{a}, \overleftarrow{n}_Y\} \to \{\mathsf{a}, \overrightarrow{m}\}$ $\{\mathsf{aMv}, \overleftarrow{n}_Y\} \to \{\mathsf{a}, \overrightarrow{m}\}$
	$\{\overrightarrow{\mathsf{stop}}, \mathsf{border}\} \to \{\overleftarrow{\mathsf{stop}}, \mathsf{border}\}$ $\{\overrightarrow{\mathsf{stop}}, \mathsf{b}\} \to \{\overleftarrow{\mathsf{stop}}, \mathsf{b}\}$ $\{\overrightarrow{\mathsf{stop}}, \mathsf{bMv}\} \to \{\overleftarrow{\mathsf{stop}}, \mathsf{b}\}$	$\{\mathsf{border}, \overleftarrow{\mathsf{stop}}\} \to \{\mathsf{border}\}$ $\{\mathsf{a}, \overleftarrow{\mathsf{stop}}\} \to \{\mathsf{a}\}$ $\{\mathsf{aMv}, \overleftarrow{\mathsf{stop}}\} \to \{\mathsf{a}\}$

Fig. 3. Translation of instructions

The `B !=0` instruction is carried out very simply: the active signal goes right. If it encounters border (counter `B` is zero) then it comes back as $\overleftarrow{n}_N$. Otherwise (counter `B` is not zero), it encounters b or bMv and comes back as $\overleftarrow{n}_Y$. On the left it meets border or a and turns to the next instruction (n+1) if it was $\overleftarrow{n}_N$ or the indicated jump (m) if it was $\overleftarrow{n}_Y$. This is all summed up in Fig. 4.

The `B++` instruction is carried out by issuing a moving b signal (*i.e.* bMv) that will be set in position by the next return of the active signal as depicted on Fig. 5. The active signal is faster than bMv so that it can fix bMv before bMv

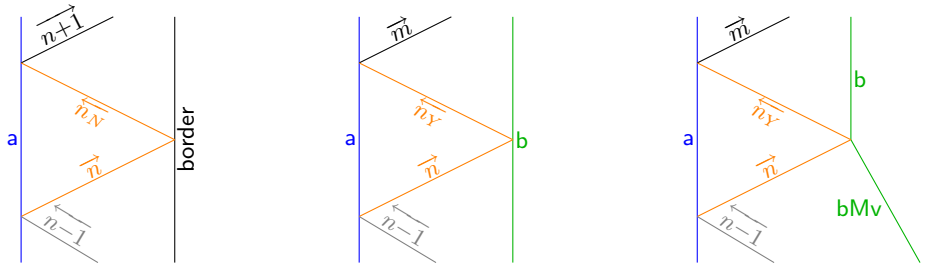


Fig. 4. Implementation of n `B != 0` m

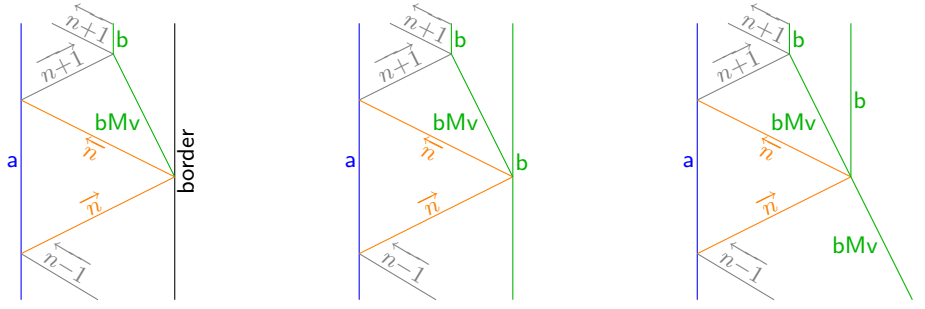


Fig. 5. Implementation of `B++`

reaches the left side. Since space is continuous there is always room for more signal in the middle. Each instruction sets the moving bMv (or aMv), if any.

The instruction `A++` is carried out similarly. The instructions `A--` and `B--` are just erasing exactly one corresponding signal, if any. The non-zero test for `A` is done by simply noticing that the left border is met only if it is zero; branching is done on the collision on the left side.

Figure 6 provides three space-time diagrams associated to different initial values. The pictures are strained vertically in order to fit.

A two-counter automaton stops when it should execute the instruction right after the last one, *i.e.*, when executing instruction $n_0{+}1$ (n_0 is the number of instructions). We call the instruction $\overrightarrow{n_0{+}1}$ ($\overleftarrow{n_0{+}1}$) $\overrightarrow{\mathsf{stop}}$ and $\overleftarrow{\mathsf{stop}}$. These signals fix any moving aMv or bMv as indicated at the bottom of Fig. 3.

This way, any two-counter automaton can be simulated by a signal machine. Signal machines thus form a model of computation which has at least Turing-computing capability.

4 Undecidability Results

Straight from the possibility to simulate Turing Machine comes a few undecidability results. The proofs are not detailed, they are only sketched as they correspond to classical reduction from the Halting problem.

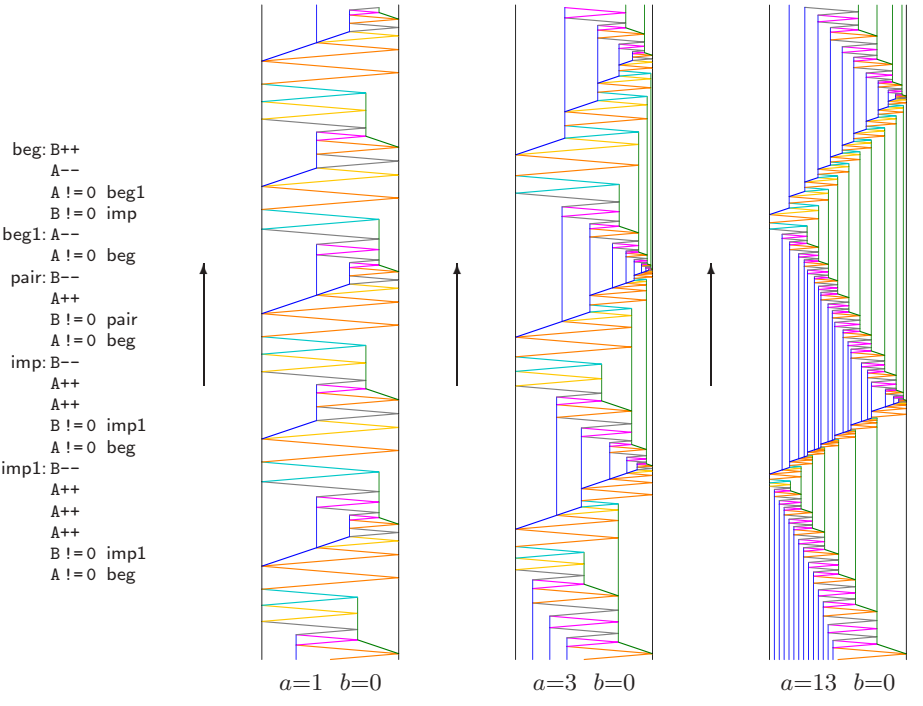


Fig. 6. A two-counter automaton and its simulations for three different initial values

In all the following problems, a signal machine is *rational* iff all its speeds are rationals and all initials positions are rational. A direct recurrence shows that all collisions take place at rational locations. This is needed for data to be addressed in classical recursion theory.

[Finite number of collisions]
Instance . A rational signal machine, and an initial configuration.
Question. Does the computation of the machine on the initial configuration stop?

The undecidability comes from the ability of rational signal machine to simulate any two-counter automaton in a way that there is finitely many collisions iff the simulation stops.

[Appearance of a given meta-signal]
Instance . A rational signal machine, an initial configuration, and a meta-signal.
Question. Does the computation of the machine on the initial configuration ever generates a signal of this meta-signal?

The undecidability comes from the ability of rational signal machine to simulate any two-counter automaton in a way that a signal $\overrightarrow{\text{stop}}$ appears iff the simulation stops.

[Collision with a given signal]
Instance . A rational signal machine, an initial configuration, and a signal in the initial configuration.
Question. Is there any collision involving the given signal on the computation of the machine on the initial configuration?

The undecidability comes from a slight modification of the machine: add a second border signal on the right and make $\overrightarrow{\mathsf{stop}}$ pass the first one and collide with the second. This second border takes part in a collision if and only if $\overrightarrow{\mathsf{stop}}$ appears.

[Disappearance of all signals]
Instance . A rational signal machine, and an initial configuration.
Question . Does the computation of the machine on the initial configuration stop on an empty configuration?

The undecidability also comes from the simulation: it is easy to modify it in such that $\overrightarrow{\mathsf{stop}}$ and $\overleftarrow{\mathsf{stop}}$ erase everything.

5 Conclusion

As long as the model is restricted to rationals, there are finitely many signals present at any instant and there is no accumulation, the model is Turing-universal and can be simulated by any Turing machine and is thus Turing-equivalent. If the "finite number of signals" condition is lifted, but signals are isolated, then this is a super-Turing model of computation following the "Infinity principle" of [30]. The analog power really appears when one of the remaining two constraints is lifted.

Allowing real values for speeds or positions is simple. These real values can be used as oracles and thus provide computing ability that goes beyond Turing-computation.

With a careful "treatment of accumulations", it is possible to access infinite Turing computation or computation on ordinals [31, 32]. Our model then becomes somehow similar to the black hole model [33, 34] as done in [17].

References

1. Ilachinski, A.: Cellular automata –a discrete universe. World Scientific (2001)
2. Boccara, N., Nasser, J., Roger, M.: Particle-like structures and interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules. Phys. Rev. A **44** (1991) 866–875
3. Durand-Lose, J.: Parallel transient time of one-dimensional sand pile. Theoret. Comp. Sci. **205** (1998) 183–193
4. Hordijk, W., Shalizi, C.R., Crutchfield, J.P.: An upper bound on the products of particle interactions in cellular automata. Phys. D **154** (2001) 240–258
5. Jakubowsky, M.H., Steiglitz, K., Squier, R.: Computing with solitons: a review and prospectus. in [35], pp. 277–297 (2002)

6. Fischer, P.C.: Generation of primes by a one-dimensional real-time iterative array. J. ACM **12** (1965) 388–394
7. Goto, E.: Ōtomaton ni kansuru pazuru [Puzzles on automata]. In Kitagawa, T., ed.: Jōhōkagaku eno michi [The Road to information science], Kyoristu Shuppan Publishing Co., Tokyo (1966) 67–92
8. Varshavsky, V.I., Marakhovsky, V.B., Peschansky, V.A.: Synchronization of interacting automata. Math. System Theory **4** (1970) 212–230
9. Lindgren, K., Nordahl, M.G.: Universal computation in simple one-dimensional cellular automata. Complex Systems **4** (1990) 299–318
10. Mazoyer, J.: On optimal solutions to the Firing squad synchronization problem. Theoret. Comp. Sci. **168** (1996) 367–404
11. Durand-Lose, J.: Intrinsic universality of a 1-dimensional reversible cellular automaton. In: STACS '97. Number 1200 in LNCS, Springer (1997) 439–450
12. Durand-Lose, J.: Reversible space-time simulation of cellular automata. Theoret. Comp. Sci. **246** (2000) 117–129
13. Delorme, M., Mazoyer, J.: Signals on cellular automata. in [35], pp. 234–275 (2002)
14. Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theoret. Comp. Sci. **217** (1999) 53–80
15. Durand-Lose, J.: Calculer géométriquement sur le plan – machines à signaux. Habilitation à diriger des recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis (2003)
16. Jacopini, G., Sontacchi, G.: Reversible parallel computation: an evolving space-model. Theoret. Comp. Sci. **73** (1990) 1–46
17. Durand-Lose, J.: Abstract geometrical computation for black hole computation (extended abstract). In Margenstern, M., ed.: Universal Machines and Computations (MCU '04). Number 3354 in LNCS, Springer (2005) 175–186
18. Asarin, E., Maler, O.: Achilles and the Tortoise climbing up the arithmetical hierarchy. In: FSTTCS '95. Number 1026 in LNCS (1995) 471–483
19. Bournez, O.: Some bounds on the computational power of piecewise constant derivative systems. In: ICALP '97. Number 1256 in LNCS (1997) 143–153
20. Bournez, O.: Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. Theoret. Comp. Sci. **210** (1999) 21–71
21. Weihrauch, K.: Introduction to computable analysis. Texts in Theoretical computer science. Springer, Berlin (2000)
22. Moore, C.: Recursion theory on the reals and continuous-time computation. Theoret. Comp. Sci. **162** (1996) 23–44
23. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and real computation. Springer, New York (1998)
24. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. J. Comput. System Sci. **50** (1995) 132–150
25. Orponen, P.: A survey of continuous-time computation theory. In Du, D.Z., Ko, K.J., eds.: Advances in Algorithms, languages and complexity, Kluwer Academic Publisher (1994) 209–224
26. Šíma, J., Orponen, P.: Computing with continuous-time Liapunov systems. In: STOC '01, ACM Press (2001) 722–731
27. Pour-El, M.B.: Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers). Trans. Amer. Math. Soc. **199** (1974) 1–28
28. Branicky, M.S.: Universal computation and other capabilities of hybrid and continuous dynamical systems. Theoret. Comp. Sci. **138** (1995) 67–100

29. Minsky, M.: Finite and infinite machines. Prentice Hall (1967)
30. Eberbach, E., Wegner, P.: Beyond Turing machines. Bull. EATCS **81** (2003) 279–304
31. Hamkins, J.D., Lewis, A.: Infinite time turing machines. J. Symb. Log. **65** (2000) 567–604.
32. Hamkins, J.D.: Infinite time Turing machines: Supertask computation. Minds and Machines **12** (2002) 521–539.
33. Earman, J., Norton, J.D.: Forever is a day: supertasks in Pitowsky and Malament-Hogarth spacetimes. Philosophy of Science **60** (1993) 22–42
34. Etesi, G., Nemeti, I.: Non-Turing computations via Malament-Hogarth space-times. Int. J. Theor. Phys. **41** (2002) 341–370.
35. Adamatzky, A., ed.: Collision based computing. Springer (2002)

Computability in Computational Geometry

Abbas Edalat[1], Ali A. Khanban[1], and André Lieutier[2]

[1] Department of Computing, Imperial College London, UK
{ae,khanban}@doc.ic.ac.uk
[2] Dassault Systemes Provence,
Aix-en-Provence & LMC/IMAG, Grenoble, France
andre_lieutier@ds-fr.com

Abstract. We promote the concept of object directed computability in computational geometry in order to faithfully generalise the well-established theory of computability for real numbers and real functions. In object directed computability, a geometric object is computable if it is the effective limit of a sequence of finitary objects of the same type as the original object, thus allowing a quantitative measure for the approximation. The domain-theoretic model of computational geometry provides such an object directed theory, which supports two such quantitative measures, one based on the Hausdorff metric and one on the Lebesgue measure. With respect to a new data type for the Euclidean space, given by its non-empty compact and convex subsets, we show that the convex hull, Voronoi diagram and Delaunay triangulation are Hausdorff and Lebesgue computable.

1 Introduction

In his "Commentaries on the Difficulties in the Postulates of Euclid's Elements", Omar Khayyam, the 11th century Persian mathematician and poet, developed the first rudimentary notion of a real number. He first showed the equivalence of Euclid's notion of ratios with that of continued fractions. Then, in a stroke of genius, he defined two ratios as equal "when they can be expressed by the ratio of integer numbers with as great a degree of accuracy as we like" [17]. This idea thus contained the first notion of a real number and the germ of the concepts of computability and computation up to any precision. Three centuries later, Ghiasseddin Jamshid Kashani, another Persian mathematician, devised the first fixed point technique for computation in analysis in the beginning of the 15th century: he used a cubic polynomial in a recursive scheme to approximate the sine of 1° correctly up to 17 decimal places; see [2] for the details.

Following the formalisation of real numbers by Cantor and Dedekind in the 19th century and the development of recursion theory by Turing, Church, Gödel and Kleene in the first half of the 20th century, the concept of a computable real number was first defined by Turing in his seminal work in mid 1930's [18] and [19]. In the decades since that work, several notions of computability for real numbers and real functions have been proposed, as for example in [13], [15],

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 117–127, 2005.

[20], [16], [10], [1], which turn out to be essentially equivalent. A computable real number in all these different but equivalent approaches is in essence the limit of an effective sequence of rational numbers, and a computable real function is one which maps computable real numbers to computable real numbers in an effective way. The effective nature of the sequence of rational numbers approximating a computable number implies that each term of the sequence gives a lower and an upper bound for the real number, with the distance between the two bounds providing a quantitative measure of the approximation. Regarding a real number as an object and a rational number as a finitary object of the same type as a real number, we can say that a computable object is defined as the effective limit of two monotonic sequences of finitary objects, providing at each stage finitary lower and upper bounds for the computable object. In this sense, we say that the computability theory of real numbers and real functions is *object directed.*

In more recent years, several attempts have been made to define the notion of computability for subsets of the Euclidean space and operations on such subsets [12], [11], [4], [7], [8], [9], [14], [3], [21], [22]. Here, there are several different approaches which give rise to a number of non-equivalent theories of computability for subsets of the Euclidean space and operations on them.

The domain-theoretic framework introduced in [7] and [8] for studying computability of subsets of Euclidean spaces and their operations is an object directed theory for computational geometry, which faithfully generalises the object directed computability theory of real numbers and real functions. As the membership predicate of a proper subset of a Euclidean space is undecidable on its boundary, subsets with the same boundary are identified in the domain-theoretic framework and thus any subset is represented by two disjoint open subsets: its interior and its exterior[1]. With respect to any enumeration of a countable basis of the Euclidean topology, a computable subset is one whose interior and exterior are each the union of an effective increasing sequence of the basis elements. Thus, computability of an object is defined by two effective sequences of the same type converging to the object. In a similar way, computability of all basic operations on subsets such as union, intersection, and Minkowski sum, as treated in [8], as well as the convex hull, Voronoi diagram and Delaunay triangulation, as dealt with in [9] and [14], are always defined in terms of sequences of finitary objects of similar type. For example, the computability of the convex hull of a finite number of points in the Euclidean space is defined using two effective sequences of interior and exterior convex rational polygons converging to the interior and the exterior of the convex hull of the points.

The object directed computability provided by the domain-theoretic model provides other distinguished features:

– All basic predicates such as membership, subset inclusion and comparison as well as all basic operations are Scott continuous and computable in this model. Thus, algorithms developed in this framework are inherently robust in contrast with classical algorithms in computational geometry, which are

[1] The exterior of a set is the interior of its complement.

non-robust due to the non-computability of comparison of real numbers or the membership predicate of a subset in classical geometry.
- In this model, one obtains robust algorithms for computing operations such as convex hull, Voronoi diagram and Delaunay triangulation with the same complexity of the corresponding non-robust classical algorithms.
- Since the computability of an object is defined in terms of effective sequences of finitary objects of a similar type, one can employ two quantitative measures of approximation: one using the Hausdorff metric and one using the Lebesgue measure.

Therefore, in this framework the notion of computability of a geometric object and the task of computing it up to any required precision by the user are synthesised into one paradigm, thus providing the foundation of a robust CAD system.

In this paper, we study the three notions of recursion theoretic computability, Hausdorff computability and Lebesgue computability of three basic computational geometry operations, namely the convex hull, Voronoi diagram and Delaunay triangulation, in the context of a general data type for the Euclidean space given by the domain of non-empty compact convex subsets of the space ordered by reverse inclusion.

2 The Mathematical Model and the New Data Type

The *solid domain* $(\mathbf{S}\mathbb{R}^d, \sqsubseteq)$ of $\mathbb{R}^d$ is the collection of pairs of disjoint open subsets of $\mathbb{R}^d$ partially ordered componentwise by subset inclusion: $(I, E) \sqsubseteq (I', E')$ iff $I \subseteq I'$ and $E \subseteq E'$; it is a bounded complete ω-continuous dcpo [8]. A classical geometric object, i.e. a subset $A \subseteq \mathbb{R}^d$ is represented in this model as $(A^\circ, (A^c)^\circ)$, where X° and X^c denote respectively the interior and the complement of a set X. More generally, we think of an element $(I, E) \in \mathbf{S}\mathbb{R}^d$ as a *partial solid* or *partial geometric objects* with *interior* I, *exterior* E and *boundary* $(I \cup E)^c$. An element (I, E) is maximal in $\mathbf{S}\mathbb{R}^d$ iff $I = (E^c)^\circ$ and $E = (I^c)^\circ$, which imply that I and E are regular[2]. The collection of pairs of interiors of disjoint dyadic (or rational) d-polytopes forms a basis for $\mathbf{S}\mathbb{R}^d$. Any partial geometric object (I, E) can be obtained as the union of these basis elements.

Our new data type is described as follows. We assume that we have lower and upper rational bounds on the coordinates $(x_k)_{1 \le k \le d}$ of an imprecisely given point $x \in \mathbb{R}^d$ in say n given directions, that is we have $\beta_j \le \sum_{k=1}^{d} a_{jk} x_k \le \gamma_j$, where $(a_{jk})_{1 \le k \le d}$ fixes the n given directions for $1 \le j \le n$. We assume that the set of directions for our data type is known in advance, and is independent from the data itself. Thus, each data point x is located within a rational d-polytope, namely the intersection of the finite number of strips given by the above inequalities. In most applications, we only have the d directions of the coordinate axes, i.e. when each coordinate of an imprecisely given point is known to lie within an interval as in interval analysis, for example when the coordinates

[2] An open set is *regular* if it is the interior of its closure

of x are given by floating point numbers. But this data type is also essential in cases when we have lower and upper bounds on some linear combination of coordinates. In Figure 1, we have shown 6 out of the 18 possible types of polygons for an imprecisely given point in $\mathbb{R}^2$, where there are precisely three directions of possible approximations: along the two coordinate axes and along the $(1,1)$ vector, corresponding to the linear combination $x_1 + x_2$.

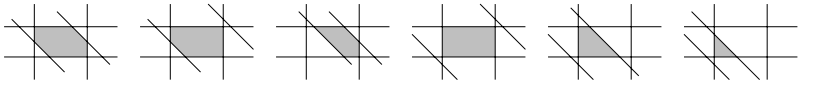


Fig. 1. Imprecise points defined by three directions $(1,0),(0,1),(1,1)$

Note that the filtered intersection of a non-empty family of convex d-polytopes in $\mathbb{R}^d$ is a non-empty, convex and compact subset. Our domain of computation is therefore the collection $(\mathbf{C}\mathbb{R}^d, \supseteq)$ of all non-empty, convex and compact subsets of $\mathbb{R}^d$ ordered by reverse inclusion and equipped with the Scott topology. It is a bounded complete ω-continuous domain with a countable basis given by the collection $\mathbf{P}\mathbb{R}^d$ of all rational convex d-polytopes in $\mathbb{R}^d$. The map $s : \mathbb{R}^d \to \mathbf{C}\mathbb{R}^d$ with $x \mapsto \{x\}$ is a topological embedding, i.e. we can identify the maximal elements of this domain with $\mathbb{R}^d$. We also note that $(\mathbf{C}\mathbb{R}^d, \supseteq)$ is order isomorphic with a sub-domain of $\mathbf{S}\mathbb{R}^d$ by identifying a non-empty convex and compact set $A \in \mathbf{C}\mathbb{R}^d$ with $(\emptyset, A^c) \in \mathbf{S}\mathbb{R}^d$, i.e. with a geometric object whose interior is empty and its exterior is the complement of A.

In this extended abstract, we restrict ourselves to computational geometry in $\mathbb{R}^2$; our results however extend to $\mathbb{R}^d$ as we will show in the full version of the paper.

2.1 Computability

We assume the reader is familiar with the notion of computability for continuous domains [10], [5], [8]. Recall that given an effective structure for a bounded complete ω-continuous dcpo with respect to an enumeration of a countable basis, a *computable element* is defined as the lub of an effective increasing sequence of basis elements. A *computable function* from a bounded complete ω-continuous dcpo with an effective structure to another such domain is a map which sends computable elements to computable elements in an effective way. We fix effective structures on $\mathbf{C}\mathbb{R}^2$ and $\mathbf{S}\mathbb{R}^2$ by using, for example, an enumeration of rational convex polygons as a basis of $\mathbf{C}\mathbb{R}^2$ and an enumeration of pairs of disjoint rational polygons as a basis of $\mathbf{S}\mathbb{R}^2$. These effective structures induce effective structures on $(\mathbf{C}\mathbb{R}^2)^N$ and $(\mathbf{S}\mathbb{R}^2)^N$ for all positive integers N.

We will define the notions of Hausdorff and Lebesgue computability in the solid domain. Let d_H denote the Hausdorff distance between non-empty compact sets. We put $d_H(\emptyset,\emptyset) = 0$ and for $Y \neq \emptyset$, $d_H(\emptyset, Y) = \infty$. The notion of Hausdorff

computability for a partial geometric object has been defined in [8]. Here, we define the notion of a nestedly Hausdorff computable map.

Definition 1. *Consider a computable map* $f : (\mathbf{C}[-a,a]^2)^N \to \mathbf{S}[-a,a]^2$, *for some* $a > 0$, *with* $f(\hat{C}) = (f_I(\hat{C}), f_E(\hat{C}))$, *where* $\hat{C} = (C_1, \ldots, C_N)$ *represents an ordered list of non-empty convex compact subsets of* $[-a,a]^2$. *Let* $\{\hat{B}_i \mid i \geq 0\}$ *be an enumeration of the basis of* $(\mathbf{C}[-a,a]^2)^N$. *Consider an arbitrary* $\hat{C} = \bigsqcup_{i\geq 0} \hat{B}_{\phi(n)}$ *with* $d_H(\hat{C}, \hat{B}_{\phi(i)}) < 2^{-i}$, *where* ϕ *is a total recursive function and* $\langle \hat{B}_{\phi(i)} \rangle_{i\geq 0}$ *is an increasing chain. We say the interior part* f_I *of* f *is* nestedly Hausdorff computable *if there exists a total recursive function* ψ_1 *such that*

$$d_H(\overline{(f_I(\hat{C}))}, \overline{f_I(\hat{B}_{\phi(\psi_1(i))})}) < 2^{-i} \text{ and } d_H((f_I(\hat{C}))^c, f_I(\hat{B}_{\phi(\psi_1(i))})^c) < 2^{-i}$$

where $\overline{A}$ *is the closure of* A *and complements are with respect to* $[-a,a]^2$. *Similarly for* f_E. *If both* f_I *and* f_E *are nestedly Hausdorff computable then we say that* f *is* nestedly Hausdorff computable.

As we will see later, the partial Delaunay triangulation map is nestedly Hausdorff computable but not Hausdorff continuous.

Proposition 1. *With the assumptions of Definition 1, suppose* $f_I(\hat{C})$ *and* $f_I(\hat{B}_{\phi(i)})$ *are regular. Then* f_I *is nestedly Hausdorff computable if there exists a total recursive function* ψ_1 *such that*

$$d_H(\partial(f_I(\hat{C})), \partial(f_I(\hat{B}_{\phi(\psi_1(i))}))) < 2^{-i},$$

where $\partial(A)$ *is the boundary of* A. *Similarly for* f_E.

Definition 2. *With the assumptions of Definition 1, we say* f_I *is* nestedly Lebesgue computable *if there exists a total recursive function* ψ_1 *such that*

$$\lambda(\overline{(f_I(\hat{C}))}, \overline{f_I(\hat{B}_{\phi(\psi_1(i))})}) < 2^{-i} \text{ and } \lambda((f_I(\hat{C}))^c, f_I(\hat{B}_{\phi(\psi_1(i))})^c) < 2^{-i}.$$

Similarly, for f_E. *If both* f_I *and* f_E *are nestedly Lebesgue computable then we say* f *is* nestedly Lebesgue computable.

Proposition 2. *With the assumptions of Definition 2, if the boundaries of* $f_I(\hat{B}_i)$ *and* $f_E(\hat{B}_i)$ *are continuous curves for each* $i \in \omega$ *and if their lengths are uniformly bounded, then* f *is Lebesgue computable.*

3 Convex Hull

The convex hull map for compact subsets is defined as:

$$\Gamma : (\mathcal{C}\mathbb{R}^2) \to \mathbf{C}\mathbb{R}^2$$

where $\mathcal{C}\mathbb{R}^2$ is the set of all non-empty compact subsets of $\mathbb{R}^2$ and $\mathbf{C}\mathbb{R}^2$ is the set of all non-empty compact convex subsets of $\mathbb{R}^2$, both with the Hausdorff metric; for any non-empty compact set C, the image $\Gamma(C)$ is the convex hull of C.

The partial convex hull map has type:

$$\begin{aligned} \mathcal{H} : (\mathbf{C}\mathbb{R}^2)^N &\to \quad \mathbf{S}\mathbb{R}^2 \\ \hat{C} &\mapsto (\mathcal{H}_I, \mathcal{H}_E), \end{aligned} \tag{1}$$

where $\hat{C} = (C_1, \ldots, C_N)$ represents an ordered list of N non-empty compact convex sets in the plane $\mathbb{R}^2$. For a given $\hat{C}$ define $R(\hat{C}) = \{\{p_1, \ldots, p_N\} \mid p_i \in C_i,\ i = 1, \ldots, N\}$ to be the collection of all possible N-element sets, each containing precisely one element from each C_i. An element $P \in R(\hat{C})$ is called a *representative set* for $\hat{C}$.

We define: $\mathcal{H}_I(\hat{C}) = (\bigcap_{P \in R(\hat{C})} \Gamma(P))^\circ$ and $\mathcal{H}_E(\hat{C}) = (\bigcup_{P \in R(\hat{C})} \Gamma(P))^c$. Thus, $\mathcal{H}_I$ is the set of points that are inside the convex hull of any representative set. Similarly, $\mathcal{H}_E$ is the set of points that are outside the convex hull of any representative set.

When $\hat{C} \in (\mathbf{C}\mathbb{R}^2)^N$ is a basis element, i.e. a list of N convex rational polygons, an algorithm has been developed [6] that computes $(\mathcal{H}_I, \mathcal{H}_E)$ as follows. Assume that there are n directions given by the unit normals d_j $(1 \le j \le n)$ with non-negative y coordinates, and ordered anti-clockwise from the positive x-axis. The unit circle is partitioned into $2n$ arcs $\widehat{d_j d_{j+1}}$ $(1 \le j \le 2n)$ with $d_{n+j} = -d_j$ for $1 \le j \le n$ and $d_{2n+1} = d_1$. Then, we have:

- $\mathcal{H}_E(\hat{C}) = (\Gamma(\{c \mid c \text{ is a corner of } C_i,\ 1 \le i \le N\}))^c$
- $\mathcal{H}_I(\hat{C}) = (\bigcap_{j=1}^{2n} \Gamma(\{c_{ij} \mid 1 \le i \le N\}))^\circ$,

where c_{ij} is a corner of C_i furthest away from the boundary of any half-plane containing C_i with unit normal in $\widehat{d_j d_{j+1}}$, see Figure 2. The above two expressions give an $N \log N$ algorithm to compute the interior and the exterior parts of the partial convex hull in rational arithmetic.

Theorem 1. *The exterior part of the convex hull map is nestedly Hausdorff computable.*

Theorem 2. *The map Γ is non-expansive with respect to the Hausdorff metric, i.e. $d_H(\Gamma(A), \Gamma(B)) \le d_H(A, B)$, and therefore Hausdorff continuous.*

Theorem 3. *The interior part of the partial convex hull map is nestedly Hausdorff computable.*

Corollary 1. *The partial convex hull map is nestedly Lebesgue computable.*

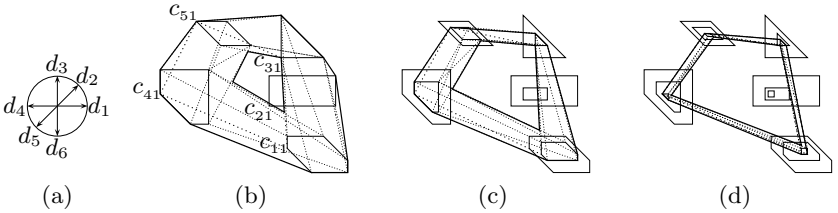


Fig. 2. (a) Three given directions have partitioned the unit circle into six arcs; (b) Partial convex hull of five partial points; (c)–(d) Partial convex hull with refined data

4 Partial Perpendicular Bisector

For a point $x \in \mathbb{R}^2$ and a compact $C \in \mathbf{C}\mathbb{R}^2$, we have the following distance functions appropriate for the Voronoi diagram of partial points. Let $d_s(x, C) = \min\{\|x - p\| : p \in C\}$ and $d_l(x, C) = \max\{\|x - p\| : p \in C\}$ be, respectively, the shortest and longest distance from x to C. For two compact subsets C_1 and C_2, we define the *partial Voronoi cell* of C_1 with respect to C_2 as:

$$\mathcal{C}_{12} = \{x \mid d_l(x, C_1) < d_s(x, C_2)\} \tag{2}$$

Similarly we define $\mathcal{C}_{21}$. The *partial perpendicular bisector* (PPB) of C_1 and C_2 is the remaining points of the plane, $\mathcal{B}(C_1, C_2) := (\mathcal{C}_{12} \cup \mathcal{C}_{21})^c = \{z \in \mathbb{R}^2 \mid \exists x \in P_1,\ y \in P_2;\ \|z - x\| = \|z - y\|\}$. The domain theoretic definition of the partial perpendicular bisector map is:

$$\begin{aligned} \mathcal{B} : \mathbf{C}\mathbb{R}^2 \times \mathbf{C}\mathbb{R}^2 &\to \mathbf{S}\mathbb{R}^2 \\ (C_1, C_2) &\mapsto (\emptyset, \mathcal{C}_{12} \cup \mathcal{C}_{21}). \end{aligned}$$

For basis elements $C_1, C_2 \in \mathbf{C}\mathbb{R}^2$, the boundary of $\mathcal{C}_{12} \cup \mathcal{C}_{21}$ consists of segments of parabolas and straight lines [6], see Figure 3(a).

Proposition 3. *The restriction of the partial perpendicular bisector map $\mathcal{B}$ to $\mathbf{C}[-a, a]^2$ is Hausdorff continuous for any $a > 0$.*

Proposition 4. *The partial perpendicular bisector map $\mathcal{B}$ is Scott continuous.*

Theorem 4. *The restriction of the partial perpendicular bisector map $\mathcal{B}$ to $\mathbf{C}[-a, a]^2$ is nestedly Hausdorff and Lebesgue computable for any $a > 0$.*

5 Partial Voronoi Diagram

We define the partial Voronoi map on a list $\hat{C} = (C_1, \ldots, C_N) \in (\mathbf{C}\mathbb{R}^2)^N$ of N polygons in the plane:

$$\mathcal{V} : (\mathbf{C}\mathbb{R}^2)^N \ \to\ (\mathbf{S}\mathbb{R}^2)^N,$$

with the ith component, $1 \le i \le N$, defined as

$$\mathcal{V}_i : \hat{C} \ \mapsto\ ((\mathcal{V}_i)_I, (\mathcal{V}_i)_E) = (\bigcap_{j \neq i} \mathcal{C}_{ij}, \bigcup_{j \neq i} \mathcal{C}_{ji}),$$

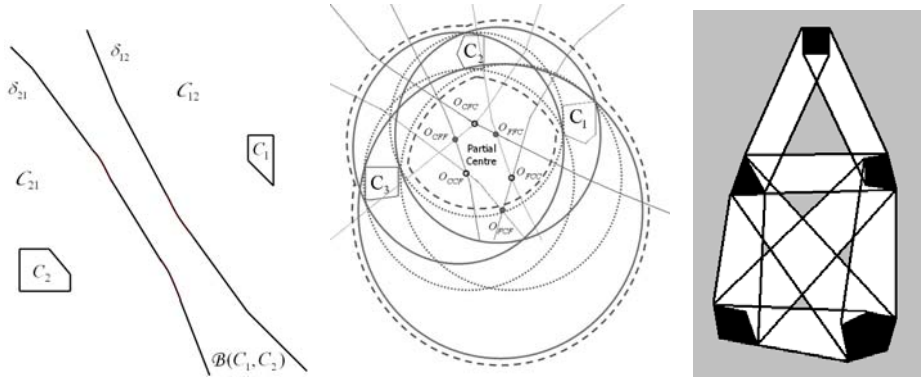
where $\mathcal{C}_{ji}$ is defined in Equation 2.

Proposition 5. *The restriction of the partial Voronoi diagram map $\mathcal{V}$ to $(\mathbf{C}[-a, a]^2)^N$ is Hausdorff continuous for any $a > 0$.*

Proposition 6. *The partial Voronoi diagram map $\mathcal{V}$ is Scott continuous.*

For a basis element $\hat{C} \in (\mathbf{C}\mathbb{R}^2)^N$, the boundaries of $\bigcap_{j \neq i} \mathcal{C}_{ij}$ and $\bigcup_{j \neq i} \mathcal{C}_{ji}$ consist of segments of parabolas and straight lines [6] as in the case of the partial perpendicular bisector.

Theorem 5. *The restriction of the partial Voronoi diagram map $\mathcal{V}$ to $(\mathbf{C}[-a, a]^2)^N$ is nestedly Hausdorff and Lebesgue computable for any $a > 0$.*



(a) (b) (c)

Fig. 3. (a) PPB of two polygons; (b) The interior and exterior of a partial disc; (c) The exterior of a partial Delaunay triangulation of five black polygons has been shown with gray colour. Note that there are two indeterminate and six legal edges

6 Partial Disc

Partial disc map has been defined by the authors in [6] as:

$$\begin{aligned} \mathcal{D}: \quad (\mathbf{C}\mathbb{R}^2)^3 \quad &\to \quad \mathbf{S}\mathbb{R}^2 \\ (C_1, C_2, C_3) &\mapsto (\mathcal{D}_I, \mathcal{D}_E), \end{aligned}$$

where $\mathcal{D}_I = \mathcal{D}_E = \emptyset$ if C_1, C_2 and C_3 are collinear, i.e. when there exists a straight line which intersects C_1, C_2 and C_3, otherwise $\mathcal{D}_I = (\bigcap\{D_{xyz} \mid x \in C_1,\ y \in C_2,\ z \in C_3\})^\circ$ and $\mathcal{D}_E = (\bigcup\{D_{xyz} \mid x \in C_1,\ y \in C_2,\ z \in C_3\})^c$, where D_{xyz} is the disc made by the circle passing through x, y, and z.

Note that $\mathcal{O}(C_1, C_2, C_3) = \{s \in \mathbb{R}^2 \mid \exists x \in C_1, y \in C_2, z \in C_3; \|x - s\| = \|y - s\| = \|z - s\|\}$, and hence $\mathcal{O}(C_1, C_2, C_3)$ is the locus of the centres of circles which intersect the three convex sets C_1, C_2 and C_3. We call $\mathcal{O}(C_1, C_2, C_3)$ the *partial centre* of the partial circumcircle of the three partial points.

Let $D(o_{CCF}, r_{CCF})$ denote the closed disc with centre o_{CCF} and radius r_{CCF}, which passes through the following three points: (i) the point of C_1 closest to o_{CCF}, (ii) the point of C_2 closest to o_{CCF} and (iii) the point of C_3 furthest from o_{CCF}; hence the subscript in o_{CCF}. Similarly, five other pairs of centres and radii are defined: (o_{CFC}, r_{CFC}), (o_{FCC}, r_{FCC}), (o_{FFC}, r_{FFC}), (o_{FCF}, r_{FCF}) and (o_{CFF}, r_{CFF}). Now, consider the three discs $D_1 = D(o_{FCC}, r_{FCC})$, $D_2 = D(o_{CFC}, r_{CFC})$ and $D_3 = D(o_{CCF}, r_{CCF})$ on the one hand and the three discs $D'_1 = D(o_{CFF}, r_{CFF})$, $D'_2 = D(o_{FCF}, r_{FCF})$ and $D'_3 = D(o_{FFC}, r_{FFC})$ on the other hand, Figure 3(b). As shown in [6] by the authors, the interior and the exterior of the partial disc are given by:

$$(\mathcal{D}_I, \mathcal{D}_E) = ((D_1 \cap D_2 \cap D_3)^\circ, (D_1' \cup D_2' \cup D_3')^c)\,.$$

Proposition 7. *The restriction of the partial disc map $\mathcal{D}$ to $(\mathbf{C}[-a,a]^2)^3$ is Hausdorff continuous for any $a > 0$.*

Proposition 8. *The partial disc map $\mathcal{D}$ is Scott continuous.*

For a basis element $(C_1, C_2, C_3) \in (\mathbf{C}\mathbb{R}^2)^3$, the centres and radii of the six discs above can be obtained using the partial perpendicular bisectors of each pair of these three partial points [6].

Theorem 6. *The restriction of the partial disc map $\mathcal{D}$ to $(\mathbf{C}[-a,a]^2)^3$ is nestedly Hausdorff and Lebesgue computable for any $a > 0$.*

7 Partial Delaunay Triangulation

We define the *partial edge* $\mathrm{Ed}(C_1, C_2)$ of two partial points C_1 and C_2 to be the convex hull of C_1 and C_2. We also define the *partial triangle* of three partial points to be their partial convex hull. Given N partial points $C_1, \ldots, C_N \in \mathbf{C}\mathbb{R}^2$, we say that $\mathrm{Ed}(C_{i_1}, C_{i_2})$ is *legal* if there exists i_3 such that for all $j \neq i_1, i_2, i_3$ we have $C_j \subset \mathcal{D}_E(C_{i_1}, C_{i_2}, C_{i_3})$, *illegal* if there exists i_3 such that there exists $j \neq i_1, i_2, i_3$ with $C_j \subset \mathcal{D}_I(C_{i_1}, C_{i_2}, C_{i_3})$ and *indeterminate* otherwise. The *partial Delaunay triangulation* map is now defined as:

$$\begin{aligned} \mathcal{T}: \quad (\mathbf{C}\mathbb{R}^2)^N \quad &\to \mathbf{S}\mathbb{R}^2 \\ (C_1, \ldots, C_N) &\mapsto (\mathcal{T}_I, \mathcal{T}_E), \end{aligned}$$

where $\mathcal{T}_I = \emptyset$ and

$$\mathcal{T}_E = (\bigcup \{\mathrm{Ed}(C_i, C_j) \mid \mathrm{Ed}(C_i, C_j) \text{ legal or indeterminate}\})^c.$$

We now proceed to show that the partial Delaunay triangulation map is nestedly Hausdorff computable. Since the interior is always empty, we only need to prove the computability for the exterior. In the example in Figure 3(c), the partial points are shown in black, while the exterior of the Delaunay triangulation, which is a disconnected set, is shown in gray. Note that the partial Delaunay triangulation map is not Hausdorff continuous, since an indeterminate partial edge may become illegal with an arbitrarily small non-nested perturbation of the input or partial points. The classical Delaunay triangulation map is similarly not Hausdorff continuous.

Proposition 9. *The partial Delaunay triangulation map $\mathcal{T}$ is Scott continuous.*

In [6], an incremental algorithm has been presented which computes the partial Delaunay triangulation of a set of partial points on average in $N \log N$ on

non-degenerate input, generalising a similar algorithm for the classical Delaunay triangulation.

Theorem 7. *The partial Delaunay triangulation map $\mathcal{T}$ is nestedly Hausdorff and Lebesgue computable.*

References

1. A. Bauer, L. Birkedal, and D. S. Scott. Equilogical spaces. *Theoretical Computer Science*, 315(1):35–59, 2004.
2. J. L. Berggren. *Episodes in the Mathematics of Medieval Islam.* Springer, 1986.
3. J. Blanck, V. Stoltenberg-Hansen, and J. V. Tucker. Domain representations of partial functions, with applications to spatial objects and constructive volume geometry. *Theoretical Computer Science*, 284:207–240, 2002.
4. V. Brattka and K. Weihrauch. Computability on subsets of Euclidean space I: Closed and compact subsets. *Theoretical Computer Science*, 219:65–93, 1999.
5. A. Edalat. Domains for computation in mathematics, physics and exact real arithmetic. *Bulletin of Symbolic Logic*, 3(4):401–452, 1997.
6. A. Edalat, A. A. Khanban, and A. Lieutier. Computational geometry with imprecise input data. Available from `http://www.doc.ic.ac.uk/~ae/papers/acm.ps`.
7. A. Edalat and A. Lieutier. Foundation of a computable solid modeling. In *Proceedings of the fifth symposium on Solid modeling and applications*, ACM Symposium on Solid Modeling and Applications, pages 278–284, 1999.
8. A. Edalat and A. Lieutier. Foundation of a computable solid modelling. *Theoretical Computer Science*, 284(2):319–345, 2002.
9. A. Edalat, A. Lieutier, and E. Kashefi. The convex hull in a new model of computation. In *Proc. 13th Canad. Conf. Comput. Geom.*, pages 93–96, 2001.
10. A. Edalat and P. Sünderhauf. A domain theoretic approach to computability on the real line. *Theoretical Computer Science*, 210:73–98, 1998.
11. X. Ge and A. Nerode. On extreme points of convex compact Turing located sets. In *Logical Foundations of Computer Science*, number 813 in LNCS, pages 114–128. Springer, 1994.
12. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer, 1988.
13. A. Grzegorczyk. On the definition of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.
14. A. A. Khanban, A. Edalat, and A. Lieutier. Computability of partial Delaunay triangulation and Voronoi diagram. In V. Brattka, M. Schröder, and K. Weihrauch, editors, *Electronic Notes in Theoretical Computer Science*, volume 66. Elsevier Science Publishers, 2002.
15. M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics.* Springer-Verlag, 1988.
16. V. Stoltenberg-Hansen and J. V. Tucker. Effective algebras. In D. G. S. Abramsky and T. S. E. M. eds., editors, *Handbook of Logic in Computer Science*, volume 4, pages 357–526. Oxford University Press, 1995.
17. D. J. Struik. Omar Khayyam, mathematician. *The Mathematics Teacher*, 51(4):280–285, 1958.

18. A. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proc. London Mathematical Soc.*, 42:230–265, 1936.
19. A. Turing. On computable numbers with an application to the Entscheidungsproblem (a correction). *Proc. London Mathematical Soc.*, 43:544–546, 1937.
20. K. Weihrauch. *Computable Analysis (An Introduction)*. Springer, 2000.
21. M. Ziegler. Computability on regular subsets of Euclidean space. *Math. Log. Q.*, 48(S1):157–181, 2002.
22. M. Ziegler. Computable operators on regular sets. *Mathematical Logic Quarterly*, pages 392–404, 2004.

SHRAD: A Language for Sequential Real Number Computation

Amin Farjudian

Department of Mathematical Sciences,
Sharif University of Technology,
Tehran, Iran
amin_farjudian@sina.sharif.edu

Since Di Gianantonio introduced his semantics for exact real number computation, there has always been a struggle to maintain data abstraction and efficiency as much as possible. The interval domain model — or its variations — can be regarded as the standard setting to obtain maximum data abstraction. As for efficiency there has been much focus on sequentiality to the extent that these two terms have become almost synonymous. [2, 3] demonstrated that there is not much one can get by sequential computation in the interval domain model. In [4, 5] we reinforced this result by exposing the limited power of (some extensions of) the sequential fragment of Real-PCF.

The previous argument suggests some sort of compromise in the beauty of the model in order to keep efficiency. One way forward is to try to sacrifice *extensionality*. This is exactly what we did in designing SHRAD [6]. There we succeeded in presenting a framework for exact real number computation which satisfies the following all at the same time:

1. It is sequential.
2. Multi-valuedness is carefully avoided.
3. A 'good degree' of expressivity is retained.

References

1. Di Gianantonio, P.: A Functional Approach to Computability on Real Numbers. PhD thesis, Università di Pisa-Genova-Udine (1993)
2. Escardó, M.H., Hofmann, M., Streicher, T.: Mediation is inherently parallel (1998) University of Edinburgh, Laboratory for Foundations of Computer Science, EPSRC report for project GR/M64840.
3. Escardó, M.H., Hofmann, M., Streicher, T.: On the non-sequential nature of the interval-domain model of exact real-number computation. Mathematical Structures in Computer Science (2004) Accepted for publication.
4. Farjudian, A.: Sequentiality and piecewise affinity in fragments of Real-PCF. To Appear in ENTCS (2003)
5. Farjudian, A.: Conservativity of wRPCF over PCF. Unpublished Manuscript (2003)
6. Farjudian, A.: Sequentiality in Real Number Computation. PhD thesis, School of Computer Science, The University of Birmingham, UK (2004) Dissertation.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, p. 128, 2005.

Borel Ranks and Wadge Degrees of Context Free ω-Languages

Olivier Finkel

Equipe de Logique Mathématique,
U.F.R. de Mathématiques, Université Paris 7,
2 Place Jussieu 75251 Paris cedex 05, France
finkel@logique.jussieu.fr.

Abstract. We determine completely the Borel hierarchy of the class of context free ω-languages, showing that, for each recursive non null ordinal α, there exist some $\mathbf{\Sigma}^0_\alpha$-complete and some $\mathbf{\Pi}^0_\alpha$-complete ω-languages accepted by Büchi 1-counter automata.

1 Introduction

Languages of infinite words accepted by finite automata were first studied by Büchi to prove the decidability of the monadic second order theory of one successor over the integers. The theory of the so called regular ω-languages is now well established and has found many applications for specification and verification of non-terminating systems; see [23, 21, 18] for many results and references. More powerful machines, like pushdown automata, Turing machines, have also been considered for the reading of infinite words, see Staiger's survey [21] and the fundamental study [6] of Engelfriet and Hoogeboom on $\mathbf{X}$-automata, i.e. finite automata equipped with a storage type $\mathbf{X}$. A way to study the complexity of ω-languages is to study their topological complexity, and particularly to locate them with regard to the Borel and the projective hierarchies. On one side all ω-languages accepted by *deterministic* $\mathbf{X}$-automata with a Muller acceptance condition are Boolean combinations of $\mathbf{\Pi}^0_2$-sets hence $\mathbf{\Delta}^0_3$-sets, [21, 6]. This implies, from Mc Naughton's Theorem, that all regular ω-languages, which are accepted by deterministic Muller automata, are also $\mathbf{\Delta}^0_3$-sets. On the other side, for *non deterministic* finite machines, the question, posed by Lescow and Thomas in [15], naturally arises: what is the topological complexity of ω-languages accepted by automata equipped with a given storage type $\mathbf{X}$? It is well known that every ω-language accepted by a Turing machine (hence also by a $\mathbf{X}$-automaton) with a Muller acceptance condition is an analytic set. In previous papers, we proved that there are context free ω-languages, accepted by Büchi or Muller pushdown automata, of every finite Borel rank, of infinite Borel rank, or even being analytic but non Borel sets, [3, 8, 10, 11]. In this paper we determine completely the Borel hierarchy of ω-languages accepted by $\mathbf{X}$-automata, for every storage type $\mathbf{X}$ such that 1-counter automata can be simulated by $\mathbf{X}$-automata. In particular, we show that, for every recursive non-null ordinal α, there are some $\mathbf{\Sigma}^0_\alpha$-complete

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 129–138, 2005.

and some $\mathbf{\Pi}^0_\alpha$-complete ω-languages accepted by real time 1-counter Büchi automata, hence also in the class CFL_ω of context free ω-languages.

We think that the surprising result obtained in this paper is of interest for both logicians working on hierarchies arising in recursion theory or in descriptive set theory, and also for computer scientists working on questions connected with non-terminating systems, like the construction of effective strategies in infinite games, [24, 27].

The paper is organized as follows. In Section 2 we define multicounter automata which will be a useful tool in the sequel. Recall on Borel hierarchy is given in Section 3. In Section 4 is studied the Borel hierarchy of ω-languages accepted by real time 8-counter automata. Our main result is proved in Section 5.

2 Multicounter Automata

We assume the reader to be familiar with the theory of formal (ω)-languages [23, 21]. We shall use usual notations of formal language theory.

When Σ is a finite alphabet, a *non-empty finite word* over Σ is any sequence $x = a_1 \dots a_k$, where $a_i \in \Sigma$ for $i = 1, \dots, k$, and k is an integer ≥ 1. The *length* of x is k, denoted by $|x|$. The *empty word* has no letter and is denoted by λ; its length is 0. For $x = a_1 \dots a_k$, we write $x(i) = a_i$ and $x[i] = x(1) \dots x(i)$ for $i \leq k$ and $x[0] = \lambda$. $\Sigma^\star$ is the *set of finite words* (including the empty word) over Σ.

The *first infinite ordinal* is ω. An *ω-word* over Σ is an ω -sequence $a_1 \dots a_n \dots$, where for all integers $i \geq 1$, $a_i \in \Sigma$. When σ is an ω-word over Σ, we write $\sigma = \sigma(1)\sigma(2) \dots \sigma(n) \dots$, where for all i, $\sigma(i) \in \Sigma$, and $\sigma[n] = \sigma(1)\sigma(2) \dots \sigma(n)$ for all $n \geq 1$ and $\sigma[0] = \lambda$.

The *prefix relation* is denoted $\sqsubseteq$: a finite word u is a *prefix* of a finite word v (respectively, an infinite word v), denoted $u \sqsubseteq v$, if and only if there exists a finite word w (respectively, an infinite word w), such that $v = u.w$. The *set of ω-words* over the alphabet Σ is denoted by Σ^ω. An *ω-language* over an alphabet Σ is a subset of Σ^ω.

Definition 1. *Let k be an integer ≥ 1. A k-counter machine (k-CM) is a 4-tuple $\mathcal{M} = (K, \Sigma, \Delta, q_0)$, where K is a finite set of states, Σ is a finite input alphabet, $q_0 \in K$ is the initial state, and the transition relation Δ is a subset of $K \times (\Sigma \cup \{\lambda\}) \times \{0,1\}^k \times K \times \{0,1,-1\}^k$. The k-counter machine $\mathcal{M}$ is said to be real time iff: $\Delta \subseteq K \times \Sigma \times \{0,1\}^k \times K \times \{0,1,-1\}^k$, i.e. iff there are not any λ-transitions.*

If the machine $\mathcal{M}$ is in state q and $c_i \in \mathbb{N}$ is the content of the i^{th} counter $\mathcal{C}_i$ then the configuration (or global state) of $\mathcal{M}$ is the $(k+1)$-tuple $(q, c_1, \dots, c_k)$.

For $a \in \Sigma \cup \{\lambda\}$, $q, q' \in K$ and $(c_1, \dots, c_k) \in \mathbb{N}^k$ such that $c_j = 0$ for $j \in E \subseteq \{1, \dots, k\}$ and $c_j > 0$ for $j \notin E$, if $(q, a, i_1, \dots, i_k, q', j_1, \dots, j_k) \in \Delta$ where $i_j = 0$ for $j \in E$ and $i_j = 1$ for $j \notin E$, then we write:

$$a : (q, c_1, \dots, c_k) \mapsto_{\mathcal{M}} (q', c_1 + j_1, \dots, c_k + j_k)$$

$\mapsto^\star_{\mathcal{M}}$ is the transitive and reflexive closure of $\mapsto_{\mathcal{M}}$. (The subscript $\mathcal{M}$ will be omitted whenever the meaning remains clear).

Thus we see that the transition relation must satisfy:
if $(q, a, i_1, \ldots, i_k, q', j_1, \ldots, j_k) \in \Delta$ *and* $i_m = 0$ *for some* $m \in \{1, \ldots, k\}$, *then* $j_m = 0$ *or* $j_m = 1$ *(but* j_m *may not be equal to* -1*).*

Let $\sigma = a_1 a_2 \ldots a_n$ *be a finite word over* Σ. *An sequence of configurations* $r = (q_i, c_1^i, \ldots c_k^i)_{1 \leq i \leq p}$, *for* $p \geq n+1$, *is called a run of* $\mathcal{M}$ *on* σ, *starting in configuration* $(p, c_1, \ldots, c_k)$, *iff:*

1. $(q_1, c_1^1, \ldots c_k^1) = (p, c_1, \ldots, c_k)$
2. *for each* $i \geq 1$, *there exists* $b_i \in \Sigma \cup \{\lambda\}$ *such that* $b_i : (q_i, c_1^i, \ldots c_k^i) \mapsto_{\mathcal{M}} (q_{i+1}, c_1^{i+1}, \ldots c_k^{i+1})$
3. $a_1.a_2.a_3 \ldots a_n = b_1.b_2.b_3 \ldots b_p$

Let $\sigma = a_1 a_2 \ldots a_n \ldots$ *be an* ω*-word over* Σ. *An* ω*-sequence of configurations* $r = (q_i, c_1^i, \ldots c_k^i)_{i \geq 1}$ *is called a run of* $\mathcal{M}$ *on* σ, *starting in configuration* $(p, c_1, \ldots, c_k)$, *iff:*

1. $(q_1, c_1^1, \ldots c_k^1) = (p, c_1, \ldots, c_k)$
2. *for each* $i \geq 1$, *there exists* $b_i \in \Sigma \cup \{\lambda\}$ *such that* $b_i : (q_i, c_1^i, \ldots c_k^i) \mapsto_{\mathcal{M}} (q_{i+1}, c_1^{i+1}, \ldots c_k^{i+1})$ *such that either* $a_1 a_2 \ldots a_n \ldots = b_1 b_2 \ldots b_n \ldots$ *or* $b_1 b_2 \ldots b_n \ldots$ *is a finite prefix of* $a_1 a_2 \ldots a_n \ldots$

The run r *is said to be complete when* $a_1 a_2 \ldots a_n \ldots = b_1 b_2 \ldots b_n \ldots$
For every such run, $\mathrm{In}(r)$ *is the set of all states entered infinitely often during run* r.

A complete run r *of* M *on* σ, *starting in configuration* $(q_0, 0, \ldots, 0)$, *will be simply called "a run of* M *on* σ*".*

Definition 2. *A Büchi* k*-counter automaton is a 5-tuple* $\mathcal{M} = (K, \Sigma, \Delta, q_0, F)$, *where* $\mathcal{M}' = (K, \Sigma, \Delta, q_0)$ *is a* k*-counter machine and* $F \subseteq K$ *is the set of accepting states. The* ω*-language accepted by* $\mathcal{M}$ *is*

$$L(\mathcal{M}) = \{\sigma \in \Sigma^\omega \mid \text{ there exists a run } r \text{ of } \mathcal{M} \text{ on } \sigma \text{ such that } \mathrm{In}(r) \cap F \neq \emptyset\}$$

Definition 3. *A Muller* k*-counter automaton is a 5-tuple* $\mathcal{M} = (K, \Sigma, \Delta, q_0, \mathcal{F})$, *where* $\mathcal{M}' = (K, \Sigma, \Delta, q_0)$ *is a* k*-counter machine and* $\mathcal{F} \subseteq 2^K$ *is the set of accepting sets of states. The* ω*-language accepted by* $\mathcal{M}$ *is*

$$L(\mathcal{M}) = \{\sigma \in \Sigma^\omega \mid \text{ there exists a run } r \text{ of } \mathcal{M} \text{ on } \sigma \text{ such that } \exists F \in \mathcal{F}\ \mathrm{In}(r) = F\}$$

The class of Büchi k-counter automata will be denoted $\mathbf{BC}(k)$.
The class of real time Büchi k-counter automata will be denoted $\mathbf{r\text{-}BC}(k)$.
The class of ω-languages accepted by Büchi k-counter automata will be denoted $\mathbf{BCL}(k)_\omega$.
The class of ω-languages accepted by real time Büchi k-counter automata will be denoted $\mathbf{r\text{-}BCL}(k)_\omega$.

It is well known that an ω-language is accepted by a (real time) Büchi k-counter automaton iff it is accepted by a (real time) Muller k-counter automaton [6]. Notice that it cannot be shown without using the non determinism of automata and this result is no longer true in the deterministic case.

Remark that 1-counter automata introduced above are equivalent to pushdown automata whose stack alphabet is in the form $\{Z_0, A\}$ where Z_0 is the bottom symbol which always remains at the bottom of the stack and appears only there and A is another stack symbol. The pushdown stack may be seen like a counter whose content is the integer N if the stack content is the word $Z_0.A^N$.

In the model introduced here the counter value cannot be increased by more than 1 during a single transition. However this does not change the class of ω-languages accepted by such automata. So the class $\mathbf{BCL}(1)_\omega$ is equal to the class $\mathbf{1\text{-}ICL}_\omega$, introduced in [9], and it is a strict subclass of the class $\mathbf{CFL}_\omega$ of context free ω-languages accepted by Büchi pushdown automata.

3 Borel Hierarchy

We assume the reader to be familiar with basic notions of topology which may be found in [16, 15, 14, 21, 18]. There is a natural metric on the set Σ^ω of infinite words over a finite alphabet Σ which is called the *prefix metric* and defined as follows. For $u, v \in \Sigma^\omega$ and $u \neq v$ let $\delta(u, v) = 2^{-l_{\mathrm{pref}(u,v)}}$ where $l_{\mathrm{pref}(u,v)}$ is the first integer n such that the $(n+1)^{st}$ letter of u is different from the $(n+1)^{st}$ letter of v. This metric induces on Σ^ω the usual Cantor topology for which *open subsets* of Σ^ω are in the form $W.\Sigma^\omega$, where $W \subseteq \Sigma^\star$. A set $L \subseteq \Sigma^\omega$ is a *closed set* iff its complement $\Sigma^\omega - L$ is an open set. Define now the *Borel Hierarchy* of subsets of Σ^ω:

Definition 4. *For a non-null countable ordinal α, the classes $\mathbf{\Sigma}^0_\alpha$ and $\mathbf{\Pi}^0_\alpha$ of the Borel Hierarchy on the topological space Σ^ω are defined as follows:*
$\mathbf{\Sigma}^0_1$ is the class of open subsets of Σ^ω, $\mathbf{\Pi}^0_1$ is the class of closed subsets of Σ^ω, and for any countable ordinal $\alpha \geq 2$:
$\mathbf{\Sigma}^0_\alpha$ is the class of countable unions of subsets of Σ^ω in $\bigcup_{\gamma<\alpha} \mathbf{\Pi}^0_\gamma$.
$\mathbf{\Pi}^0_\alpha$ is the class of countable intersections of subsets of Σ^ω in $\bigcup_{\gamma<\alpha} \mathbf{\Sigma}^0_\gamma$.

For a countable ordinal α, a subset of Σ^ω is a Borel set of *rank* α iff it is in $\mathbf{\Sigma}^0_\alpha \cup \mathbf{\Pi}^0_\alpha$ but not in $\bigcup_{\gamma<\alpha}(\mathbf{\Sigma}^0_\gamma \cup \mathbf{\Pi}^0_\gamma)$.

There are also some subsets of Σ^ω which are not Borel. In particular the class of Borel subsets of Σ^ω is strictly included into the class $\mathbf{\Sigma}^1_1$ of *analytic sets* which are obtained by projection of Borel sets, see for example [21, 15, 18, 14] for more details. The (lightface) class Σ^1_1 of *effective analytic sets* is the class of sets which are obtained by projection of arithmetical sets. It is well known that a set $L \subseteq \Sigma^\omega$, where Σ is a finite alphabet, is in the class Σ^1_1 iff it is accepted by a Turing machine with a Büchi or Muller acceptance condition [21].

We now define completeness with regard to reduction by continuous functions. For a countable ordinal $\alpha \geq 1$, a set $F \subseteq \Sigma^\omega$ is said to be a $\mathbf{\Sigma}^0_\alpha$ (respectively, $\mathbf{\Pi}^0_\alpha$, $\mathbf{\Sigma}^1_1$)-*complete set* iff for any set $E \subseteq Y^\omega$ (with Y a finite alphabet): $E \in \mathbf{\Sigma}^0_\alpha$ (respectively, $E \in \mathbf{\Pi}^0_\alpha$, $E \in \mathbf{\Sigma}^1_1$) iff there exists a continuous function $f : Y^\omega \to \Sigma^\omega$ such that $E = f^{-1}(F)$. $\mathbf{\Sigma}^0_n$ (respectively $\mathbf{\Pi}^0_n$)-complete sets, with n an integer ≥ 1, are thoroughly characterized in [20].

4 Borel Hierarchy of ω-Languages in r-BCL(8)$_\omega$

It is well known that if $L \subseteq \Sigma^\omega$ is accepted by a Turing machine with a Büchi acceptance condition and is a Borel set of rank α, then α is smaller than $\omega_1^{\rm CK}$, where $\omega_1^{\rm CK}$ is the first non-recursive ordinal, usually called the Church-Kleene ordinal. Moreover for every non null countable ordinal $\alpha < \omega_1^{\rm CK}$, there exist some $\mathbf{\Sigma}^0_\alpha$-complete and some $\mathbf{\Pi}^0_\alpha$-complete sets in the class Σ^1_1 of ω-languages accepted by Turing machines with a Büchi acceptance condition.

On the other hand it is well known that every Turing machine can be simulated by a (non real time) 2-counter automaton. Thus for every non null countable ordinal $\alpha < \omega_1^{\rm CK}$, there exist some $\mathbf{\Sigma}^0_\alpha$-complete and some $\mathbf{\Pi}^0_\alpha$-complete ω-languages in the class $\mathbf{BCL}(2)_\omega$. We shall prove the following proposition.

Proposition 5. *For every non null countable ordinal $\alpha < \omega_1^{\rm CK}$, there exist some $\mathbf{\Sigma}^0_\alpha$-complete and some $\mathbf{\Pi}^0_\alpha$-complete ω-languages in the class* $\mathbf{r\text{-}BCL}(8)_\omega$.

In order to prove this result, we first state the two following lemmas.

Let Σ be an alphabet having at least two letters, E be a new letter not in Σ, S be an integer ≥ 1, and $\theta_S : \Sigma^\omega \to (\Sigma \cup \{E\})^\omega$ be the function defined, for all $x \in \Sigma^\omega$, by:

$$\theta_S(x) = x(1).E^S.x(2).E^{S^2}.x(3).E^{S^3}.x(4)\ldots x(n).E^{S^n}.x(n+1).E^{S^{n+1}}\ldots$$

Lemma 6. *Let Σ be an alphabet having at least two letters and let $L \subseteq \Sigma^\omega$ be a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) subset of Σ^ω for some ordinal $\alpha \geq 2$. Then the ω-language $\theta_S(L)$ is a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) subset of $(\Sigma \cup \{S\})^\omega$.*

Lemma 7. *Let Σ be an alphabet having at least two letters and let $L \subseteq \Sigma^\omega$ be an ω-language in the class $\mathbf{BCL}(2)_\omega$. Then there exists an integer $S \geq 1$ such that $\theta_S(L)$ is in the class* $\mathbf{r\text{-}BCL}(8)_\omega$.

5 Borel Hierarchy of ω-Languages in r-BCL(1)$_\omega$

We shall firstly prove the following result.

Proposition 8. *Let $k \geq 2$ be an integer. If, for some ordinal $\alpha \geq 2$, there is a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) ω-language in the class $\mathbf{r\text{-}BCL}(k)_\omega$, then there is some $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) ω-language in the class $\mathbf{r\text{-}BCL}(1)_\omega$.*

To simplify the exposition of the proof of this result, firstly, we are going to sketch the proof for $k = 2$. Next we shall explain the modifications to do in order to infer the result for the integer $k = 8$ which is in fact the only case we shall need in the sequel. (However our main result will show that the proposition is true for every integer $k \geq 2$).

For that purpose we define first a coding of ω-words over a finite alphabet Σ by ω-words over the alphabet $\Sigma \cup \{A, B, 0\}$ where A, B and 0 are new letters not in Σ. We shall code an ω-word $x \in \Sigma^\omega$ by the ω-word $h(x)$ defined by

$$h(x) = A.0^6.x_1.B.0^{6^2}.A.0^{6^2}.x_2.B.0^{6^3}.A.0^{6^3}.x_3.B \ldots B.0^{6^n}.A.0^{6^n}.x_n.B \ldots$$

This coding defines a mapping $h : \Sigma^\omega \to (\Sigma \cup \{A, B, 0\})^\omega$. The function h is continuous because for all ω-words $x, y \in \Sigma^\omega$ and each positive integer n, it holds that $\delta(x, y) < 2^{-n} \to \delta(h(x), h(y)) < 2^{-n}$.

Lemma 9. *Let Σ be a finite alphabet and $(h(\Sigma^\omega))^- = (\Sigma \cup \{A, B, 0\})^\omega - h(\Sigma^\omega)$. If $\mathcal{L} \subseteq \Sigma^\omega$ is $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete), for a countable ordinal $\alpha \geq 2$, then $h(\mathcal{L}) \cup h(\Sigma^\omega)^-$ is a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) subset of $(\Sigma \cup \{A, B, 0\})^\omega$.*

In order to apply Lemma 9, we want now to prove that if $L(\mathcal{A}) \subseteq \Sigma^\omega$ is accepted by a real time 2-counter automaton $\mathcal{A}$ with a Büchi acceptance condition then $h(L(\mathcal{A})) \cup h(\Sigma^\omega)^-$ is accepted by a 1-counter automaton with a Büchi acceptance condition. We firstly prove the following lemma.

Lemma 10. *Let Σ be a finite alphabet and h be the coding of ω-words over Σ defined as above. Then $h(\Sigma^\omega)^- = (\Sigma \cup \{A, B, 0\})^\omega - h(\Sigma^\omega)$ is accepted by a real time 1-counter Büchi automaton.*

We would like now to prove that if $L(\mathcal{A}) \subseteq \Sigma^\omega$ is accepted by a real time 2-counter automaton $\mathcal{A}$ with a Büchi acceptance condition then $h(L(\mathcal{A}))$ is in $\mathbf{BCL}(1)_\omega$. We cannot show this, so we are firstly going to define another ω-language $\mathcal{L}(\mathcal{A})$ accepted by a 1-counter Büchi automaton and we shall prove that $h(L(\mathcal{A})) \cup h(\Sigma^\omega)^- = \mathcal{L}(\mathcal{A}) \cup h(\Sigma^\omega)^-$.

We shall need the following notion. Let $N \geq 1$ be an integer such that $N = 2^x.3^y.N_1$ where x, y are positive integers and $N_1 \geq 1$ is an integer which is neither divisible by 2 nor by 3. Then we set $P_2(N) = x$ and $P_3(N) = y$. So $2^{P_2(N)}$ is the greatest power of 2 which divides N and $2^{P_3(N)}$ is the greatest power of 3 which divides N.

Let then a 2-counter Büchi automaton $\mathcal{A} = (K, \Sigma, \Delta, q_0, F)$ accepting the ω-language $L(\mathcal{A}) \subseteq \Sigma^\omega$. The ω-language $\mathcal{L}(\mathcal{A})$ is the set of ω-words over the alphabet $\Sigma \cup \{A, B, 0\}$ in the form

$$A.u_1.v_1.x_1.B.w_1.z_1.A.u_2.v_2.x_2.B.w_2.z_2.A \ldots A.u_n.v_n.x_n.B.w_n.z_n.A \ldots$$

where, for all integers $i \geq 1$, $v_i, w_i \in 0^+$, $u_i, z_i \in 0^\star$, $|u_1| = 5$, $|u_{i+1}| = |z_i|$ and there is a sequence $(q_i)_{i \geq 0}$ of states of K and integers $j_i, j'_i \in \{-1; 0; 1\}$, for $i \geq 1$, such that for all integers $i \geq 1$:

$$x_i : (q_{i-1}, P_2(|v_i|), P_3(|v_i|)) \mapsto_{\mathcal{A}} (q_i, P_2(|v_i|) + j_i, P_3(|v_i|) + j'_i)$$

and

$$|w_i| = |v_i|.2^{j_i}.3^{j'_i}$$

Moreover some state $q_f \in F$ occurs infinitely often in the sequence $(q_i)_{i \geq 0}$. Notice that the state q_0 of the sequence $(q_i)_{i \geq 0}$ is also the initial state of $\mathcal{A}$.

Lemma 11. *Let $\mathcal{A}$ be a real time 2-counter Büchi automaton accepting ω-words over the alphabet Σ and $\mathcal{L}(\mathcal{A}) \subseteq (\Sigma \cup \{A, B, 0\})^\omega$ be defined as above. Then $\mathcal{L}(\mathcal{A})$ is accepted by a 1-counter Büchi automaton $\mathcal{B}$.*

Lemma 12. *Let $\mathcal{A}$ be a real time 2-counter Büchi automaton accepting ω-words over the alphabet Σ and $\mathcal{L}(\mathcal{A}) \subseteq (\Sigma \cup \{A, B, 0\})^\omega$ be defined as above. Then $L(\mathcal{A}) = h^{-1}(\mathcal{L}(\mathcal{A}))$, i.e. $\forall x \in \Sigma^\omega \quad h(x) \in \mathcal{L}(\mathcal{A}) \longleftrightarrow x \in L(\mathcal{A})$.*

Proof. Let $\mathcal{A}$ be a real time 2-counter Büchi automaton accepting ω-words over the alphabet Σ and $\mathcal{L}(\mathcal{A}) \subseteq (\Sigma \cup \{A, B, 0\})^\omega$ be defined as above. Let $x \in \Sigma^\omega$ be an ω-word such that $h(x) \in \mathcal{L}(\mathcal{A})$. So $h(x)$ may be written

$$h(x) = A.0^6.x_1.B.0^{6^2}.A.0^{6^2}.x_2.B.0^{6^3}.A.0^{6^3}.x_3.B \ldots B.0^{6^n}.A.0^{6^n}.x_n.B \ldots$$

and also

$$h(x) = A.u_1.v_1.x_1.B.w_1.z_1.A.u_2.v_2.x_2.B.w_2.z_2.A \ldots A.u_n.v_n.x_n.B.w_n.z_n.A \ldots$$

where, for all integers $i \geq 1$, $v_i, w_i \in 0^+$, $u_i, z_i \in 0^\star$, $|u_1| = 5$, $|u_{i+1}| = |z_i|$ and there is a sequence $(q_i)_{i \geq 0}$ of states of K and integers $j_i, j'_i \in \{-1; 0; 1\}$, for $i \geq 1$, such that for all integers $i \geq 1$:

$$x_i : (q_{i-1}, P_2(|v_i|), P_3(|v_i|)) \mapsto_{\mathcal{A}} (q_i, P_2(|v_i|) + j_i, P_3(|v_i|) + j'_i)$$

and

$$|w_i| = |v_i|.2^{j_i}.3^{j'_i}$$

some state $q_f \in F$ occurring infinitely often in the sequence $(q_i)_{i \geq 0}$.

In particular, $u_1 = 0^5$ and $u_1.v_1 = 0^6$ thus $|v_1| = 1 = 2^0.3^0$. We can prove by induction on the integer $i \geq 1$ that, for all integers $i \geq 1$, $|w_i| = |v_{i+1}| = 2^{P_2(|w_i|)}.3^{P_3(|w_i|)}$. Moreover, setting $c_1^i = P_2(|v_i|)$ and $c_2^i = P_3(|v_i|)$, we can prove that for each integer $i \geq 1$ it holds that

$$x_i : (q_{i-1}, c_1^i, c_2^i) \mapsto_{\mathcal{A}} (q_i, c_1^{i+1}, c_2^{i+1})$$

But there is some state $q_f \in K$ which occurs infinitely often in the sequence $(q_i)_{i \geq 1}$. This implies that $(q_{i-1}, c_1^i, c_2^i)_{i \geq 1}$ is a successful run of $\mathcal{A}$ on x thus $x \in L(\mathcal{A})$.

Conversely it is easy to see that if $x \in L(\mathcal{A})$ then $h(x) \in \mathcal{L}(\mathcal{A})$. This ends the proof of Lemma 12. □

Remark 13. *The simulation, during the reading of $h(x)$ by the 1-counter Büchi automaton $\mathcal{B}$, of the behaviour of the real time 2-counter Büchi automaton $\mathcal{A}$ reading x, can be achieved, using a coding of the content (c_1, c_2) of two counters by a single integer $2^{c_1}.3^{c_2}$ and the* **special shape** *of ω-words in $h(\Sigma^\omega)$ which allows the propagation of the counter value of $\mathcal{B}$. This will be sufficient here, because of the previous lemmas, and in particular of the fact that $h(\Sigma^\omega)^-$ is in the class $\mathbf{r}\text{-}\mathbf{BCL}(1)_\omega$. and we can now end the proof of Proposition 8.*

End of Proof of Proposition 8. Let $\alpha \geq 2$ be an ordinal. Assume that there is a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) ω-language $L(\mathcal{A}) \subseteq \Sigma^\omega$ which is accepted by a real time 2-counter Büchi automaton $\mathcal{A}$. By Lemma 9, $h(\mathcal{L}) \cup h(\Sigma^\omega)^-$ is a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) subset of $(\Sigma \cup \{A, B, 0\})^\omega$. On the other hand Lemma 12 states that $L(\mathcal{A}) = h^{-1}(\mathcal{L}(\mathcal{A}))$ and this implies that $h(L(\mathcal{A})) \cup h(\Sigma^\omega)^- = \mathcal{L}(\mathcal{A}) \cup h(\Sigma^\omega)^-$. But we know by Lemmas 10 and 11 that the ω-languages $h(\Sigma^\omega)^-$ and $\mathcal{L}(\mathcal{A})$ are in the class $\mathbf{BCL}(1)_\omega$ thus their union is also accepted by a 1-counter Büchi automaton. Therefore $h(L(\mathcal{A})) \cup h(\Sigma^\omega)^-$ is a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) ω-language in the class $\mathbf{BCL}(1)_\omega$.

We can now easily show that there is a $\mathbf{\Sigma}^0_\alpha$-complete (respectively, $\mathbf{\Pi}^0_\alpha$-complete) ω-language in the class $\mathbf{r\text{-}BCL}(1)_\omega$, using the two following facts. (1) $h(\Sigma^\omega)^-$ is accepted by a *real time* 1-counter Büchi automaton; (2) $\mathcal{L}(\mathcal{A})$ is accepted by a (non real time) 1-counter Büchi automaton $\mathcal{B}$ but at most 5 consecutive λ-transitions can occur during the reading of an ω-word x by $\mathcal{B}$ (see the proof of Lemma 11 in the full version of this paper).

In order to prove Proposition 8 for the integer $k = 8$, we reason in a similar way. We first replace the integer $6 = 2.3$ by the product of the eight first prime numbers:

$$K = 2.3.5.7.11.13.17.19 = 9699690$$

and the mapping h by the mapping h_K, defined for all $x \in \Sigma^\omega$, by:

$$h_K(x) = A.0^K.x_1.B.0^{K^2}.A.0^{K^2}.x_2.B.0^{K^3}.A.0^{K^3}.x_3.B \ldots B.0^{K^n}.A.0^{K^n}.x_n.B \ldots$$

We define also, for every 8-counter Büchi automaton $\mathcal{A}$, an ω-language $\mathcal{L}(\mathcal{A})$, accepted by a 1-counter Büchi automaton, such that $L(\mathcal{A}) = h_K^{-1}(\mathcal{L}(\mathcal{A}))$.

The essential change is that now the content $(c_1, c_2, \ldots, c_8)$ of eight counters is coded by the product $2^{c_1}.3^{c_2}. \ldots .(17)^{c_7}.(19)^{c_8}$.

Details will be included in the full version of this paper. □

From the results of Section 4 and Proposition 8, we can now state the following result.

Theorem 14. *Let $\mathcal{C}$ be a class of ω-languages such that:*

$$\mathbf{r\text{-}BCL}(1)_\omega \subseteq \mathcal{C} \subseteq \Sigma^1_1.$$

(a) If $L \in \mathcal{C}$ is a Borel set of rank α, then α is smaller than ω_1^{CK}.
(b) For every non null countable ordinal $\alpha < \omega_1^{\mathrm{CK}}$, there exists some $\mathbf{\Sigma}^0_\alpha$-complete and some $\mathbf{\Pi}^0_\alpha$-complete ω-languages in the class $\mathcal{C}$.

The Wadge hierarchy is a great refinement of the Borel hierarchy, [4, 25]. Looking carefully at the proofs given in this paper, we can easily show the following strengthening of Theorem 14 (see the proof in the full version of this paper).

Theorem 15. *The Wadge hierarchy of the class $\mathbf{r\text{-}BCL}(1)_\omega$, hence also of the class CFL_ω, or of every class $\mathcal{C}$ such that $\mathbf{r\text{-}BCL}(1)_\omega \subseteq \mathcal{C} \subseteq \Sigma^1_1$, is the Wadge hierarchy of the class Σ^1_1 of ω-languages accepted by Turing machines with a Büchi acceptance condition.*

6 Concluding Remarks

We have completely determined the Borel hierarchy of classes $\mathbf{r\text{-}BCL}(1)_\omega$ and CFL_ω and showed that their Wadge hierarchy is also the Wadge hierarchy of the class Σ^1_1. The methods used in this paper are different from those used in previous papers on context free ω-languages [8, 7, 10, 11], where we gave an inductive construction of some $\mathbf{\Delta}^0_\omega$ context free ω-languages of a given Borel rank or Wadge degree, using work of Duparc on the Wadge hierarchy of $\mathbf{\Delta}^0_\omega$ Borel sets, [4]. However it will be possible to combine both methods for the effective construction of ω-languages in the class $\mathbf{r\text{-}BCL}(1)_\omega$, and of 1-counter Büchi automata accepting them, of a given Wadge degree among the ε_ω degrees obtained in [7] for $\mathbf{\Delta}^0_\omega$ context free ω-languages.

Finally we mention that in a forthcoming paper we apply similar methods to the study of topological properties of infinitary rational relations and we prove that their Wadge and Borel hierarchies are equal to the corresponding hierarchies of of the classes $\mathbf{r\text{-}BCL}(1)_\omega$, CFL_ω or Σ^1_1, [12].

References

1. J. Berstel. *Transductions and context free languages.* Teubner Studienbücher Informatik, 1979.
2. R.S. Cohen and A.Y. Gold. Theory of ω-languages, parts one and two. *Journal of Computer and System Science*, 15:169–208, 1977.
3. J. Duparc, O. Finkel, and J.-P. Ressayre. Computer science and the fine structure of Borel sets. *Theoretical Computer Science*, 257(1–2):85–105, 2001.
4. J. Duparc. Wadge hierarchy and Veblen hierarchy: Part 1: Borel sets of finite rank. *Journal of Symbolic Logic*, 66(1):56–86, 2001.
5. J. Duparc. A hierarchy of deterministic context free ω-languages. *Theoretical Computer Science*, 290(3):1253–1300, 2003.
6. J Engelfriet and H. J. Hoogeboom. X-automata on ω-words. *Theoretical Computer Science*, 110(1):1–51, 1993.
7. O. Finkel. On the Wadge hierarchy of omega context free languages. In *Proceedings of the International Workshop on Logic and Complexity in Computer Science, held in Honor of Anatol Slissenko for his 60th Birthday*, pages 69–79, Créteil, France, 2001.
8. O. Finkel. Topological properties of omega context free languages. *Theoretical Computer Science*, 262(1–2):669–697, 2001.
9. O. Finkel. Wadge hierarchy of omega context free languages. *Theoretical Computer Science*, 269(1–2):283–315, 2001.
10. O. Finkel. Borel hierarchy and omega context free languages. *Theoretical Computer Science*, 290(3):1385–1405, 2003.
11. O. Finkel. On omega context free languages which are Borel sets of infinite rank. *Theoretical Computer Science*, 299(1-3):327–346, 2003.
12. O. Finkel. On the accepting power of two-tape Büchi automata. 2004. preprint.
13. J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.
14. A. S. Kechris. *Classical descriptive set theory.* Springer-Verlag, New York, 1995.

15. H. Lescow and W. Thomas. Logical specifications of infinite computations. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *A Decade of Concurrency*, volume 803 of *Lecture Notes in Computer Science*, pages 583–621. Springer, 1994.
16. Y. N. Moschovakis. *Descriptive set theory*. North-Holland Publishing Co., Amsterdam, 1980.
17. M. Nivat. Sur les ensembles de mots infinis engendrés par une grammaire algébrique. *RAIRO Informatique Théorique et Applications*, 12(3):259–278, v, 1978.
18. D. Perrin and J.-E. Pin. *Infinite words, automata, semigroups, logic and games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.
19. P. Simonnet. *Automates et théorie descriptive*. PhD thesis, Université Paris VII, 1992.
20. L. Staiger. Hierarchies of recursive ω-languages. *Elektronische Informationsverarbeitung und Kybernetik*, 22(5-6):219–241, 1986.
21. L. Staiger. ω-languages. In *Handbook of formal languages, Vol. 3*, pages 339–387. Springer, Berlin, 1997.
22. L. Staiger and K. Wagner. Automatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulärer Folgenmengen. *Elektron. Informationsverarbeit. Kybernetik*, 10:379–392, 1974.
23. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, Formal models and semantics, pages 135–191. Elsevier, 1990.
24. W. Thomas. Infinite games and verification (extended abstract of a tutorial). In *Proceedings of the International Conference on Computer Aided Verification CAV 2002*, volume 2404 of *Lecture Notes in Computer Science*, pages 58–64. Springer, 2002.
25. W. Wadge. *Reducibility and determinateness in the Baire space*. PhD thesis, University of California, Berkeley, 1983.
26. K. Wagner. On ω-regular sets. *Information and Control*, 43(2):123–177, 1979.
27. I. Walukiewicz. Pushdown processes: games and model checking. *Information and Computation*, 157:234–263, 2000.

Fewer Epistemological Challenges for Connectionism

Artur S. d'Avila Garcez*

Dept. of Computing
School of Informatics
City University
London EC1V OHB, UK
aag@soi.city.ac.uk

Abstract. Seventeen years ago, John McCarthy wrote the note *Epistemological challenges for connectionism* as a response to Paul Smolensky's paper *On the proper treatment of connectionism*. I will discuss the extent to which the four key challenges put forward by McCarthy have been solved, and what are the new challenges ahead. I argue that there are fewer epistemological challenges for connectionism, but progress has been slow. Nevertheless, there is now strong indication that neural-symbolic integration can provide effective systems of expressive reasoning and robust learning due to the recent developments in the field.

1 Introduction

This paper is about the integration of neural networks and symbol processing; it is about how to represent, learn, and compute expressive forms of symbolic knowledge using neural networks. I believe this is the way forward towards the provision of an integrated system of expressive reasoning and robust learning. The provision of such a system, integrating the two most fundamental phenomena of intelligent cognitive behaviour (i.e. the ability to learn from experience and the ability to reason from what has been learned) has been recently identified by Leslie Valiant as a key challenge for computer science [25]. The goal is to produce biologically plausible models with integrated reasoning and learning capability, in which neural networks provide the inspiration and the machinery necessary for cognitive computation and learning, while logics provide practical reasoning and explanation capabilities to the models, facilitating the interaction between them and the outside world.

In what follows, I will briefly review my recent work (joint with Luis Lamb and Dov Gabbay) on how to integrate logic and neural networks [8, 9]. I will then address the open question of how to represent variables effectively in neural networks, which emerges from my recent work (joint with Dov Gabbay) on how to

* I'm grateful to Luis C. Lamb for useful discussions. This work was partly supported by The Nuffield Foundation UK.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 139–149, 2005.

combine neural networks in a principled way [7]. Throughout, I will try and put the recent advances on neural-symbolic integration in the context of John McCarthy's note *Epistemological challenges for connectionism* [17], written as a response to Paul Smolensky's paper *On the proper treatment of connectionism* [22]. Briefly, McCarthy identifies four knowledge representation problems for neural networks: the problem of *elaboration tolerance* (the ability of a representation to be elaborated to take additional phenomena into account); the *propositional fixation* of neural networks (based on the assumption that neural networks cannot represent relational knowledge); the problem of how to make use of any available *background knowledge* as part of learning, and the problem of how to obtain domain *descriptions* from trained networks as opposed to mere discriminations.

I will start by giving examples of how we represent propositional modal logic (and thus relational knowledge) in neural networks, pointing the reader to the papers in the area. I will then discuss our proposal for combining (fibring) neural networks, and how it may allow us to represent variables. In what regards the challenges put forward by McCarthy, in a nutshell, the problem of elaboration tolerance may be resolved by having networks that are fibred in a hierarchy (this is similar to the idea of using self-organising maps [12], e.g., for language processing, in which the lower levels of abstraction are used for the formation of concepts that are then used at the higher levels of the hierarchy); in the case of the so-called propositional fixation of neural networks, connectionist modal logic shows that, as a matter of fact, neural networks can encode relational knowledge (in the form of accessibility relations) [9]; as for learning with background knowledge, this can be achieved by translating symbolic rules into the initial architecture of a neural network; whereas problem description can be obtained by rule extraction from trained neural networks. In the past decade, a number of such translation algorithms [8, 9, 13, 23] and knowledge extraction algorithms [1, 3, 19, 24] has been proposed.

Nevertheless, there are still challenges ahead, particularly in what regards the effective integration of expressive reasoning and robust learning. In this case, we cannot afford to lose on the learning capability side as we add reasoning capability to neural networks. This means that we cannot depart from the idea that neural networks are composed of simple processing units organised in a massively parallel way (and allow for some *clever* neurons to perform complex symbolic computation). We also would like our models to be biologically plausible, not as a principle but in a pragmatic way. There have been recent advances in brain imaging, which offer us data we can make use of to get insight into new forms of representation. Human beings are quite extraordinary at performing practical reasoning as they go about their daily business. *There are cases where the human computer, slow as it is, is faster than Artificial Intelligence systems. Why are we faster? Is it the way we perceive knowledge as opposed to the way we represent it? Do we know immediately which rules to select and apply? We must look for the correct representation in the sense that it mirrors the way we perceive and apply the rules* [10]. Ultimately, Neural-Symbolic integration is about asking and trying to answer these questions.

2 Neural-Symbolic Integration

For neural-symbolic integration to be effective in complex applications, we need to investigate how to represent, reason, and learn expressive logics in neural networks. We also need to find effective ways of expressing the knowledge encoded in a trained network in a comprehensible symbolic form.

There are two ways to move forward and benefit from neural-symbolic integration. The first is to take standard neural networks and try and find out which logics they can represent. The other is to take well established logics and concepts (e.g. recursion) and try and encode them in a neural network architecture. This needs to be carried out in a systematic way. Whenever we show that a particular logic can be represented by a particular neural network, we need to show that the network and the logic are in fact equivalent (a way to do this is to prove that the network computes the semantics of the logic). Similarly, if we develop a knowledge extraction algorithm, we need to make sure that it is correct in the sense that it produces rules that are encoded in the network, and that it is complete in the sense that it produces rules that increasingly approximate the exact behaviour of the network.

In the past twenty years, a number of models for neural-symbolic integration has been proposed. Broadly speaking, researchers have made contributions to three main areas. Neural-symbolic systems provide either: (*i*) a logical characterisation of a connectionist system; (*ii*) a connectionist implementation of a logic; or (*iii*) a hybrid system bringing together advantages from connectionist systems and symbolic artificial intelligence [15]. Key contributions to the area were given by Ron Sun [23], Lokendra Shastri [20], and Steffen Hölldobler [14] on the knowledge representation side, by Jude Shavlik [21] on learning with background knowledge, and by Sebastian Thrun on knowledge extraction [24], among others. The reader is referred to [6] for a general presentation of the subject of neural-symbolic integration, and to [4] for a more advanced collection of papers on the subject.

Neural-symbolic systems [6] contain six main phases: (1) *symbolic knowledge insertion*; (2) *inductive learning with examples*; (3) *massively parallel deduction*; (4) *theory fine-tuning*; (5) *symbolic knowledge extraction*; and (6) *feedback* (see Figure 1). In phase (1), symbolic knowledge is translated into the initial architecture of a neural network with the use of a *Translation Algorithm*. In phase (2), the neural network is trained with examples by a neural learning algorithm, which revises the theory given in phase (1) as *background knowledge*. In phase (3), the network can be used as a massively parallel system to compute the logical consequences of the theory encoded in it. In phase (4), information obtained from the computation carried out in phase (3) may be used to help fine-tuning the network to better represent the problem domain. This mechanism can be used, for example, to resolve inconsistencies between the background knowledge and the training examples. In phase (5), the result of training is explained by the extraction of revised symbolic knowledge. As with the insertion of rules, the *Extraction Algorithm* must be provably correct, so that each rule extracted is guaranteed to be encoded in the network. Finally, in phase (6), the knowledge

extracted may be analysed by an expert to decide if it should feed the system once again, closing the learning cycle. A typical application of Neural-Symbolic Systems is in safety-critical domains, e.g. power plant fault diagnosis, where the neural network can be used to detect a fault quickly, triggering safety procedures, while the knowledge extracted from it can be used to explain the reasons for the fault later on. If mistaken, this information can be used to fine tune the learning system.

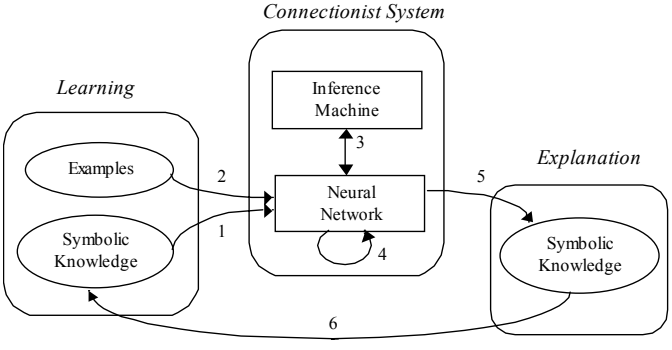


Fig. 1. Neural-Symbolic Learning Systems

In this paper, we focus on knowledge representation (phase (1) above). First, let us see how the *Translation Algorithm* works in the case of general logic programs[1]. Before we proceed, let us define the type of neural network used here. An artificial neural network is a directed graph. A unit in this graph is characterised, at time t, by its input vector $I_i(t)$, its input potential $U_i(t)$, its activation state $A_i(t)$, and its output $O_i(t)$. The units (neurons) of the network are interconnected via a set of directed and weighted connections. If there is a connection from unit i to unit j then $W_{ji} \in \Re$ denotes the *weight* associated with such a connection. The input potential of neuron i ($U_i(t)$) is obtained by applying the *propagation rule* of neuron i (g_i) such that $U_i(t) = g_i(I_i(t), W_i)$, where $I_i(t)$ is the input vector $(x_1(t), x_2(t), ..., x_n(t))$ to neuron i at time t, and W_i denotes the weight vector $(W_{i1}, W_{i2}, ..., W_{in})$ to neuron i. In addition, θ_i (an extra weight with input always fixed at 1) is known as the *threshold* of neuron i.

The *activation state* of neuron i ($A_i(t)$) is a bounded real or integer number given by its *activation rule* (h_i). In general, h_i does not depend on the previous activation state of the neuron, and the propagation rule g_i is a weighted sum such that $A_i(t) = h_i(\sum_j((W_{ij} \cdot x_j(t)) - \theta_i))$. Finally, in general, the output is given by the identity function, and thus $O_i(t) = A_i(t)$.

The units of a neural network can be organised in layers. A *n-layer feed-forward network* N is an acyclic graph. N consists of a sequence of layers and

[1] A general clause is a rule of the form $L_1, ..., L_k \rightarrow A$, where A is an atom and L_i $(1 \leq i \leq k)$ is a literal. A general logic program is a finite set of general clauses.

connections between successive layers, containing one input layer, $n-2$ hidden layers and one output layer, where $n \geq 2$. When $n = 3$, we say that N is a *single hidden layer network*. When each unit occurring in the *i-th* layer is connected to each unit occurring in the $i+1$-*st* layer, we say that N is a *fully-connected network*.

Now, let $\mathcal{P}$ be a general logic program, and let $\mathcal{N}$ be a single hidden layer feedforward neural network. Each clause (r_l) of $\mathcal{P}$ can be mapped from the input layer to the output layer of $\mathcal{N}$ through one neuron (N_l) in the single hidden layer of $\mathcal{N}$. Intuitively, the *Translation Algorithm* from $\mathcal{P}$ to $\mathcal{N}$ implements the following conditions: (**C1**) The input potential of a hidden neuron (N_l) can only exceed N_l's threshold (θ_l), activating N_l, when all the positive antecedents of r_l are assigned the truth-value *true* while all the negative antecedents of r_l are assigned $false$; and (**C2**) The input potential of an output neuron (A) can only exceed A's threshold (θ_A), activating A, when at least one hidden neuron N_l that is connected to A is activated.

Example: Consider the logic program $\mathcal{P} = \{B \wedge C \wedge \neg D \rightarrow A; E \wedge F \rightarrow A; B\}$. The *Translation Algorithm* derives the network $\mathcal{N}$ of Figure 2, setting weights (W) and thresholds (θ) in such a way that conditions (**C1**) and (**C2**) above are satisfied. Note that, if $\mathcal{N}$ ought to be fully-connected, any other link (not shown in Figure 2) should receive weight zero initially.

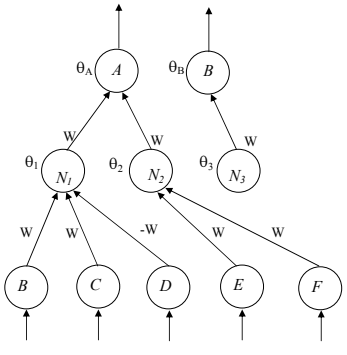


Fig. 2. Neural Network for Logic Programming

Note that, in the above example, each input and output neuron of $\mathcal{N}$ is associated with an atom of $\mathcal{P}$. As a result, each input and output vector of $\mathcal{N}$ can be associated with an interpretation for $\mathcal{P}$. Note also that each hidden neuron N_l corresponds to a rule r_l of $\mathcal{P}$. In order to compute the stable models of $\mathcal{P}$, output neuron B should feed input neuron B such that $\mathcal{N}$ is used to iterate $T_{\mathcal{P}}$, the fixed-point operator of $\mathcal{P}$. $\mathcal{N}$ will eventually converge to a stable state which is identical to a stable model of $\mathcal{P}$ [6].

Details about the translation and extraction algorithms, their proofs of correctness, and extensions to other types of logic program can be found in [6], together with algorithms to deal with inconsistencies and experimental results

in the areas of DNA sequence analysis, power systems fault diagnosis, and the evolution of requirements in software engineering.

3 Connectionist Modal Logic

Let us now consider the case of modal logic programs, which extend logic programs with the *necessity* ($\Box$) and *possibility* ($\Diamond$) modalities, according to an accessibility relation $\mathcal{R}(\omega_i, \omega_j)$ on possible worlds ω_i and ω_j. I will give an example of how neural networks can represent such modalities. The basic idea is simple. Instead of having a single network, if we now allow a number of networks (like the one in Figure 2) to occur in an ensemble, and we label the networks as w_1, w_2, etc, we can talk about having x in w_1 and having x in w_2. In this way, we can see w_1 as a possible world and w_2 as another, and this allows us to represent modal logic programs[2]. This is of interest in connection with McCarthy's conjecture on the propositional fixation of neural networks because there is a well established translation between propositional modal logic and the two-variable fragment of first order logic[3] [26], which indicates that neural-symbolic systems may go beyond propositional logic, thus contradicting McCarthy's conjecture.

Example: Let $\mathcal{P} = \{\omega_1 : r \rightarrow \Box q;\ \omega_1 : \Diamond s \rightarrow r;\ \omega_2 : s;\ \omega_3 : q \rightarrow \Diamond p;\ \mathcal{R}(\omega_1;\omega_2),\ \mathcal{R}(\omega_1,\omega_3)\}$. The network ensemble $\mathcal{N}$ in Figure 3 is equivalent to $\mathcal{P}$. Take network $\mathcal{N}_1$ (representing ω_1). To implement the semantics of $\Diamond$, output neurons of the form $\Diamond\alpha$ should be connected to output neurons α in an arbitrary network $\mathcal{N}_i$ (representing ω_i) to which $\mathcal{N}_1$ is related. For example, taking $i = 2$, $\Diamond s$ in $\mathcal{N}_1$ is connected to s in $\mathcal{N}_2$. To implement the semantics of $\Box$, output neurons $\Box\alpha$ should be connected to output neurons α in every network $\mathcal{N}_i$ to which $\mathcal{N}_1$ is related. For example, $\Box q$ in $\mathcal{N}_1$ is connected to q in both $\mathcal{N}_2$ and $\mathcal{N}_3$. Dually, taking $\mathcal{N}_2$, output neurons α need to be connected to output neurons $\Diamond\alpha$ and $\Box\alpha$ in every world $\mathcal{N}_j$ related to $\mathcal{N}_2$. For example, s in $\mathcal{N}_2$ is connected to $\Diamond s$ in $\mathcal{N}_1$ via the hidden neuron denoted by $\vee$ in Figure 3, while q in $\mathcal{N}_2$ is connected to $\Box q$ in $\mathcal{N}_1$ via the hidden neuron denoted by $\wedge$. Similarly, q in $\mathcal{N}_3$ is connected to $\Box q$ in $\mathcal{N}_1$ via $\wedge$. The translation terminates when all output neurons have been considered. The translation algorithm defines the weights and thresholds of the network in such a way that it can be shown to compute a fixed-point semantics of the modal logic program associated to it (for any extended modal

[2] An *extended modal program* is a finite set of clauses C of the form $\omega_i : ML_1, ..., ML_n \rightarrow MA$, where ω_i is a label representing a world in which the associated clause holds, and $M \in \{\Box, \Diamond\}$, together with a finite set of relations $\mathcal{R}(\omega_i, \omega_j)$ between worlds ω_i and ω_j in C.

[3] In [26], Vardi states that "(propositional) modal logic, in spite of its apparent propositional syntax, is essentially a first-order logic, since the necessity and possibility modalities quantify over the set of possible worlds... the states in a Kripke structure correspond to domain elements in a relational structure, and modalities are nothing but a limited form of quantifiers". In the same paper, Vardi then proves that propositional modal logics correspond to fragments of first-order logic.

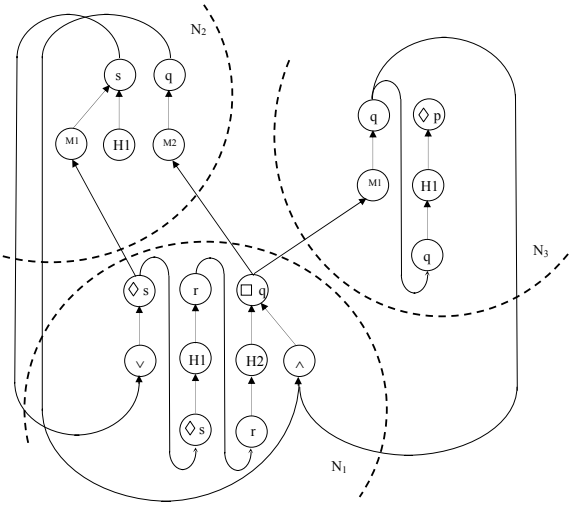


Fig. 3. Connectionist Modal Logic

program $\mathcal{P}$ there exists an ensemble of feedforward neural networks $\mathcal{N}$ with a single hidden layer and semi-linear neurons, such that $\mathcal{N}$ computes the modal fixed-point operator $MT_{\mathcal{P}}$ of $\mathcal{P}$). The proof and details about the algorithm can be found in [9]. Finally, as we link the neurons in the output layer to the corresponding neurons in the input layer of each network $\mathcal{N}_i$, the ensemble can be used to compute the modal program in parallel. In this example, we connect output neurons $\Diamond s$ and r to input neurons $\Diamond s$ and r, respectively, in $\mathcal{N}_1$, and output neuron q to input neuron q in $\mathcal{N}_3$. The ensemble converges to a stable state containing $\{\Diamond s, r, \Box q\}$ in ω_1, $\{s, q\}$ in ω_2, and $\{q, \Diamond s\}$ in ω_3.

4 Fibring Neural Networks

In Connectionist Modal Logic (CML), one needs to create copies of certain concepts. As a result, CML cannot deal with infinite domains, since this would require infinitely many copies. An alternative is to map the instances of a variable onto the reals, and then use real numbers as inputs to a neural network as a way of representing variables. This has been done in [15], in which a theorem shows that the fixed-point semantics of first order logic programs can be approximated arbitrarily well by neural networks very similar to the one depicted in Figure 2. However, the question of which neural network approximates a given first order program remained, since no translation algorithm has been introduced in [15]. Recently, we have followed the idea of representing variables as real numbers, and proposed a translation algorithm from first order acyclic programs to neural network ensembles [2]. The algorithm makes use of *fibring* of neural networks [7], which we discuss in the sequel. Briefly, the idea is to use a neural network to iterate a global counter n. For each clause C_i in the logic

program, this counter is combined (fibred) with another neural network, which determines whether C_i outputs an atom of level n for a given interpretation I. This allows us to translate programs having an infinite number of ground instances into a finite neural network structure (e.g. $\neg even(x) \rightarrow even(s(x))$ for $x \in \mathbb{N}, s(x) = x + 1$), and to prove that indeed the network approximates the fixed-point semantics of the program. The translation is made possible because fibring allows one to implement a key feature of symbolic computation in neural networks, namely, recursion.

The idea of fibring neural networks is simple. Fibred networks may be composed not only of interconnected neurons but also of other networks, forming a recursive architecture. A fibring function then defines how this recursive architecture must behave by defining how the networks in the ensemble relate to each other. Typically, the fibring function will allow the activation of neurons in one network (A) to influence the change of the weights in another network (B) (e.g. by allowing the activation state of a neuron in A to be multiplied by the weights of neurons in B). Intuitively, this may be seen as training network B at the same time that one runs network A. Interestingly, albeit being a combination of simple and standard neural networks, fibred networks can approximate any polynomial function in an unbounded domain, thus being more expressive than standard feedforward networks (which are universal approximators of functions in compact, i.e. closed and bounded, domains only) [7]. For example, fibred networks compute $f(x) = x^2$ exactly for $x \in \mathbb{R}$.

Figure 4 exemplifies how a network (B) can be fibred into a network (A). Of course, the idea of fibring is not only to organise networks as a number of subnetworks (A, B, etc). In Figure 4, for example, the output neuron of A is expected to be a neural network (B) in its own right. The input, weights, and output of B may depend on the activation state of A's output neuron, according to the fibring function φ. One such function may be simply to *multiply* the weights of B by the activation state of A's output neuron. Fibred networks can be trained from examples in the same way that standard networks are (for

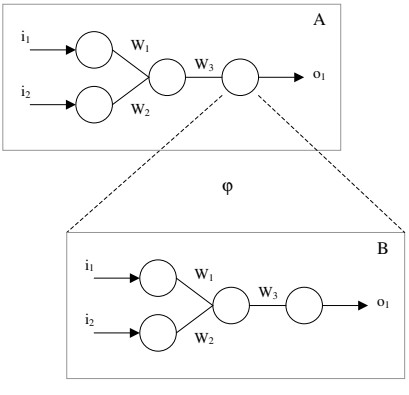


Fig. 4. Fibring Neural Networks

example, with the use of *backpropagation* [18]). Networks A and B above, e.g., could have been trained separately before having been fibred. Notice also that, in addition to using different fibring functions, networks can be fibred in a number of different ways as far as their architectures are concerned. Network B above, e.g., could have been fibred into a hidden neuron of network A.

I believe that fibring can contribute to the solution of the problem of elaboration tolerance by offering a principled and modular way of combining networks. Network A could have been trained, e.g., with a robot's visual system, while network B would have been trained with its planning system, and fibring would serve to perform the composition of the two systems (along the lines of Gabbay's methodology for fibring logical systems [11]). Of course, a lot of work still remains to be done in this area, particularly in what regards the question of how one should go about fibring networks in real applications.

5 Concluding Remarks

I see CML as an example of how neural networks can contribute to logic, and I see fibring as an example of how logic can bring insight into neural networks. CML offers a parallel model of computation to modal logic that, at the same time, can be integrated with an efficient learning system. Fibring is a clear example of how concepts from symbolic computation may help in the development of new neural network models (this does not necessarily conflicts with the concept of biological plausibility, e.g. fibring functions can be understood as a model of *presynaptic weights*, which play an important role in biological neural networks).

Together with its algorithms for learning from examples and background knowledge [6] and for knowledge extraction from trained neural networks [5] (which I have neglected in this paper), I believe that neural-symbolic integration finally starts to address all the challenges put forward by McCarthy. On the other hand, there are new challenges now, which arise directly from our goal of integrating reasoning and learning in a principled way, as put forward by Valiant [25].

In my opinion, the key challenges ahead are: how to get a *constructive translation* of variables into simple neural networks, and how to have a *sound and complete extraction* method that is also efficient for large-scale networks. Let me try and explain what I mean by a constructive translation. In the propositional case, when we look at a neural network, we can see the literals and their relationship with other literals explicitly represented as neurons and their connections with other neurons in the network. In the same way, we would like to be able to look at a first order neural network and see the variables and their relationship with other variables explicitly represented in the network. Although fibring allows us to translate first order programs into neural networks, the current translation algorithm does not produce one such constructive view of the network. As a result, we still do not know how to learn first order programs using neural networks, and I believe that a constructive translation would help shed light into this learning problem. Due to the success of the propositional case, I

am convinced that such a representation would allow for effective learning if it kept the networks simple. This is still a big challenge. Of course, we will need to be much more precise as we develop this work, and we will need to keep an eye on the recent developments in the area of logic and learning [16].

References

1. R. Andrews, J. Diederich, and A. B. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based Systems*, 8(6):373–389, 1995.
2. S. Bader, A. S. d'Avila Garcez, and P. Hitzler. Computing first order logic programs by fibring artificial neural networks. In Proceedings of International FLAIRS Conference, Florida, USA, AAAI Press, 2005.
3. G. Bologna. Is it worth generating rules from neural network ensembles? *Journal of Applied Logic, Special issue on Neural-Symbolic Integration*, A. S. d'Avila Garcez, D. Gabbay, S. Hölldobler, J. G. Taylor (eds.), 2(3):325–348, 2004.
4. I. Cloete and J. M. Zurada, editors. *Knowledge-Based Neurocomputing*. The MIT Press, 2000.
5. A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125:155–207, 2001.
6. A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Perspectives in Neural Computing. Springer-Verlag, 2002.
7. A. S. d'Avila Garcez and D. M. Gabbay. Fibring neural networks. In *Proceedings of 19th National Conference on Artificial Intelligence AAAI'04*, pages 342–347, San Jose, California, USA, AAAI Press. July 2004.
8. A. S. d'Avila Garcez and L. C. Lamb. Reasoning about time and knowledge in neural-symbolic learning systems. In S. Thrun, L. Saul, and B. Schoelkopf, editors, *Advances in Neural Information Processing Systems 16*, Proceedings of the NIPS 2003 Conference, pages 921–928, Vancouver, Canada, MIT Press. December 2004.
9. A. S. d'Avila Garcez, L. C. Lamb, K. Broda, and D. M. Gabbay. Applying connectionist modal logics to distributed knowledge representation problems. *International Journal of Artificial Intelligence Tools*, 13(1):115–139, 2004.
10. D. M. Gabbay. *Elementary Logics: a Procedural Perspective*. Prentice Hall, 1998.
11. D. M. Gabbay. *Fibring Logics*. Oxford Univesity Press, 1999.
12. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.
13. P. Hitzler, S. Hlldobler, and A. Seda. Logic programs and connectionist networks. *Journal of Applied Logic, Special issue on Neural-Symbolic Integration*, A. S. d'Avila Garcez, D. Gabbay, S. Hölldobler, J. G. Taylor (eds.), 2(3):245–272, 2004.
14. S. Hölldobler. Automated inferencing and connectionist models. Postdoctoral Thesis, Intellektik, Informatik, TH Darmstadt, 1993.
15. S. Hölldobler, Y. Kalinke, and H. P. Storr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, F. Kurfess (ed.), 11(1):45–58, 1999.
16. J. W. Lloyd. *Logic for Learning: Learning Comprehensible Theories from Structured Data*. Springer-Verlag, 2003.

17. J. McCarthy. Epistemological challenges for connectionism. *Behavior and Brain Sciences*, 11(1):44, 1988.
18. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, 1986.
19. R. Setiono. Extracting rules from neural networks by pruning and hidden-unit splitting. *Neural Computation*, 9:205–225, 1997.
20. L. Shastri. Advances in SHRUTI: a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, F. Kurfess (ed.), 11:79–108, 1999.
21. J. W. Shavlik. An overview of research at Wisconsin on knowledge-based neural networks. In *Proceedings of the International Conference on Neural Networks ICNN96*, pages 65–69, Washington, DC, 1996.
22. P. Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74, 1988.
23. R. Sun. Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, 75(2):241–296, 1995.
24. S. B. Thrun. Extracting provably correct rules from artificial neural networks. Technical report, Institut fur Informatik, Universitat Bonn, 1994.
25. L. G. Valiant. Three problems in computer science. *Journal of the ACM*, 50(1):96–99, 2003.
26. M. Y. Vardi. Why is modal logic so robustly decidable. In N. Immerman and P. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *Discrete Mathematics and Theoretical Computer Science*, pages 149–184. 1997.

An Algebraic View on Exact Learning from Queries

Ricard Gavaldà*

Department of Software (LSI),
Universitat Politècnica de Catalunya,
Barcelona, Spain
gavalda@lsi.upc.edu

Computational Learning Theory has two goals: one, proposing rigorous models of learning tasks that can be carried out by computers. Two, designing algorithms that provably learn whole classes of concepts in these models within some efficiency criterion.

Angluin's *exact learning from queries* [1, 2] is one of the best studied such models. It assumes that the concept to learn is a function taken from some fixed class. A learning algorithm is allowed to ask certain types of queries about this function, and terminate having identified the function exactly. The learning task is completely specified once a function class and the allowed types of queries are specified.

We will survey some recent work on the learnability of classes of functions defined by programs over monoids or semigroups, in the mentioned model of exact learning from queries. In [3, 4], *expressions* over a monoid M were considered; expressions on n variables over M compute functions from M^n to M. More recently, together with A. Chattopadhyay, K. Arnsfelt Hansen, and D. Thérien, we have started the study of boolean functions computed by the standard notion of *boolean program* over a semigroup [5, 6]; these programs, over n variables, compute boolean functions $\{0,1\}^n \mapsto \{0,1\}$.

As was to be expected, there is a strong relation between algebraic properties of the semigroup and the complexity of the associated learning problems. In many contexts, it is possible to design learning algorithms that learn whole (and natural) varieties of semigroups. Often, one can show that in fact the limit of learnability (e.g., using a fixed set of query types) coincides exactly with such a natural variety of semigroups.

More in general, we regard this work as trying to identify algebraic properties that lead to, or prevent, learnability. We believe that this approach may provide useful insights on the deep reasons why learning tasks are easy or hard, by explaining why and when some of the algorithmic techniques designed in Computational Learning Theory will work.

* Partially supported by MCYT TIC 2002-04019 (MOISES), TIC2001-1577-C03-02 (LOGFAC), and by CIRIT 1997SGR-00366.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 150–151, 2005.

Of particular interest is the fact that many of the classes of boolean functions that appear naturally in this study had been independently studied in Computational Learning Theory, without any reference to semigroups (see [7] for some more examples).

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75** (1987) 87–106
2. Angluin, D.: Queries and concept learning. Machine Learning **2** (1988) 319–342
3. Gavaldà, R., Thérien, D.: Learning expressions over monoids. In: Proc. 18th Intl. Symposium on Theoretical Aspects of Computer Science (STACS'01). Volume 2010 of Lecture Notes in Computer Science., Springer (2001) 283–293
4. Gavaldà, R., Thérien, D., Tesson, P.: Learning expressions and programs over monoids. Technical Report R01–38, Departament LSI, Universitat Politecnica de Catalunya (2001) Submitted to journal.
5. Barrington, D.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. Journal of Computer and System Sciences **38** (1989) 150–164
6. Barrington, D.M., Straubing, H., Thérien, D.: Non-uniform automata over groups. Information and Computation **89** (1990) 109–132
7. Gavaldà, R., Thérien, D.: Algebraic characterization of small classes of boolean functions. In: Proc. 20th Intl. Symposium on Theoretical Aspects of Computer Science (STACS'03). Volume 2607 of Lecture Notes in Computer Science., Springer (2003) 331–342

The Church-Turing Thesis: Breaking the Myth*

Dina Goldin[1] and Peter Wegner[2]

[1] Computer Science & Engineering Department,
University of Connecticut, 371 Fairfield Rd., Unit 2155,
Storrs, CT 06269, USA
dqg@cse.uconn.edu
[2] Computer Science Department,
Brown University, Providence, RI 02912, USA
pw@cs.brown.edu

Abstract. According to the interactive view of computation, communication happens *during* the computation, not before or after it. This approach, distinct from concurrency theory and the theory of computation, represents a paradigm shift that changes our understanding of what is computation and how it is modeled. Interaction machines extend Turing machines with interaction to capture the behavior of concurrent systems, promising to bridge these two fields. This promise is hindered by the widespread belief, incorrectly known as the Church-Turing thesis, that no model of computation more expressive than Turing machines can exist. Yet Turing's original thesis only refers to the *computation of functions* and explicitly excludes other computational paradigms such as interaction. In this paper, we identify and analyze the historical reasons for this widespread belief. Only by accepting that it is false can we begin to properly investigate formal models of interaction machines. We conclude the paper by presenting one such model, *Persistent Turing Machines* (PTMs). PTMs capture *sequential interaction*, which is a limited form of concurrency; they allow us to formulate the *Sequential Interaction Thesis*, going beyond the expressiveness of Turing machines and of the Church-Turing thesis.

1 Introduction

The fields of the theory of computation and concurrency theory have historically had different concerns. The theory of computation views computation as a *closed-box* transformation of inputs to outputs, completely captured by Turing machines (TMs). By contrast, concurrency theory focuses on the *communication* aspect of computing systems, which is not captured by Turing machines – referring both to the communication between computing components in a system, and the communication between the computing system and its environment. As a

* We sincerely thank the anonymous reviewers for their comments, which were very helpful in revising this paper.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 152–168, 2005.

result of this division of labor, there has been little in common between these fields.

According to the interactive view of computation, communication (input/output) happens *during* the computation, not before or after it. This approach, distinct from either concurrency theory or the theory of computation, represents a paradigm shift that changes our understanding of what computation is and how it is modeled [21]. Interaction machines extend Turing machines with interaction to capture the behavior of concurrent systems, promising to bridge these two fields.

The idea that Turing machines do not capture all computation, and that interaction machines are more expressive, seems to fly in the face of accepted dogma, hindering its acceptance within the theory community. In particular, the Church-Turing thesis is commonly interpreted to imply that Turing machines model all computation. It is a myth that the original thesis is equivalent to this interpretation of it. In fact, the Church-Turing thesis only refers to the computation of *functions*, and specifically excludes interactive computation. The original contribution of this paper is to identify and analyze the historical reasons for this myth.

It is time to recognize that today's computing applications, such as web services, intelligent agents, operating systems, and graphical user interfaces, cannot be modeled by Turing machines; alternative models are needed. Only by facing the fact that this reinterpretation is a myth can we begin to properly investigate these alternative models. We present one such model, *Persistent Turing Machines* (PTMs), originally formalized in [8]. PTMs capture *sequential interaction*, which is a limited form of concurrency; they allow us to formulate the *Sequential Interaction Thesis*, going beyond the expressiveness of Turing machines and of the Church-Turing thesis.

2 The Turing Thesis Myth

Turing's famous 1936 paper [18] developed the *Turing Machine* (TM) model and showed that TMs have the expressiveness of algorithms (now known as the Church-Turing Thesis).

Church-Turing Thesis: Whenever there is an effective method (algorithm) for obtaining the values of a mathematical function, the function can be computed by a TM.

TMs are identified with the notion of *effectiveness*; *computability* is the current term for the same notion. Specificially, they capture computable functions as effective transformations from finite strings (natural numbers) to finite strings.

The Church-Turing thesis has since been reinterpreted to imply that Turing Machines model *all* computation, rather than just functions. This claim, which we call the *Strong Church-Turing Thesis*, is part of the mainstream theory of

computation. In particular, it can be found in today's popular undergraduate theory textbooks such as [17]:

Strong Church-Turing Thesis: A TM can do (compute) anything that a computer can do.

It is a myth that the original Church-Turing thesis is equivalent to this interpretation of it; Turing himself would have denied it. In the same famous paper, he also introduced interactive *choice machines* as another model of computation distinct from TMs and not reducible to it. Choice machines extend TMs to interaction by allowing a human operator to make choices during the computation.

In fact, the Strong Church-Turing Thesis is incorrect – the function-based behavior of algorithms does not capture all forms of computation. For example, as explained in [21], interactive protocols are not algorithmic. Yet the myth of the correctness of the Strong Turing Thesis is dogmatically accepted by most computer scientists. As Denning recently wrote [5], "we are captured by a historic tradition that sees programs as mathematical functions".

The reasons for the widespread acceptance of what we call the "Turing Thesis myth" can be traced to the establishment of computer science as a separate discipline in the 1960's. To understand these reasons better, we can identify three distinct claims that make up the myth, and examine them individually:

Claim 1. All computable problems are function-based.
Reason: Adoption of mathematical principles for the fundamental notions of computation, identifying computability with the computation of functions, as well as with TMs.

Claim 2. All computable problems can be described by an algorithm.
Reason: Adoption of algorithms as the central and complete concept of computer science.

Claim 3. Algorithms are what computers do.
Reason: Broadening the concept of an algorithm to make it more practical.

We will investigate these claims and analyze why they have emerged in the computer science community.

3 What Is Computation?

The first reason for the Turing Thesis myth is related to the basic understanding of the notion of computation, as adopted by the theory of computation community; we refer to it as the "mathematical worldview".

3.1 The Mathematical Worldview

The theory of computation predates the establishment of computer science as a discipline, having been a part of mathematics before the 1960's. Its founders

include such notable mathematicians as Godel, Kleene, Church, and Turing.[1] Mathematicians naturally equated the notion of computability with the computation of functions. Martin Davis's 1958 textbook [4], popular among computer scientists, reflected the *mathematical worldview* that all computation is function-based, and therefore captured by TMs. It begins as follows:

> "This book is an introduction to the theory of computability and non-computability, usually referred to as the theory of recursive functions... the notion of TM has been made central in the development."

In particular, this view assumes that all computation is *closed*. There is no input or output taking place during the computation; any information needed during the computation is provided at the outset as part of the input. These assumptions are embodied by the semantics of TMs.

Mathematical Worldview: All computable problems are function-based.

The mathematical worldview was enthusiastically adopted by early leaders of the computer science community, including Von Neumann, Knuth, Karp, Rabin, and Scott. Mathematics has been used as a foundation of physics and other scientific disciplines, and it was believed that mathematics could be used as a basis for computer science. Davis's book proved very influential, cementing the acceptance of the mathematical worldview among computer scientists of the 1950's and 1960's. The mathematical worldview is the first of the three claims that constitute the Turing Thesis myth.

TMs, which transform input strings to output strings, have served from the onset as a formal model for function-based computation:

Turing Thesis Corollary: A problem is *solvable* if there exists a Turing machine for computing it.

The legitimacy of this corollary is based on two premises. The first one is the Turing Thesis, which equates function-based computation with TMs. The second one, usually left unstated, is the mathematical worldview – the assumption that all computable problems are function-based.

The perceived validity of this corollary was greatly strengthened by the many early attempts to find models of computation that are more expressive than TMs, for example extending the number of tapes or reading heads on the machine. All these attempts failed, because they continued to adhere to the mathematical worldview, and never considered problems that are not function-based. We return to this issue in Section 6.1.

[1] While Turing's training and original contributions were mathematical, we believe that his later work classifies him as a computer scientist rather than a mathematician – perhaps the first one.

3.2 The Interactive Paradigm

The mathematical worldview can be contrasted with the *engineering worldview*, where computation is viewed as an ongoing transformation of inputs to outputs – e.g., control systems, or operating systems. The question "*what do operating systems compute?*" has been a conundrum for the theoretical community, since they never terminate, and therefore never formally produce an output. Yet it is clear that they *do* compute, and that their computation is both useful and important to capture formally.

While the Church-Turing Thesis remains true, the mathematical worldview no longer reflects the nature of computational problems. An example of such a problem is *driving home from work* [21]:

Driving Home from Work: create a car that is capable of driving us home from work, where the locations of both work and home are provided as input parameters.

Assuming that the driving is to take place in a real-world environment, this problem is not computable within a function-based computational paradigm.

Consider the input to such a function. It would have to be detailed enough so the car could predict the direction and strength of the wind at each point in the drive, so as to compensate for it. It should also enable the car to anticipate the location of all pedestrians, so as to avoid running over them. As we discuss in [6], this is impossible – there is no such computable function. However, the problem *is* computable by a control mechanism, as in a robotic car, that continuously receives video input of the road and actuates the wheel and brakes accordingly.

The computation performed by automatic cars and operating systems is *interactive*, where input and output happen *during* the computation, not before or after it. This approach, distinct from either concurrency theory or the theory of computation, represents a paradigm change to our understanding of what is computation, and how it should be modeled. This conceptualization of computation allows, for example, the *entanglement* of inputs and outputs; later inputs to the computation may depend on earlier outputs. Such entanglement is impossible in the mathematical worldview, where all inputs precede computation, and all outputs follow it.

The driving example represents an empirical proof of the claim that interactive computation is more expressive than function-based computation, i.e. it can solve a greater range of problems. However, to accept this claim, one has to broaden one's notion of a problem beyond what is prescribed by the mathematical worldview. Driving home from work, queuing jobs within an operating system, or controlling factory equipment, are all legitimate problems on par with finding common factors or choosing the next move on a given chess board.

4 Algorithms and Computability

The notion of an *algorithm* is a mathematical concept much older than Turing machines; perhaps the oldest example is *Euclid's algorithm* for finding common divisors. This concept has been enthusiastically adopted by the computer science community, who have since broadened its meaning. This adoption and the subsequent broadening are sources of the second and third reasons for the Turing Thesis myth.

4.1 The Original Role of Algorithms

Algorithms are "recipes" for carrying out function-based computation, that can be followed mechanically.

Role of Algorithm: Given some finite input x, an algorithm describes the steps for effectively transforming it to an output y, where y is $f(x)$ for some recursive function f.

Like mathematical formulae, algorithms tell us how to compute a value; unlike them, algorithms may involve what we now call *loops*.

Knuth's famous and influential textbook, *The Art of Computer Programming, Vol. 1* [10] in the late 1960's popularized the centrality of algorithms in computer science. In his discussion of an algorithm, Knuth was consistent with the mathematical function-based foundations of the theory of computation. He explicitly specified that algorithms are closed; no new input is accepted once the computation has begun:

> "An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins."

Knuth distinguished algorithms from arbitrary computation that may involve I/O. One example of a problem that is not algorithmic is the following instruction from a recipe [10]: "toss lightly until the mixture is crumbly." This problem is not algorithmic because it is impossible for a computer to know how long to mix; this may depend on conditions such as humidity that cannot be predicted with certainty ahead of time. In the function-based mathematical worldview, all inputs must be specified at the start of the computation, preventing the kind of feedback that would be necessary to determine when it's time to stop mixing. The problem of driving home from work also illustrates the sort of problems that Knuth meant to exclude.

The notion of an algorithm is inherently informal, since an algorithmic description is not restricted to any single language or formalism. The first high-level programming language developed expressly to specify algorithms was ALGOL (ALGOrithmic Language). Introduced in the late 50's and refined through the 1960's, it was the standard for the publication of algorithms. True to the function-based conceptualization of algorithms, ALGOL provided no constructs for input and output, considering these operations outside the concern of algorithms. Not surprisingly, this absence hindered the adoption of ALGOL by the industry for commercial applications.

Knuth's careful discussion of algorithmic computation remains definitive to this day; in particular, it serves as the basis of the authors' understanding of this term. His discussion of algorithms ensures their function-based behavior and guarantees their equivalence with TMs [10]:

> "There are many other essentially equivalent ways to formulate the concept of an effective computational method (for example, using TMs)."

4.2 Algorithms Made Central

The 1960's saw a proliferation of undergraduate computer science (CS) programs; this increase was accompanied by intense activity towards establishing the legitimacy of this new discipline in the eyes of the academic community. The Association for Computing Machinery (ACM) played a central role in this activity. In 1965, it enunciated the justification and description of CS as a discipline [1], which served as a basis of its 1968 recommendations for undergraduate CS programs [2]; one of the authors (PW) was among the primary writers of the 1968 report.

ACM's description of CS [1] identified effective transformation of information as a central concern:

> "Computer science is concerned with information in much the same sense that physics is concerned with energy... The computer scientist is interested in discovering the pragmatic means by which information can be transformed."

By viewing algorithms as transformations of input to output, ACM adapted an algorithmic approach to computation; this is made explicit in the next sentence of the report:

> "This interest leads to inquiry into effective ways to represent information, effective algorithms to transform information, effective languages with which to express algorithms... and effective ways to accomplish these at reasonable cost."

Having a central algorithmic concern, analogous to the concern with energy in physics, helped to establish CS as a legitimate academic discipline on a par with physics.

Algorithms, modeled by TMs, have remained central to computer science to this day. The coexistence of the informal (algorithm-based) and the formal (TM-based) approaches to defining solvable problems can be found in all modern textbooks on algorithms or computability. This has proved tremendously convenient for computer scientists, by allowing us to describe function-based computation informally using "pseudocode", with the knowledge that an equivalent Turing machine can be constructed.

However, neither mathematics nor the ACM provided an explicit agreed-upon definition of an algorithm. As we will see, the inconsistencies in the various definitions of this term greatly contributed to the Turing Thesis myth.

4.3 Algorithms Redefined

The 1960's decision by theorists and educators to place algorithms at the center of CS was clearly reflected in early undergraduate textbooks. However, various textbooks chose to define this term differently. While some textbooks such as [10] were careful to explicitly restrict algorithms to those that compute functions, and are therefore TM-equivalent, most left the restriction unstated.

An early example is [9], one of the first textbooks on the theory of computation (whose later editions are being used to this day). Their discussion of algorithms does not *explicitly* preclude non-functional computation, such as driving home from work:

> "A procedure is a finite sequence of instructions that can be mechanically carried out, such as a computer program... A procedure which always terminates is called an algorithm."

However, the prohibition against obtaining inputs dynamically during the computation is *implicitly* present in [9]. After all, ALGOL, the language then used for writing algorithmic programs, did not offer any constructs for input and output. Their examples of various problems also make it clear that only function-based computation was considered.

Yet other early textbooks, such as [12], explicitly broadened the notion of algorithms to include problems beyond those that can be solved by TMs. On the surface, their definition of an algorithm is no different from [9]:

> "An algorithm is a recipe, a set of instructions or the specifications of a process for doing something. That something is usually solving a problem of some sort."

However, their examples of computable problems are no longer function based, admitting just the kind of computation that Knuth had rejected. Two such examples, that can supposedly be solved by an algorithm, are making potato vodka and filling a ditch with sand; driving home from work would fit right in, too.

The subject of [12] was programming methodology rather than the theory of computation, and the mathematical principles that underpin our models of computation were cast aside for the sake of practicality. This approach, reflecting the centrality of algorithms without being restricted to the computation of functions, is typical of non-theory textbooks.

[12] made no claims of TM-equivalence for their "algorithms". However, the students were not made aware that their notion of algorithms is different from Knuth's, and that the set of problems considered computable had thereby been enlarged. By pairing Rice's broader conceptualization of algorithms (and hence of computable problems) with theories claiming that every computable problem can be computed by TMs, the algorithm-focused CS curriculum left students

with the impression that this broader set of problems could also be solved by TMs, giving rise to the Turing Thesis myth.[2]

4.4 Algorithms Today

A recent ACM SIGACT Newsletter acknowledges that of all undergraduate CS subjects, theoretical computer science has changed the least over the decades [16]. While the practical computer scientists have long since followed the lead of [12] and broadened the concept of algorithms beyond the computation of functions, theoretical computer science has retained the mathematical worldview that frames computation as function-based, and delimits our notion of a computational problem accordingly. This is true at least at the undergraduate level, despite advanced complexity theoretic work that ventures outside this worldview, such as on-line and distributed algorithms, Arthur-Merlin games, and interactive proofs.

The result is a dichotomy, where the computer science community thinks of algorithms as synonymous with the general notion of computation ("what computers do") yet at the same time as being equivalent to Turing machines. This dichotomy can be found in today's popular textbooks such as [17]. Their discussion of algorithms is very broad, but the equivalence with TMs is taken for granted:

> "an algorithm is a collection of simple instructions for carrying out some task. Commonplace in everyday life, algorithms sometimes are called procedures or recipes... The TM merely serves as a precise model for the definition of algorithm."

While their traditional selection of computational problems is all function-based, this description of algorithms certainly leaves an impression that tasks such as operating system processes are considered algorithmic. After all, these are tasks that computers carry out all the time.

The result of this dichotomy is the Strong Turing Thesis. It is commonplace in the computing literature, including [17]:

> "A TM can do anything that a computer can do."

5 What Is a Turing Machine?

Other claims have been used in support of the Strong Turing Thesis. We do not believe they played a major role in giving rise to the myth, yet they continue to serve as "corroboration" of its correctness. In this section, we discuss two such claims:

[2] In private conversation with one of the authors (DG) in the fall of 1999, Knuth expressed some misgivings about his definition of an algorithm, and shared plans to broaden it if that text were ever rewritten. It is not clear what his plans were regarding the claim of equivalence between algorithms and TMs.

Claim 4. TMs serve as a general model for computers.
Reason: Misunderstanding the definition of a TM.
Claim 5. TMs can simulate any computer.
Reason: Misattributing general purpose computing to TMs.

These claims, related to the limitations of early computers, are discussed in this section.

5.1 Syntax and Semantics

According to the set-theoretic definition, a TM consists of a finite set of states, a read/write head, a tape, and a control mechanism for transitioning between states and performing read/write actions on the tape. At this level, the description of a TM is similar to that of a computer. The differences, as pointed out in [20], are that the computer's memory is not infinite, and it is accessed randomly rather than sequentially. But these differences are relatively minor, and the following claim has been made:

Nature of computers: TMs serve as a general model for computers.

This claim has been used in support of the Turing Thesis myth.

Just as for any other class of automata such as FSA (finite state automata), the set-theoretic definition does not completely capture TMs; it captures their *syntax*, but not *semantics*.

Syntax of a TM: what does it consist of?
Semantics: how does it compute?

As defined by Turing [18], TM semantics prescribe that every computation starts in an identical configuration (except for the contents of the read/write tape), and the contents of the tape cannot be modified from the "outside" during the computation. This can be contrasted with Turing's alternative models of *choice machines* and *oracle machines*. The complete TM definition includes both the set-theoretic syntactic definition and the semantic definition.

> Statements about TM expressiveness, such as the Turing Thesis, fundamentally depend on their semantics, as defined by Turing. If these semantics were defined differently, it may (or may not) produce an equivalent machine.

Early computers did in fact compute as prescribed above. However, while perhaps reflecting TM syntax, the computation of modern computers is no longer based on the same semantics. Unlike TMs, new inputs arrive continuously (think of an operating system, or a document processor); the output is also produced continuously (in case of the document processor, it is the screen display of the document). There is I/O entanglement; later inputs are affected by earlier outputs and vise-versa. All this renders a computer's behavior non-functional; it no longer computes a function from the input to the output, and TM no longer serves as an appropriate model for this *interactive* computation.

5.2 The Universal Turing Machine

A *Universal Turing Machine* is a special TM introduced by Turing [18], that can simulate any other TM. It served as the inspiration for the notion of *general-purpose computing*. Turing himself saw a direct parallel between the capability of a computer to accept and execute programs, and his notion of a universal machine.

The principle of universality can easily be extended to any other class of algorithmic machines. As long as each machine in that class can be captured by a finite description, prescribing what this machine would do in every possible configuration, a TM can be created to simulate all machines in that class:

Universality Thesis: Any class of effective devices for computing functions can be simulated by a TM.

Analogously to the Turing Thesis, the Universality Thesis combines with the mathematical worldview to obtain the following corollary:

Universality Corollary: Any computer can be simulated by a TM.

This corollary is the second of the two claims that have been used to "corroborate" the Turing Thesis myth. Again, the undergraduate textbooks played a key role. In order to present the expressiveness of TMs in a more accessible (and dramatic) fashion, the Universality Corollary was melded with the Turing Thesis Corollary, resulting in the following statement that (incorrectly) summarized the role of TMs:

Strong Turing Thesis: A TM can do anything that a computer can do.

6 Time for New Models

6.1 Extending Turing Machines

The history of modifying or extending TMs is at least as old as the theory of computation. By TMs, we mean Turing's *automatic machines* as defined in his original paper [18], and all the versions of these machines that are equivalent to the original. Indeed, the versions one obtains by modifying or extending TMs are *not* TMs, unless and until equivalence with the original has been proven. For example, Turing's automatic machines had a binary alphabet and an infinite tape; the Turing machine we use now typically has an arbitrary alphabet and a semi-infinite tape. Before this version could be called a Turing machine, a proof was needed of the equivalence of the two models. In general, the equivalence of TM versions cannot be taken for granted. For example, if only right transitions are allowed, the resulting model is not equivalent, having the expressiveness of an FSA rather than a TM.

The example above is a *restriction* on TM computations. More common are *extensions*. All TM extensions that can be found in theory textbooks, such as increasing the number of tapes or changing the alhabet, are algorithmic. In the

case of algorithmic extensions, the Church-Turing thesis applies, and it can be taken from granted that the new model is equivalent to the original. However, as a result of the Turing Thesis myth, it is common to assume the equivalence of any TM extension to the original, and we no longer expect formal proofs of this.

To capture the contemporary interactive use of computers, the more recent TM extensions have tended to be non-algorithmic, with computation that spans multiple inputs and outpus to the underlying TM. Nowadays we consider non-terminating interactive computations of Turing machines, persistent Turing machines, nets of Turing machines, Turing machines with evolvable architecture, etc., exactly for reasons of capturing "all computers", or "all computations". Indeed, the recent proliferation of such models can be viewed as a paradigm shift in the field of models of compuation.

While they usually share TM syntax, the semantics of these new extensions is different; for example, in the case of persistent Turing machines, it is based on dynamic input streams and persistence. Out of habit, researchers have continued to assume that these extensions are equivalent to the original TM. But in the case of such non-algorithmic extensions, Turing's thesis does not apply, and equivalence can no longer be taken for granted. Indeed, it no longer holds, as discussed next.

6.2 Modeling Interactive Computation

Wegner [21, 22] has conjectured that interactive models of computation are more expressive than "algorithmic" ones such as Turing machines. It would therefore be interesting to see what minimal extensions are necessary to Turing machines to capture the salient aspects of interactive computing. Motivated by these goals, [8] investigates a new way of interpreting Turing-machine computation, one that is both interactive and persistent – *persistent Turing machines* (PTMs); we discuss this work here.

A PTM is a nondeterministic 3-tape Turing machine (N3TM) with a read-only input tape, a read/write work tape, and a write-only output tape. Upon receiving an input token from its environment on its input tape, a PTM computes for a while and then outputs the result to the environment on its output tape, and this process is repeated forever. A PTM computes *concurrently* with its *environment*, both acting as consumers of each other's outputs and producers of each other's inputs.

In addition to having *dynamic stream semantics*, PTM computations are *persistent* in the sense that a notion of "memory" (work-tape contents) is maintained from one computation step to the next, where each PTM computation step represents an N3TM computation. *Decider PTMs* are an important subclass of PTMs; a PTM is a *decider* if it does not have divergent (non-halting) computations.

The notions of interaction and persistence in PTMs are formalized in terms of the *persistent stream language* (PSL) of a PTM. Given a PTM, its persistent stream language is the set of infinite sequences (interaction streams) of pairs

of the form (w_i, w_o) representing the input and output strings of a single PTM computation step. Persistent stream languages induce a natural, stream-based notion of equivalence for PTMs.

The first result concerning PTMs is that the class of PTMs is isomorphic to *interactive transition systems* (ITS), a very general kind of effective transition systems, thereby allowing one to view PTMs as ITSs "in disguise". ITSs come with three notions of behavioral equivalence: *ITS isomorphism*, *interactive bisimulation*, and *interactive stream equivalence*, where ITS isomorphism refines interactive bisimulation, and interactive bisimulation refines interactive stream equivalence.

A similar result is established for decider PTMs and decider ITSs. These results address a question heretofore left unanswered concerning the relative expressive power of Turing machines and transition systems, namely: "What extensions are required of Turing machines so that they can simulate transitions systems?"

An infinite hierarchy is defined, of successively finer equivalences for PTMs over finite interaction-stream prefixes; it is shown that the limit of this hierarchy does *not* coincide with PSL-equivalence. The presence of this "gap" can be attributed to the fact that the transition systems corresponding to PTM computations naturally exhibit *unbounded nondeterminism*. In contrast, classical Turing-machine computations have bounded nondeterminism, i.e., any nondeterministic TM can produce only a finite number of distinct outputs for a given input string.

Amnesic PTMs are a special type of PTMs where each new computation begins with a blank work tape, with a corresponding notion of *amnesic stream languages* (ASLs). The class of ASLs is strictly contained in the class of PSLs. Also, ASL-equivalence coincides with the equivalence induced by considering interaction-stream prefixes of length one, the bottom of the equivalence hierarchy; this hierarchy therefore collapses in the case of amnesic PTMs.

ASLs are representative of the classical view of Turing-machine computation, extending TMs with dynamic stream-based semantics but without persistence. One may consequently conclude that, in a stream-based setting, the extension of the Turing-machine model with persistence is a nontrivial one.

Finally, the notion of a *universal PTM* is introduced. Similarly to a *universal Turing machine*, a universal PTM can simulate the behavior of an arbitrary PTM. The class of *sequential interactive computations* is also introduced:

Sequential Interactive Computation: A sequential interactive computation continuously interacts with its environment by alternately accepting an input string and computing a corresponding output string. Each output-string computation may be both nondeterministic and history-dependent, with the resultant output string depending not only on the current input string, but also on all previous input strings.

Examples of sequential interaction include sequential JAVA objects, static C routines, single-user databases, network protocols, and our original example of

driving home from work. A sequential interactive analogue to the Turing Thesis is provided:

Sequential Interaction Thesis: Any sequential interactive computation can be performed by a persistent Turing machine.

This hypothesis, when combined with other results in the paper, implies that the class of sequential interactive computations is more expressive than the class of algorithmic computations, and thus is capable of solving a wider range of problems – proving Wegner's conjecture.

TMs capture *effective* transformations over finite strings, but the notion of effectiveness also applies to operations on higher-level objects such as functions, see the theory of *recursive functionals* [14, 15]. Extensions are needed to the Church-Turing thesis to capture this effectiveness [15]; it is conjectured that the Sequential Interaction Thesis captures the expressiveness of recursive functionals.

It has been also conjectured [22] that *multi-agent* interaction is more expressive than sequential, or *single-agent* interaction. These conjectures remain to be proven.

6.3 Turing Thesis Myth Corrected

We have discussed the origins for the popularity of the Turing Thesis myth, having identified three distinct claims that comprise it:

Claim 1. (Mathematical worldview) All computable problems are function-based.
Claim 2. (Focus on algorithms) All computable problems can be described by an algorithm.
Claim 3. (Practical approach) Algorithms are what computers do.

Furthermore, we looked at two more claims that have been used to corroborate the Turing Thesis myth:

Claim 4. (Nature of computers) TMs serve as a general model for computers.
Claim 5. (Universality corollary) TMs can simulate any computer.

For each of these claims, there is a grain of truth. By reformulating them to bring out the hidden assumptions, misunderstandings are removed. The following versions of the above statements are correct:

Corrected Claim 1. All algorithmic problems are function-based.
Corrected Claim 2. All function-based problems can be described by an algorithm.
Corrected Claim 3. Algorithms are what early computers used to do.
Corrected Claim 4. TMs serve as a general model for early computers.
Corrected Claim 5. TMs can simulate any algorithmic computing device.

Furthermore, the following claim is also correct:

Claim 6: TMs cannot compute all problems, nor can they do everything that real computers can do.

This claim, while incompatible with original claims, is perfectly consistent with their corrected versions. It contradicts the Strong Turing Thesis, exposing the fallacy of the Turing Thesis myth. Dispelling this myth has grown more important as the practice of computing is becoming more and more interactive. Its algorithmic fundamental identity no longer serves it well.

7 Conclusion

Hoare, Milner and others have long realized that TMs do not model all of computation [23]. However, when their theory of concurrent computation was first developed in the late '70s, it was premature to openly challenge TMs as a complete model of computation. Concurrency theory positions *interaction* as orthogonal to *computation*, rather than a part of it. By separating interaction from computation, the question whether the models for CCS and the π-calculus went beyond Turing machines and algorithms was avoided.

Researchers in other areas of theoretical computer science have also found need for interactive models of computation, such as Input/Output automata for distributed algorithms [11] and Interactive TMs for interactive proofs [7]. However, the issue of the expressiveness of interactive models vis-a-vis TMs was not raised until the mid-1990's, when the model of interaction machines as a more expressive extension of TMs was first proposed by one of the authors [21].

The theoretical framework for sequential interaction machines, as a persistent stream-based extension to TMs, was completed by the other author in [8]; it is discussed above. Van Leeuwen, a Dutch expert on the theory of computation, proposed an alternate extension in [19]. In addition to interaction, other ways to extend computation beyond Turing machines have been considered, such as quantum computing.

While not part of CS Theory, the field of AI has perhaps gone the furthest in explicitly recognizing the expressiveness gains of moving beyond algorithms. In the early 1990's, Rodney Brooks convincingly argued against the algorithmic approach of "good old-fashioned AI", positioning interaction is a prerequisite for intelligent system behavior [3]. This argument has been adopted by the mainstream AI community, whose leading textbooks recognize that interactive *agents* are a better model of intelligent behaviors than simple input/output functions [13].

In the last three decades, computing technology has shifted from mainframes and microstations to networked embedded and mobile devices, with the corresponding shift in applications from number crunching and data processing to the Internet and ubiquitous computing. We believe it is no longer premature to encompass interaction as part of computation. A paradigm shift is necessary in our notion of computational problem solving so it can better model the services provided by today's computing technology.

The *Strong Church-Turing Thesis* reinterprets the Church-Turing Thesis to imply that Turing Machines model all computation. In this paper, we provided a new analysis of the historical reasons for the widespread acceptance of the myth that the two versions of the thesis are equivalent. However, the assumption that all of computation can be algorithmically specified is still widely accepted in the CS community, and interaction machines have been criticized as an unnecessary Kuhnian paradigm shift.

By facing the fact that this reinterpretation is a myth we can move forward with formal models of interaction machines, which extend Turing machines with interaction to capture the behavior of concurrent systems. We presented one such model, *Persistent Turing Machines* (PTMs), that promises to bridge theory of computation and concurrency theory. PTMs capture sequential interaction, which is a limited form of concurrency. They can also be viewed as *interactive transition systems*, with corresponding notions of observational equivalence. Furthermore, they are have been shown to be more expressive than Turing machines. It is hoped that PTMs will lay the foundation for a new theory of interactive computation which will bridge the current theories of computation and concurrency.

References

1. An Undergraduate Program in Computer Science-Preliminary Recommendations, A Report from the ACM Curriculum Committee on Computer Science. *Comm. ACM* 8(9), Sep. 1965, pp. 543-552.
2. Curriculum 68: Recommendations for Academic Programs in Computer Science, A Report of the ACM Curriculum Committee on Computer Science. *Comm. ACM* 11(3), Mar. 1968, pp. 151-197.
3. R. Brooks. Intelligence Without Reason. *MIT AI Lab Technical Report 1293.*
4. M. Davis. *Computability & Unsolvability.* McGraw-Hill, 1958.
5. P. Denning. The Field of Programmers Myth. *Comm. ACM*, July 2004.
6. E. Eberbach, D. Goldin, P. Wegner. Turing's Ideas and Models of Computation. In *Alan Turing: Life and Legacy of a Great Thinker*, ed. Christof Teuscher, Springer 2004.
7. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comp.*, 18(1):186–208, 1989.
8. D. Goldin, S. Smolka, P. Attie, E. Sonderegger. Turing Machines, Transition Systems, and Interaction. *Information & Computation J.*, Nov. 2004.
9. J.E. Hopcroft, J.D. Ullman. *Formal languages and their relation to automata.* Addison-Wesley, 1969.
10. D. Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms.* Addison-Wesley, 1968.
11. N. Lynch, M. Tuttle. An Introduction to Input/Output automata. *CWI Quarterly*, 2(3):219–246, Sep. 1989. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.
12. J. K. Rice, J. N. Rice. *Computer Science: Problems, Algorithms, Languages, Information and Computers.* Holt, Rinehart and Winston, 1969.
13. S. Russell, P. Norveig. *Artificial Intelligence: A Modern Approach.* Addison-Wesley, 1994.

14. H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.
15. L. Sanchis. *Recursive Functionals*, North Holland, 1992.
16. *SIGACT News*, ACM Press, March 2004, p. 49.
17. M. Sipser. *Introduction to the Theory of Computation*, PWS Publishing Company, 1997.
18. A. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem, *Proc. London Math. Soc.*, 42:2, 1936, pp. 230-265; A correction, ibid, 43, 1937, pp. 544-546.
19. J. van Leeuwen, J. Wiedermann. The Turing Machine Paradigm in Contemporary Computing. in *Mathematics Unlimited - 2001 and Beyond*, eds. B. Enquist and W. Schmidt, Springer-Verlag, 2000.
20. P. Wegner. *Programming Languages, Information Structures and Machine Organization*, McGraw-Hill, 1968.
21. P. Wegner. Why Interaction is More Powerful Than Algorithms. *Comm. ACM*, May 1997.
22. P. Wegner. Interactive Foundations of Computing. *Theoretical Computer Science* 192, Feb. 1998.
23. P. Wegner, D. Goldin. Computation Beyond Turing Machines. *Comm. ACM*, Apr. 2003.

Robust Simulations of Turing Machines with Analytic Maps and Flows*

Daniel S. Graça[1,2,**], Manuel L. Campagnolo[3,2], and Jorge Buescu[4]

[1] Departamento de Matemática, Faculdade de Ciências e Tecnologia,
Universidade do Algarve, Campus de Gambelas,
8005-139 Faro, Portugal
dgraca@ualg.pt
[2] Center for Logic and Computation, Departamento de Matemática,
Instituto Superior Técnico, Universidade Técnica de Lisboa,
1049-001 Lisboa, Portugal
[3] Departamento de Matemática, Instituto Superior de Agronomia,
Universidade Técnica de Lisboa,
1349-017 Lisboa, Portugal
mlc@math.isa.utl.pt
[4] Center for Mathematical Analysis, Geometry and Dynamical Systems,
Departamento de Matemática, Instituto Superior Técnico,
Universidade Técnica de Lisboa,
1049-001 Lisboa, Portugal
jbuescu@math.ist.utl.pt

Abstract. In this paper, we show that closed-form analytic maps and flows can simulate Turing machines in an error-robust manner. The maps and ODEs defining the flows are explicitly obtained and the simulation is performed in real time.

1 Introduction

Since the pioneering work of Turing in the 1930s, the Turing machine has become the standard paradigm for computation. With the appearance and rapid development of digital computers its role has become increasingly important. In this paper we show that the behavior of Turing machines can be embedded in robust and analytic analog systems defined on continuous spaces.

Several authors have proved that finite dimensional maps and flows can simulate Turing machines. The general approach is to associate each configuration of a Turing machine to a point of $\mathbb{R}^n$, and to show that there is a dynamical system with state space

* Our interest in the questions addressed in this paper was raised by past discussions with Félix Costa and Cris Moore. This work was partially supported by *Fundação para a Ciência e a Tecnologia* (FCT) and FEDER via the Center for Logic and Computation - CLC, the project ConTComp POCTI/MAT/45978/2002 and grant SFRH/BD/17436/2004. Additional support was also provided by the *Fundação Calouste Gulbenkian* through the *Programa Gulbenkian de Estímulo à Investigação*.

** D. Graça wishes to thank Carlos Lourenço for helpful discussions.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 169–179, 2005.

in $\mathbb{R}^n$ that embeds its evolution. It is known that Turing machines can be simulated on compact spaces, even of low dimension [1, 2, 3]. While compactness is a desirable property of dynamical systems, it is probably too strong a requirement since it is believed that no analytic map on a compact, finite dimensional space can simulate a Turing machine through a reasonable encoding [4]. However, most physical systems turn out to be analytic, at least in the classical world of Physics. Even the physical model underlying digital computers is analytic, although their behavior is idealized as discrete.

The requirement of compactness has another drawback since it prevents systems capable of simulating an arbitrary Turing machine to exhibit robustness to noise. For instance, Casey [5, 6] showed that in the presence of bounded analog noise, recurrent neural networks can only recognize regular languages. This result was later generalized in [7] to other analog discrete-time computational systems. Robustness is a critical issue in analog models since non-computable behavior might arise when the use of exact real quantities is allowed. For instance, the results of Pour-El, Richards and Zhong [8, 9] show that there is a three-dimensional wave equation, with computable initial conditions, such that its unique solution is not computable. However, that behavior is ruled out in the presence of noise [10]. Recurrent analog neural networks are another known case where non-computable behavior can occur if real parameters are represented with infinite precision [3].

In this paper we will show that Turing machines can be simulated by finite dimensional maps and flows which are both analytic and robust. We will consider simulations on unbounded spaces. Our work is in some sense related to [11], where a constructive simulation of Turing machines using closed-form analytic maps is presented. However, in [11] it is not discussed how the presence of noise affects the computational power of the model. We prove here that any Turing machine M can be simulated by a closed-form analytic map $f_M : \mathbb{R}^3 \to \mathbb{R}^3$, even in the case where some noise is added to the initial configuration or during the evolution of the system.

The previously mentioned results show that finite dimensional maps are capable of simulating the *transition function* of an arbitrary Turing machine. In that respect, those are results about the computational power of hybrid systems, which are continuous with respect to the state space but evolve discretely in time. Another approach has been to simulate the evolution of Turing machines with continuous flows in $\mathbb{R}^n$ [12, 13, 14]. Even if it is known that those flows can be infinitely differentiable, no analytic form of iterating the map that simulates the transition function of a Turing machine had been proposed before. Furthermore, it is known that analytic differentially algebraic functions, which include most of the usual mathematical analytic functions, are not closed under iteration [15], which suggests that continuous-time computational models which are closed under iteration must contain some non-analytic functions [16]. However, since we only have to iterate functions in the vicinity of integers, we are able to show that any Turing machine M can be robustly simulated by some system of differential equations $y' = g_M(y, t)$, where g_M is analytic and t represents the time steps of M.

It is worthwhile to notice that our work can be included, in some sense, in the wider topic of stable dynamical systems. In fact, there has been a long tradition of considering only structurally stable systems [17] when modelling physical systems. The argument is that, due to measurement uncertainties, qualitative properties of a system should not

change with small perturbations. Guckenheimer and Holmes [18] refer to this approach as the "stability dogma". However, recent developments in the theory of dynamical systems suggest that this is too restrictive to account for all meaningful systems [19]. In fact, one can relax the previous condition and demand stability only for those properties of interest for the system under consideration. Here, we have chosen the latter line of work: our only concern is that each system performs a simulation of a Turing machine robust to perturbations.

The paper can be outlined as follows. In Section 2 we introduce the ideas and concepts related to simulations robust to perturbations. Section 3 provides tools that will be necessary in Section 4. In Section 4, we prove (in a constructive manner) the main results of this paper: each Turing machine can be simulated by an analytic map, or by ODEs even under the influence of (small) errors. The maps and ODEs are explicitly obtained, by using expressions involving the composition of polynomials and trigonometric functions, and only computable constants are used. We end describing some connections of this paper with previous results on continuous-time models of computation.

2 Simulation of Turing Machines

Before stating the main results, we describe succinctly some aspects of our error-robust simulation of Turing machines. For now, we will be only concerned with discrete time simulations. Therefore we want to obtain a map that "captures" the behavior of the transition function. We will code each configuration into a triple $(x, y, z) \in \mathbb{N}^3$, and prove that the simulation still works if this triple is slightly perturbed. Without loss of generality, consider a Turing machine M using 10 symbols, the blank symbol $B = 0$, and symbols $1, 2, ...9$. Let

$$...B\,B\,B\,a_{-k}\,a_{-k+1}...\,a_{-1}\,a_0\,a_1...\,a_n\,B\,B\,B... \tag{1}$$

represent the tape contents of the Turing machine M. We suppose the head to be reading symbol a_0 and $a_i \in \{0, 1, ..., 9\}$ for all i. We also suppose that M has m states, represented by numbers 1 to m. For convenience, we consider that if the machine reaches a halting configuration it moves to the same configuration. We assume that, in each transition, the head either moves to the left, moves to the right, or does not move. Take

$$y_1 = a_0 + a_1 10 + ... + a_n 10^n \qquad y_2 = a_{-1} + a_{-2} 10 + ... + a_{-k} 10^{k-1}$$

and let q be the state associated to the current configuration. Then the triple $(y_1, y_2, q) \in \mathbb{N}^3$ gives the current configuration of M. We now can state the first main result of this paper as follows:[1]

Theorem 1. *Let $\theta : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of a Turing machine M, under the encoding described above and let $0 < \delta < \varepsilon < 1/2$. Then θ admits an analytic extension $f_M : \mathbb{R}^3 \to \mathbb{R}^3$, robust to perturbations in the following sense: for all f such*

[1] We take $\|(x_1, ..., x_n)\|_\infty = \max_{1 \le i \le n} |x_i|$ and $\|f\|_\infty = \sup_{x \in \mathbb{R}} \|f(x)\|_\infty$, where f is a real function. If $f : A \to A$ is a function, then $f^{[k]}$ denotes the kth iterate of f.

that $\|f - f_M\|_\infty \leq \delta$ *and for all* $\bar{x}_0 \in \mathbb{R}^3$ *satisfying* $\|\bar{x}_0 - x_0\|_\infty \leq \varepsilon$, *where* $x_0 \in \mathbb{N}^3$ *represents an initial configuration,*

$$\left\| f^{[j]}(\bar{x}_0) - \theta^{[j]}(x_0) \right\|_\infty \leq \varepsilon \qquad \text{for all } j \in \mathbb{N}.$$

A few remarks are in order. First, and as noticed before, we implicitly assumed that if y is a halting configuration, then $\theta(y) = y$. Secondly, we notice that the upper bound ($\frac{1}{2}$) on ε results from the encoding we have chosen, which is over the integers. In fact, the bound is maximal with respect to that encoding.

Incidentally, we notice that Theorem 1 can be stated using the notion of *shadowing* in dynamical systems (cf. [20, 21]), which is formally defined as below.

Definition 2. *Let* $f : A \to A$ *be a map,* $\varepsilon > 0$, *and* $\{p_i\}_{i\in\mathbb{N}} \subseteq A$. *Then* $\{p_i\}_{i\in\mathbb{N}}$ *is a* ε*-pseudo-orbit of* f *if* $|p_{i+1} - f(p_i)| < \varepsilon$ *for all* $i \in \mathbb{N}$. *For* $x \in A$, *we say that* $\{f^{[i]}(x)\}_{i\in\mathbb{N}}$ ε*-shadows the pseudo-orbit* $\{p_i\}_{i\in\mathbb{N}}$ *if* $\left|f^{[i]}(x) - p_i\right| < \varepsilon$.

In short, we say that $\{p_i\}_{i\in\mathbb{N}}$ is a good approximation of some system whose dynamics is given by f, if $\{f^{[i]}(x)\}_{i\in\mathbb{N}}$ ε-shadows $\{p_i\}_{i\in\mathbb{N}}$. Using the previous definition, we can restate Theorem 1 by saying that the sequence $\{f_M^{[j]}(x_0)\}_{j\in\mathbb{N}}$ of configurations ε-shadows $\{f^{[j]}(\bar{x}_0)\}_{j\in\mathbb{N}}$. We now present the other main results.

Theorem 3. *Let* $\theta : \mathbb{N}^3 \to \mathbb{N}^3$ *be the transition function of a Turing machine M, under the encoding described above and let* $0 < \varepsilon < 1/4$. *Then there is an analytic function* $z : \mathbb{R}^4 \to \mathbb{R}^3$ *with the following property:*

$$\left\| z(x_0, j) - \theta^{[j]}(x_0) \right\|_\infty \leq \varepsilon$$

for all $j \in \mathbb{N}$, *where* $x_0 \in \mathbb{N}^3$ *represents an initial configuration.*

As a matter of fact, we will prove the following "robust" version of Theorem 3.

Theorem 4. *In the conditions of Theorem 3, there is an analytic function* $g_M : \mathbb{R}^6 \to \mathbb{R}^6$ *such that the ODE* $z' = g_M(z, t)$ *robustly simulates* M *in the following sense: there is some* $0 < \eta < 1/2$ *such that for all* g *satisfying* $\|g - g_M\|_\infty < 1/2$, *and for all* $\bar{x}_0 \in \mathbb{R}^3$ *satisfying* $\|\bar{x}_0 - x_0\|_\infty \leq \varepsilon$, *the solution* z *of*

$$z' = g(z, t), \qquad z(0) = (\bar{x}_0, \bar{x}_0)$$

has the following property: for all $j \in \mathbb{N}$ *and for all* $t \in [j, j + 1/2]$,[2]

$$\left\| z_2(t) - \theta^{[j]}(x_0) \right\|_\infty \leq \eta.$$

[2] For simplicity, we denote z by (z_1, z_2), where $z_1, z_2 \in \mathbb{R}^3$.

3 Preliminary Results

This section is devoted to the presentation of results that, while not very interesting on their own, will be useful when proving Theorem 1. As our first task, we introduce an analytic extension $\omega : \mathbb{R} \to \mathbb{R}$ for the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = n \bmod 10$. This function will be necessary when simulating Turing machines. It will be used to read symbols written in the tape. To achieve this purpose, we can use a periodic function, of period 10, such that $\omega(i) = i$, for $i = 0, 1, ..., 9$. Then, using trigonometric interpolation (cf. [22–pp. 176-182]), one may take

$$\omega(x) = a_0 + a_5 \cos(\pi x) + \left(\sum_{j=1}^{4} a_j \cos\left(\frac{j\pi x}{5}\right) + b_j \sin\left(\frac{j\pi x}{5}\right) \right), \qquad (2)$$

where $a_0, ..., a_5, b_1, ..., b_4$ are computable coefficients that can be explicitly obtained by solving a system of linear equations.

It is easy to see that ω is uniformly continuous in $\mathbb{R}$. Hence, for every $\varepsilon \in (0, 1/2)$, there will be some $\zeta_\varepsilon > 0$ satisfying

$$\forall n \in \mathbb{N},\ x \in [n - \zeta_\varepsilon, n + \zeta_\varepsilon] \ \Rightarrow\ |\omega(x) - n \bmod 10| \leq \varepsilon. \qquad (3)$$

When simulating a Turing machine, we will also need to keep the error under control. In many cases, this will be done with the help of the error-contracting function defined by

$$\sigma(x) = x - 0.2 \sin(2\pi x).$$

The function σ is a contraction on the vicinity of integers:

Lemma 5. *Let $n \in \mathbb{Z}$ and let $\varepsilon \in [0, 1/2)$. Then there is some contracting factor $\lambda_\varepsilon \in (0, 1)$ such that, for $\forall \delta \in [-\varepsilon, \varepsilon]$, $|\sigma(n + \delta) - n| < \lambda_\varepsilon \delta$.*

Remark 6. *Throughout the remainder of this paper, we suppose that $\varepsilon \in [0, 1/2)$ is fixed and that λ_ε is the respective contracting factor given by Lemma 5.*

The function σ will be used in our simulation to keep the error controlled when bounded quantities are involved (e.g., the actual state, the symbol being read, etc.). We will also need another error-contracting function that controls the error for unbounded quantities. This will be achieved with the help of the function $l_3 : \mathbb{R}^2 \to \mathbb{R}$, that has the property that whenever $\bar{a}$ is an approximation of $a \in \{0, 1, 2\}$, then $|l_3(\bar{a}, y) - a| < 1/y$, for $y > 0$. In other words, l_3 is an error-contracting map, where the error is contracted by an amount specified by the second argument of l_3. We start by defining a preliminary function l_2 satisfying similar conditions, but only when $a \in \{0, 1\}$.

Lemma 7. *Let $l_2 : \mathbb{R}^2 \to \mathbb{R}$ be given by $l_2(x, y) = \frac{1}{\pi} \arctan(4y(x - 1/2)) + \frac{1}{2}$. Suppose also that $a \in \{0, 1\}$. Then, for any $\bar{a}, y \in \mathbb{R}$ satisfying $|a - \bar{a}| \leq 1/4$ and $y > 0$, we get $|a - l_2(\bar{a}, y)| < 1/y$.*

Lemma 8. *Let $a \in \{0,1,2\}$ and let $l_3 : \mathbb{R}^2 \to \mathbb{R}$ be given by*

$$l_3(x,y) = l_2((\sigma^{[d+1]}(x)-1)^2, 3y).(2l_2(\sigma^{[d]}(x)/2, 3y) - 1) + 1,$$

where $d = 0$ if $\varepsilon \leq 1/4$ and $d = \lceil -\log(4\varepsilon)/\log \lambda_\varepsilon \rceil$ otherwise. Then for any $\bar{a}, y \in \mathbb{R}$ satisfying $|a - \bar{a}| \leq \varepsilon$ and $y \geq 2$, we have $|a - l_3(\bar{a}, y)| < 1/y$.

The following lemma can be easily proved by induction on n.

Lemma 9. *If $|\alpha_i|, |\bar{\alpha}_i| \leq K$ for $i = 1, ..., n$ then*

$$|\alpha_1 ... \alpha_n - \bar{\alpha}_1 ... \bar{\alpha}_n| \leq (|\alpha_1 - \bar{\alpha}_1| + ... + |\alpha_n - \bar{\alpha}_n|)\, K^{n-1}.$$

4 Robust Analytic Simulations of Turing Machines

In this section we show, in a constructive manner, how to simulate a Turing machine with an analytic map robust to (small) perturbations. We will first prove the following theorem.

Theorem 10. *Let $\theta : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of some Turing machine. Then, given some $0 \leq \varepsilon < 1/2$, θ admits an analytic extension $h_M : \mathbb{R}^3 \to \mathbb{R}^3$ with the property that*

$$\|(y_1, y_2, q) - (\bar{y}_1, \bar{y}_2, \bar{q})\|_\infty \leq \varepsilon \;\Rightarrow\; \|\theta(y_1, y_2, q) - h_M(\bar{y}_1, \bar{y}_2, \bar{q})\|_\infty \leq \varepsilon. \quad (4)$$

Proof. We will show how to construct h_M with analytic functions:

1. **Determine the symbol being read.** Let a_0 be the symbol being actually read by the Turing machine M. Then $\omega(y_1) = a_0$, where ω is given by (2). But what about the effect of the error present in $\bar{y}_1$? Since $|y_1 - \bar{y}_1| \leq \varepsilon$,

$$|a_0 - \omega \circ \sigma^{[l]}(\bar{y}_1)| \leq \varepsilon, \quad \text{with } l = \left\lceil \left| \frac{\log(\zeta_\varepsilon/\varepsilon)}{\log \lambda_\varepsilon} \right| \right\rceil, \quad (5)$$

   where ζ_ε is given by (3). Then pick $\bar{y} = \omega \circ \sigma^{[l]}(\bar{y}_1)$ as an approximation of the symbol being currently read. Similarly, $\omega \circ \sigma^{[l]}(\bar{y}_2)$ gives an approximation of a_{-1}, with error bounded by ε.

2. **Determine the next state.** The map that returns the next state is defined by polynomial interpolation. This can be done as follows. Let y be the symbol being currently read and q the current state. Recall that m denotes the number of states and $k = 10$ is the number of symbols. One may take

$$q_{next} = \sum_{i=0}^{9} \sum_{j=1}^{m} \left(\prod_{\substack{r=0 \\ r \neq i}}^{9} \frac{(y-r)}{(i-r)} \right) \left(\prod_{\substack{s=1 \\ s \neq j}}^{m} \frac{(q-s)}{(j-s)} \right) q_{i,j},$$

where $q_{i,j}$ is the state that follows symbol i and state j. However, we are dealing with the approximations $\bar{q}$ and $\bar{y}$. Therefore, we define instead

$$\bar{q}_{next} = \sum_{i=0}^{9}\sum_{j=1}^{m}\left(\prod_{\substack{r=0\\ r\neq i}}^{9}\frac{(\sigma^{[n]}(\bar{y})-r)}{(i-r)}\right)\left(\prod_{\substack{s=1\\ s\neq j}}^{m}\frac{(\sigma^{[n]}(\bar{q})-s)}{(j-s)}\right)q_{i,j}, \tag{6}$$

with

$$n = \left\lceil \frac{\log(10m^2K^{m+7}(m+8))}{-\log\lambda_\varepsilon}\right\rceil, \qquad K = \max\{9.5, m+1/2\}.$$

With this choice for n, the error of $\sigma^{[n]}(\bar{y})$ and $\sigma^{[n]}(\bar{q})$ is such that

$$9|y-\sigma^{[n]}(\bar{y})| + (m-1)|q-\sigma^{[n]}(\bar{q})| \leq \frac{\varepsilon}{10m^2K^{m+7}}. \tag{7}$$

Thus, from (6), (7) and Lemma 9, we conclude that $|\bar{q}_{next} - q_{next}| \leq \varepsilon$.

3. **Determine the symbol to be written on the tape.** Using a similar construction, the symbol to be written, s_{next}, can be approximated with precision ε, i.e. $|s_{next} - \bar{s}_{next}| \leq \varepsilon$.

4. **Determine the direction of the move for the head.** Let h denote the direction of the move of the head, where $h = 0$ denotes a move to the left, $h = 1$ denotes a "no move", and $h = 2$ denotes a move to the right. Then, again, the "next move" h_{next} can be approximated by means of a polynomial interpolation as in steps 3 and 4, therefore obtaining $|h_{next} - \bar{h}_{next}| \leq \varepsilon$.

5. **Update the tape contents.** We define functions $\bar{P}_1, \bar{P}_2, \bar{P}_3$ which are intended to approximate the tape contents after the head moves left, does not move, or moves right, respectively. Let H be a "sufficiently good" approximation of h_{next}, yet to be determined. Then, the next value of y_1, y_1^{next}, can be approximated by

$$\bar{y}_1^{next} = \bar{P}_1\,\tfrac{1}{2}(1-H)(2-H) + \bar{P}_2\,H(2-H) + \bar{P}_3\,(-\tfrac{1}{2})H(1-H), \tag{8}$$

with

$$\bar{P}_1 = 10(\sigma^{[j]}(\bar{y}_1) + \sigma^{[j]}(\bar{s}_{next}) - \sigma^{[j]}(\bar{y})) + \sigma^{[j]}\circ\omega\circ\sigma^{[l]}(\bar{y}_2),$$
$$\bar{P}_2 = \sigma^{[j]}(\bar{y}_1) + \sigma^{[j]}(\bar{s}_{next}) - \sigma^{[j]}(\bar{y}), \quad \bar{P}_3 = \frac{\sigma^{[j]}(\bar{y}_1) - \sigma^{[j]}(\bar{y})}{10},$$

where $j \in \mathbb{N}$ is sufficiently large and l is given by (5). Notice that when exact values are used, $\bar{y}_1^{next} = y_1^{next}$. The problem in this case is that $\bar{P}_1$ depends on $\bar{y}_1$, which is not a bounded value. Thus, if we simply take $\bar{H} = \bar{h}_{next}$, the error of the term $(1-H)(2-H)/2$ is arbitrarily amplified when this term is multiplied by $\bar{P}_1$. Hence, $\bar{H}$ must be a sharp estimate of h_{next}, proportional to $\bar{y}_1$. Therefore, using Lemma 8 and the definition of y_1, one can see that it is suffices to take

$$H = l_3(\bar{h}_{next}, 10000\,(\bar{y}_1 + 1/2) + 2).$$

Using the same argument for $\bar{P}_2$ and $\bar{P}_3$, we conclude that $|\bar{y}_1^{next} - y_1^{next}| < \varepsilon$. Similarly, and for the left side of the tape, we can define $\bar{y}_2^{next}$ such that $|\bar{y}_2^{next} - y_2^{next}| < \varepsilon$,.

Finally, $h_M : \mathbb{R}^3 \to \mathbb{R}^3$ is defined by $h_M(\bar{y}_1, \bar{y}_2, \bar{q}) = (\bar{y}_1^{next}, \bar{y}_2^{next}, \bar{q}_{next})$. □

We shall now prove the main results of this paper.

Proof of Theorem 1. Let $0 \leq \delta < \varepsilon$. Then, using Theorem 10, one can find a map h_M such that (4) holds. Let $i \in \mathbb{N}$ satisfy $\sigma^{[i]}(\varepsilon) \leq \varepsilon - \delta$. Define a map $f_M = \sigma^{[i]} \circ h_M$. Then, if $x_0 \in \mathbb{N}^3$ is an initial configuration,

$$\|\bar{x}_0 - x_0\|_\infty \leq \varepsilon \;\; \Rightarrow \;\; \|f_M(\bar{x}_0) - \theta(x_0)\|_\infty \leq \varepsilon - \delta.$$

Thus, by triangular inequality, if $\|\bar{x}_0 - x_0\|_\infty \leq \varepsilon$, then

$$\|f(\bar{x}_0) - \theta(x_0)\|_\infty \leq \|f(\bar{x}_0) - f_M(\bar{x}_0)\|_\infty + \|f_M(\bar{x}_0) - \theta(x_0)\|_\infty \leq \delta + (\varepsilon - \delta) = \varepsilon.$$

This proves the result for $j = 1$. For $j > 1$, we proceed by induction. □

Proof of Theorem 4. (Sketch) We adapt the construction in [12] to simulate the iteration of the transition function of a TM with ODEs, using our Theorem 1 to generalize Branicky's construction to analytic and robust flows. To iterate a function θ we use a pair of functions to control the evolution of two "simulation" variables z_1 and z_2. Both simulation variables have values close to x_0 at $t = 0$. The first variable is iterated during half of an unit period while the second remains approximately constant (its derivative is kept close to zero by a control function that involves our error-contracting function l_2). Then, the first variable remains controlled during the following half unit period of time and the second variable is brought up close to it. Therefore, at time $t = 1$ both variables have values close to $\theta(x_0)$. Theorem 1 shows that there exists some analytic function robust to errors that simulates θ. This allow us to repeat the process an arbitrary number of times, keeping the error under control.

We begin with some preliminary results. There exists an ODE whose solution can be as close as desired to an arbitrary fixed value $b \in \mathbb{R}$ at $t = 1/2$, for any initial condition at $t = 0$. Let $\phi : \mathbb{R} \to \mathbb{R}^+$ be some function. For an arbitrary error $\gamma > 0$ we define a perturbed version, where we allow an error $\rho \geq 0$ on b and a perturbation term bounded by $\delta \geq 0$:

$$z' = -c(z - \bar{b}(t))^3 \phi(t) + E(t), \qquad \text{with } c \geq \left(2\gamma^2 \textstyle\int_0^{1/2} \phi(t)dt\right)^{-1}. \tag{9}$$

where $\left|\bar{b}(t) - b\right| \leq \rho$ and $|E(t)| \leq \delta$. Using the theory of ODEs, we can conclude that $\left|z(\frac{1}{2}) - b\right| < \gamma + \rho + \delta/2$ regardless to the initial condition at $t = 0$.

For the control functions mentioned above, we use $s : \mathbb{R} \to [-\frac{1}{8}, 1]$ defined by

$$s(t) = \frac{1}{2}\left(\sin^2(2\pi t) + \sin(2\pi t)\right).$$

On $[0, 1/2]$ s ranges between 0 and 1 and on $[1/2, 1]$ s ranges between $-\frac{1}{8}$ and 0.

We can now present the proof of the theorem. Let M be some Turing machine, let f_M be a map satisfying the conditions of Theorem 10 (replacing ε by γ), and let $\bar{x}_0 \in \mathbb{R}^3$ be an approximation, with error ε, of some initial configuration x_0. Take also $\delta < 1/2$ and $\gamma > 0$ such that $2\gamma + \delta/2 \leq \varepsilon < 1/2$ (we suppose, without loss of generality, that $\delta/2 < \varepsilon$). This condition will be needed later. Consider the system of differential equations $z' = g_M(z,t)$ given by

$$z_1' = c_1(z_1 - f_M \circ \sigma^{[m]}(z_2))^3\, \phi_1(t,z_1,z_2),\ \ z_2' = c_2(z_2 - \sigma^{[n]}(z_1))^3\, \phi_2(t,z_1,z_2) \quad (10)$$

with initial conditions $z_1(0) = z_2(0) = \bar{x}_0$, where

$$\phi_1(t,z_1,z_2) = l_2\left(\theta(t), \tfrac{c_1}{\gamma}(z_1 - f_M \circ \sigma^{[m]}(z_2))^4 + \tfrac{c_1}{\gamma} + 10\right)$$
$$\phi_2(t,z_1,z_2) = l_2\left(\theta(-t), \tfrac{c_2}{\gamma}(z_2 - \sigma^{[n]}(z_1))^4 + \tfrac{c_2}{\gamma} + 10\right).$$

Because we want to show that the ODE $z' = g_M(z,t)$ simulates M in a robust manner, we also assume that an error of amplitude not exceeding δ is added to the right side of the equations in (10). Our simulation variables are z_1, z_2 and the control functions are ϕ_1, ϕ_2. Since ϕ_1, ϕ_2 are analytic they cannot be constant on any open interval as in [12]. However, our construction guarantees that one of the control functions is kept close to zero, while the other one reaches a value close to 1. For instance, on $[0, 1/2]$ $|s(-t)| \leq 1/8$ and, by Lemma 7, ϕ_2 is therefore less than $\gamma(c_2 \left\|z_2 - \sigma^{[n]}(z_1)\right\|_\infty^3)^{-1}$. This guarantees that z_2' is sufficiently small on $[0, 1/2]$ and, therefore,

$$\left\|z_2(\tfrac{1}{2}) - x_0\right\|_\infty < (\gamma + \delta)/2 + \varepsilon < \tfrac{1}{2}.$$

Hence, for m large enough $\|\sigma^{[m]}(z_2) - x_0\| < \gamma$. Moreover, on some subinterval of $[0, 1/2]$ $s(t)$ is close to 1 and therefore ϕ_1 is also close to 1. Thus, the behavior of z_1 is given by (9) and $\left\|z_1(\frac{1}{2}) - \theta(x_0)\right\|_\infty < 2\gamma + \delta/2 \leq \varepsilon$.

Now, for interval $[1/2, 1]$ the roles of z_1 and z_2 are switched. One concludes that if $n \in \mathbb{N}$ is chosen so that $\sigma^{[n]}(5\gamma/2 + \delta) < \gamma$, then $\|z_2(1) - f_M(x_0)\|_\infty < 2\gamma + \delta/2 \leq \varepsilon$. We can repeat this process for z_1 and z_2 on subsequent intervals, which shows that for $j \in \mathbb{N}$, if $t \in [j, j + \frac{1}{2}]$ then $\|z_2(t) - \theta^{[j]}(x_0)\|_\infty \leq \varepsilon$ as claimed. □

Notice that all the functions we use in the proof above are analytic. Moreover, note that if we apply the error-contracting function σ to z_1 we can make the error arbitrarily small. Therefore, Theorem 4 implies Theorem 3.

5 Final Remarks

We showed that robust analytic maps and flows can simulate Turing machines, filling some existing gaps on the literature on this subject.

There are several connections of this work and previous results on continuous-time computational models. In particular, it is not difficult to verify [23] that the function z in Theorem 4 is computable by Shannon's General Purpose Analog Computer (GPAC). Moreover, according to [16] z also belongs to the (analytic) subclass $[0, 1, -1, U; COMP, I]$ of Moore's real recursive functions.

We proved lower computational bounds for analytic systems robust in the sense of Theorems 1 and 4. Can we show that the computational power of those systems lies in the realm of Turing computability, in analogy with the upper bounds in [5] for compact domains? We leave this question to the reader.

References

1. Moore, C.: Unpredictability and undecidability in dynamical systems. Phys. Rev. Lett. **64** (1990) 2354–2357
2. Koiran, P., Cosnard, M., Garzon, M.: Computability with low-dimensional dynamical systems. Theoret. Comput. Sci. **132** (1994) 113–128
3. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural networks. J. Comput. System Sci. **50** (1995) 132–150
4. Moore, C.: Finite-dimensional analog computers: Flows, maps, and recurrent neural networks. In Calude, C., Casti, J., Dinneen, M., eds.: 1st International Conference on Unconventional Models of Computation - UMC'98, Springer (1998) 59–71
5. Casey, M.: The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. Neural Comp. **8** (1996) 1135–1178
6. Casey, M.: Correction to proof that recurrent neural networks can robustly recognize only regular languages. Neural Comp. **10** (1998) 1067–1069
7. Maass, W., Orponen, P.: On the effect of analog noise in discrete-time analog computations. Neural Comput. **10** (1998) 1071–1095
8. Pour-El, M.B., Richards, J.I.: The wave equation with computable initial data such that its unique solution is not computble. Adv. Math. **39** (1981) 215–239
9. Pour-El, M.B., Zhong, N.: The wave equation with computable initial data whose unique solution is nowhere computable. Math. Log. Quart. **43** (1997) 499–509
10. Weihrauch, K., Zhong, N.: Is wave propagation computable or can wave computers beat the Turing machine? Proc. London Math. Soc. **85** (2002) 312–332
11. Koiran, P., Moore, C.: Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. Theoret. Comput. Sci. **210** (1999) 217–223
12. Branicky, M.S.: Universal computation and other capabilities of hybrid and continuous dynamical systems. Theoret. Comput. Sci. **138** (1995) 67–100
13. Campagnolo, M.L., Moore, C., Costa, J.F.: An analog characterization of the Grzegorczyk hierarchy. J. Complexity **18** (2002) 977–1000
14. Mycka, J., Costa, J.F.: Real recursive functions and their hierarchy. J. Complexity **20** (2004) 835–857
15. Campagnolo, M.L., Moore, C., Costa, J.F.: Iteration, inequalities, and differentiability in analog computers. J. Complexity **16** (2000) 642–660
16. Campagnolo, M.L.: The complexity of real recursive functions. In Calude, C.S., Dinneen, M.J., Peper, F., eds.: UMC'02. LNCS 2509. Springer (2002) 1–14
17. Hirsch, M.W., Smale, S.: Differential Equations, Dynamical Systems, and Linear Algebra. Academic Press (1974)
18. Guckenheimer, J., Holmes, P.: Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields. Springer (1983)
19. Viana, M.: Dynamical systems: moving into the next century. In Engquist, B., Schmid, W., eds.: Mathematics Unlimited - 2001 and Beyond. Springer (2001) 1167–1178
20. Pilyugin, S.Y.: Shadowing in Dynamical Systems. Springer (1999)

21. Grebogi, C., Poon, L., Sauer, T., Yorke, J., Auerbach, D.: Shadowability of chaotic dynamical systems. In: Handbook of Dynamical Systems. Volume 2. Elsevier (2002) 313–344
22. Atkinson, K.E.: An Introduction to Numerical Analysis. 2nd edn. John Wiley & Sons (1989)
23. Graça, D.S., Costa, J.F.: Analog computers and recursive functions over the reals. J. Complexity **19** (2003) 644–664

Infinitary Computability with Infinite Time Turing Machines

Joel David Hamkins*

Mathematics Department,
The College of Staten Island of The City University of New York,
2800 Victory Boulevard,
Staten Island, NY 10314, USA
and
The Graduate Center of The City University of New York,
365 Fifth Avenue,
New York, NY 10016, USA
http://jdh.hamkins.org

Abstract. Recent developments in the theory of infinite time Turing machines include the solution of the infinitary P versus NP problem and the rise of infinitary computable model theory.

Infinite time Turing machines extend the operation of ordinary Turing machines into transfinite ordinal time. By doing so, they provide a theoretical model of infinitary computability, while remaining close in spirit to many of the methods and concepts of classical computability. The model gives rise to a robust theory of infinitary computability on the reals, such as notions of computability for functions $f : \mathbb{R} \to \mathbb{R}$ and notions of decidability for sets $A \subseteq \mathbb{R}$, with a rich degree structure. In this brief article, I would like to announce and explain a few of the most recent developments in the theory of infinite time Turing machines. These developments include the rise of infinitary complexity theory, with a solution of the infinite time Turing machine analogue of the P versus NP question, and the development of infinitary computable model theory. Much of the work on infinite time Turing machines lies within the boundaries between set theory, descriptive set theory, computability theory and computable model theory.

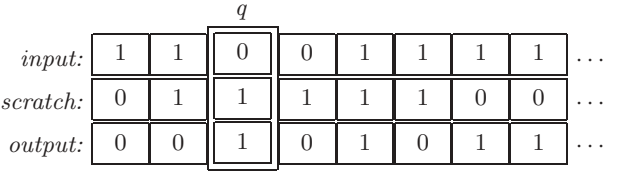
1 Infinite Time Turing Machines

Infinite time Turing machines were first considered by Hamkins and Kidder in 1989, with the main introduction provided by Hamkins and Lewis [1], to which I refer all readers for a full development of the introductory theory. Here, let me review the basic operation of the machines and the key concepts. An infinite time

* The author's research has been supported in part by grants from the Research Foundation of CUNY.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 180–187, 2005.

Turing machine has the same hardware as a classical three-tape Turing machine, with a head reading and writing on a semi-infinite paper tape, moving left and right in accordance with the rigid instructions of a finite program with finitely many states. At successor steps of computation, the machine operates in exactly



the classical manner, following the program instructions. The computation is extended into transfinite ordinal time simply by defining the configuration of the machine at limit ordinal stages. For any limit ordinal ξ, the configuration of the machine at time ξ, by definition, places the machine in the special *limit* state, with the head on the left-most cell, and each cell of the tape displaying the lim sup of the values appearing in that cell before ξ. Thus, if the values in that cell had previously stabilized before ξ, then the limit value agrees with this stabilized value, and otherwise, when the cell has alternated infinitely often between 0 and 1, the limit value is 1 (using lim inf and 0 in this case gives rise to an equivalent theory). Since this completely specifies the configuration of the machine at time ξ, the machine can naturally continue computing to stage $\xi + 1$ and so on according to the program. Output is given only when the machine explicitly attains the *halt* state, and computation ceases when this occurs.

The machines provide a notion of infinitary computability. Specifically, for any program p the corresponding infinite time computable function φ_p is defined by $\varphi_p(x) = y$ when program p on input x gives output y. Since there is plenty of time for computation, the natural context for input and output involves infinite binary sequences, or Cantor space ${}^{\omega}2$, and with the readers' permission I will denote this space by $\mathbb{R}$ and refer to such sequences as reals. Thus, we have a notion of infinite time *computable* functions $f : \mathbb{R} \to \mathbb{R}$. A set $A \subseteq \mathbb{R}$ is infinite time *decidable* if its characteristic function is infinite time computable. The set A is infinite time *semi-decidable* if the constant function $1 \restriction A$ with domain A is infinite time computable. This is equivalent to A being the domain of an infinite time computable function (but not equivalent to A being the range of such a function). The initial results in [1] show that the arithmetic sets are exactly those that are decidable in time uniformly less than ω^2 and the hyperarithmetic sets are those that are decidable in time less than some recursive ordinal. Every Π^1_1 set is decidable, and the class of decidable sets is contained in Δ^1_2.

An easy cofinality argument establishes that every computation either halts or repeats by some countable ordinal stage. An ordinal α is *clockable* if there is a computation $\varphi_p(0)$ halting in exactly α steps (meaning that the α^{th} step moves to the *halt* state). A real x is *writable* if it is the output of a computation $\varphi_p(0)$, and an ordinal is writable if it is coded by such a real. There are of course only countably many clockable and writable ordinals, because there are

only countably many programs. Both the clockable and writable ordinals extend through all the recursive ordinals and far beyond; their supremum is recursively inaccessible and more. While the writable ordinals form an initial segment of the ordinals, there are gaps in the clockable ordinals, intervals of non-clockable ordinals below the supremum of the clockable ordinals. The gap structure itself becomes quite complicated, with limits of gaps sometimes being gaps and so on, and ultimately it exhibits the same complexity as the infinite time version of the halting problem. Nevertheless, [2] established that the supremum of the clockable and writable ordinals is the same. A real x is *eventually writable* if there is a computation $\varphi_p(0)$ for which x appears on the output tape from some point on (even if the computation does not halt), and x is *accidentally writable* if it appears on any of the tapes at any stage during a computation $\varphi_p(0)$. By coding ordinals with reals, we obtain the notions of eventually and accidentally writable ordinals. If λ is the supremum of the clockable or writable ordinals, ζ is the supremum of the eventually writable ordinals and Σ is the supremum of the accidentally writable ordinals, then [1] establishes $\lambda < \zeta < \Sigma$. Welch [3] showed that $L_\lambda \prec_{\Sigma_1} L_\zeta \prec_{\Sigma_2} L_\Sigma$, and furthermore, these ordinals are characterized as the least example of this pattern.

Many of the fundamental constructions of classical finite time computability theory carry over to the infinite time context. For example, one can prove the infinite time analogues of the *smn*-theorem, the Recursion theorem and the undecidability of the infinite time halting problem, by essentially the classical arguments. Some other classical facts, however, do not directly generalize. For example, it is not true in the infinite time context that if the graph of a function f is semi-decidable, then the function is computable. This is a consequence of the following:

Theorem 1 (Lost Melody Theorem). *There is a real c such that $\{c\}$ is infinite time decidable, but c is not writable.*

The real c is like the lost melody that you can recognize yes-or-no when someone sings it to you, but which you cannot sing on your own; it is a real that exhibits sufficient internal structure that $\{c\}$ is decidable, but is too complicated itself to be writable. If $f(x) = c$ is the function with constant value c, then f is not computable because c is not writable, but the graph is decidable, because we can recognize whether a pair has the form (x, c).

The infinite time analogue of the halting problem breaks into lightface and boldface versions, $h = \{\, p \mid \varphi_p(p) \downarrow \}$ and $H = \{\, (p, x) \mid \varphi_p(x) \downarrow \}$, respectively. These are both semi-decidable and not decidable, but in the infintary context, they are not computably equivalent.

When it comes to oracles, one can of course use an individual real as an oracle in exactly the classical manner, by starting the computation with the oracle real written out on an extra tape. But because we have a notion of decidability and undecidability for *sets* of reals, one wants of course to be able to use a set of reals as an oracle. This is done by adding an extra oracle tape, initially filled with 0s, but allowing the machine to write on this tape and make queries of the oracle. Thus, the machine is able to know of any given real that it can produce,

whether that real is in the oracle set or not. The result is a notion of relative computabiliy $\varphi_p^A(x)$, a notion of reduction $A <_\infty B$ and a notion of equivalence $A \equiv_\infty B$, with a rich theory of the infinite time Turing degrees. For any set A, we have the lightface jump $A^\triangledown$ and the boldface jump $A^\blacktriangledown$, corresponding to the two halting problems. One can show $A <_\infty A^\triangledown <_\infty A^\blacktriangledown$, as well as $A^{\triangledown\blacktriangledown} \equiv_\infty A^\blacktriangledown$ and a great number of other interesting interactions. In [4], we settled the infinite time analogue of Post's problem, the question of whether there are intermediate semi-decidable degrees between 0 and the jump $0^\triangledown$. The answer cuts both ways:

Theorem 2. *The infinite time analogue of Post's problem has both affirmative and negative solutions.*

1. *There are no reals z with $0 <_\infty z <_\infty 0^\triangledown$.*
2. *There are sets of reals A with $0 <_\infty A <_\infty 0^\triangledown$. Indeed, there are incomparable semi-decidable sets of reals $A \perp_\infty B$.*

In other work, Welch [5] found minimality in the infinite time Turing degrees. Hamkins and Seabold [6] analyzed one-tape versus multi-tape infinite time Turing machines, and Benedikt Löwe [7] observed the connection between infinite time Turing machines and revision theories of truth.

2 P vs. NP for Infinite Time Turing Machines

Let me now turn to more recent work. Ralf Schindler [8] initiated the study of infinite time complexity theory by solving the infinite time Turing machine analogue of the P versus NP question. To define the class polynomial class P in the infinite time context, Schindler observed simply that all reals have length ω and the polynomial functions of ω are bounded by those of the form ω^n. Thus, a set $A \subseteq \mathbb{R}$ is in P if there is a program p and a natural number n such that p decides A and halts on all inputs in time before ω^n. The set A is in NP if there is a program p and a natural number n such that $x \in A$ if and only if there is y such that p accepts (x, y), and p halts on all inputs in time less than ω^n. Schindler proved $\mathrm{P} \neq \mathrm{NP}$ for infinite time Turing machines in [8], using methods from descriptive set theory to analyze the complexity of the classes P and NP. This work has now been improved in [9] to the following, where the class co-NP consists of the complements of sets in NP.

Theorem 3. $\mathrm{P} \neq \mathrm{NP} \cap \text{co-NP}$ *for infinite time Turing machines.*

Proof. This proof appears in [9]. Since P is contained in NP and closed under complements, it follows that $\mathrm{P} \subseteq \mathrm{NP} \cap \text{co-NP}$. To see that the inclusion is proper, consider the halting problem for computations halting before ω^ω:

$$h_{\omega^\omega} = \{\, p \mid \varphi_p(p) \text{ halts in fewer than } \omega^\omega \text{ steps} \,\}.$$

I claim that $h_{\omega^\omega} \notin P$. If one could decide h_{ω^ω} in time before ω^ω, then one could compute the function $f(p) = 1$, if $p \notin h_{\omega^\omega}$, diverge otherwise, and one could

do so in time before ω^ω for input $p \notin h_{\omega^\omega}$. If this algorithm for computing f is carried out by program q, then $q \notin h_{\omega^\omega}$ if and only if $f(q) \downarrow = 1$, which holds if and only if $\varphi_q(q)$ halts in fewer than ω^ω steps, which holds if and only if $q \in h_{\omega^\omega}$, a contradiction.

Let me now show that $h_{\omega^\omega} \in$ NP. The idea is to verify whether $p \in h_{\omega^\omega}$ by inspecting (a code for) the computation sequence of $\varphi_p(p)$ up to ω^ω. To set this up, fix a recursive relation $\lhd$ on ω having order type ω^ω and a computable method of coding infinite sequences of reals as reals, so that any real z can be interpreted as coding an infinite sequence of reals $\langle z_n \mid n \in \omega \rangle$. Viewing n as representing the ordinal α of its order type in $\lhd$, we may therefore view any real z as an ω^ω-sequence of reals $\langle (z)_\alpha \mid \alpha < \omega^\omega \rangle$. This coding is computable in the sense that given any $n \in \omega$ representing α with respect to $\lhd$, we can uniformly compute any digit of $(z)_\alpha$.

Consider the algorithm accepting (p, z) if in the sense above the real z codes a halting sequence of snapshots $\langle (z)_\alpha \mid \alpha < \omega^\omega \rangle$ of the computation $\varphi_p(p)$. That is, first, each $(z)_\alpha$ codes the complete configuration of an infinite time Turing machine, including the tape contents, the head position, the state and the program; second, the snapshot $(z)_{\alpha+1}$ is computed correctly from the previous snapshot $(z)_\alpha$, taking the convention that the snapshots should simply repeat after a halt; third, the limit snapshots $(z)_\xi$ for limit ordinals ξ are updated correctly from the previous snapshots $(z)_\alpha$ for $\alpha < \xi$; and finally, fourth, one of the snapshots shows the computation to have halted. Since these requirements are merely an arithmetic condition on the code z, they can be checked by an infinite time Turing machine in time uniformly before ω^2. And since $p \in h_{\omega^\omega}$ if and only if the computation sequence for $\varphi_p(p)$ halts before ω^ω, it follows that $p \in h_{\omega^\omega}$ exactly if there is a real z such that (p, z) is accepted by this algorithm. Thus, $h_{\omega^\omega} \in$ NP.

To see that $h_{\omega^\omega} \in$ co-NP, simply change the fourth requirement to check that none of the snapshots show the computation to have halted. This change means that the input (p, z) will be accepted exactly when z codes a sequence of snapshots of the computation $\varphi_p(p)$, exhibiting it not to have halted in ω^ω many steps. Since there is a real z like this if and only if $p \notin h_{\omega^\omega}$, it follows that the complement of h_{ω^ω} is in NP, and so $h_{\omega^\omega} \in$ co-NP. □

Corollary 4. P $\neq$ NP *for infinite time Turing machines.*

The analysis of [9] provides a deeper analysis of the complexity classes, revealing the structural reasons why P $\neq$ NP $\cap$ co-NP must be true. This analysis places the classes P and NP within a larger hierarchy of complexity classes. Specifically, for any ordinal α, we define the class P_α to include all $A \subseteq \mathbb{R}$ such that there is an ordinal $\beta < \alpha$ and a program p such that p decides A and halts on all inputs in time less than β. Similarly, A is in NP_α when there is an ordinal $\beta < \alpha$ and a program p such that $x \in A$ if and only if there is y such that p accepts (x, y) and p halts on all inputs in time less than β. In this terminology, the classes P and NP are simply $\mathrm{P}_{\omega^\omega}$ and $\mathrm{NP}_{\omega^\omega}$, in the lower middle part of the hierarchy. Two of the structural facts we identified in [9] are the following:

Theorem 5. *The classes* NP_α *are identical for* $\omega + 2 \leq \alpha \leq \omega_1^{\mathrm{CK}}$. *Nevertheless,* $\mathrm{P}_{\alpha+1} \subsetneq P_{\alpha+2}$ *for any clockable limit ordinal* α.

Corollary 6. $\mathrm{P}_\alpha \subsetneq \mathrm{NP}_\alpha \cap \text{co-}\mathrm{NP}_\alpha$ *for any ordinal* α *with* $\omega + 2 \leq \alpha < \omega_1^{\mathrm{CK}}$.

Proof. The point is that by Theorem 5 the classes P_α are steadily increasing, while the classes $\mathrm{NP}_\alpha \cap \text{co-}\mathrm{NP}_\alpha$ remain the same. Since P_α is contained within $\mathrm{NP}_\alpha \cap \text{co-}\mathrm{NP}_\alpha$, it follows for α in that range that P_α can never equal $\mathrm{NP}_\alpha \cap \text{co-}\mathrm{NP}_\alpha$. □

Nevertheless, we attain equality at the supremum ω_1^{CK} with

$$\mathrm{P}_{\omega_1^{\mathrm{CK}}} = \mathrm{NP}_{\omega_1^{\mathrm{CK}}} \cap \text{co-}\mathrm{NP}_{\omega_1^{\mathrm{CK}}}.$$

In fact, [9] shows that this is an instance of the equality $\Delta_1^1 = \Sigma_1^1 \cap \Pi_1^1$.

This same pattern of inequality $\mathrm{P}_\alpha \subsetneq \mathrm{NP}_\alpha \cap \text{co-}\mathrm{NP}_\alpha$ is mirrored higher in the hierarchy, whenever α lies strictly within a contiguous block of clockable ordinals, with the corresponding $\mathrm{P}_\beta = \mathrm{NP}_\beta \cap \text{co-}\mathrm{NP}_\beta$ for any β that begins a gap in the clockable ordinals. In addition, the question is settled in [9] for the other complexity classes P^+, P^{++} and P^f.

3 Infinite Time Computable Model Theory

Computable model theory is model theory with a view to the computability of the structures and theories that arise. Infinite time computable model theory carries out this program with the notion of infinite time computability provided by infinite time Turing machines. The classical theory began decades ago with such topics as computable completeness (Does every decidable theory have a decidable model?) and computable categoricity (Does every isomorphic pair of computable models have a computable isomorphism?), and the field has now matured into a sophisticated analysis of the complexity spectrum of countable models and theories.

The motivation for a broader context is that, while classical computable model theory is necessarily limited to countable models and theories, the infinitary computability context allows for *uncountable* models and theories. Many of the computational constructions in computable model theory generalize from structures built on $\mathbb{N}$, using finite time computability, to structures built $\mathbb{R}$, using infinite time computability. The uncountable context opens up new questions, such as the infinitary computable Löwenheim-Skolem Theorem, which have no finite time analogue.

In joint work, Miller, Seabold, Warner and I have observed that the infinite time version of Myhill's theorem, a computable version of the Cantor-Schröder-Bernstein Theorem, holds when the inverses of the computable injections are also computable, but can fail when they are not. The infinite time computable Completeness Theorem holds for countable languages coded in the natural numbers, but can fail for uncountable languages coded in the reals. The computable downward Löwenheim-Skolem theorem fails:

Theorem 7. *There is an infinite time computable structure of size continuum having no proper infinite time computable elementary substructure.*

Some of the most interesting current results involve computable quotients. A structure has an infinite time computable *presentation* if it is isomorphic to a computable structure, and has a computable *quotient presentation* if it is isomorphic to the quotient of a computable structure by a computable equivalence relation (a congruence). For structures on $\mathbb{N}$, in either the finite or infinite time context, these notions are equivalent, because one can computably find the least element of any equivalence class. For structures on $\mathbb{R}$, however, computing such distinguished elements of every equivalence class is not always possible.

Question 8. Does every structure with an infinite time computable quotient presentation have an infinite time computable presentation?

Our answer is that it depends on the set theoretic background.

Theorem 9. *The answer to Question 8 is independent of* ZFC*. Specifically,*

1. *It is relatively consistent with* ZFC *that every structure with an infinite time computable quotient presentation has an infinite time computable presentation.*
2. *It is relatively consistent with* ZFC *that there is a structure having an infinite time computable quotient presentation, but no infinite time computable presentation.*

Let me briefly sketch some of the ideas appearing in the proof. In order to construct an infinite time computable presentation of a structure, given a computable quotient presentation, we'd like somehow to select a representative from each equivalence class, in a computably effective manner, and build a structure on these representatives. Under the set theoretic assumption $V = L$, we can attach to the L-least member of each equivalence class an escort real that is powerful enough to reveal that it is the L-least member of its class, and build a computable presentation out of these escorted pairs of reals. (In particular, the new presentation is not built out of mere representatives from the original class, since these reals may be too weak; they need the help of their escorts.) Thus, if $V = L$, then every structure with a computable quotient presentation has a computable presentation. On the other side of the independence, we prove statement 2 by the method of forcing. The structure $\langle \omega_1, < \rangle$ always has a computable quotient presentation built from reals coding well orders, but there are forcing extensions in which no infinite time computable set has size ω_1, on descriptive set theoretic grounds. In these extensions, therefore, $\langle \omega_1, < \rangle$ has a computable quotient presentation, but no computable presentation.

References

1. Hamkins, J.D., Lewis, A.: Infinite time Turing machines. J. Symbolic Logic **65** (2000) 567–604
2. Welch, P.: The lengths of infinite time Turing machine computations. Bulletin of the London Mathematical Society **32** (2000) 129–136
3. Welch, P.: Eventually infinite time Turing machine degrees: Infinite time decidable reals. Journal of Symbolic Logic **65** (2000) 1193–1203
4. Hamkins, J.D., Lewis, A.: Post's problem for supertasks has both positive and negative solutions. Archive for Mathematical Logic **41** (2002) 507–523
5. Welch, P.: Friedman's trick: Minimality arguments in the infinite time Turing degrees. in "Sets and Proofs", Proceedings ASL Logic Colloquium **258** (1999) 425–436
6. Hamkins, J.D., Seabold, D.: Infinite time Turing machines with only one tape. Mathematical Logic Quarterly **47** (2001) 271–287
7. Löwe, B.: Revision sequences and computers with an infinite amount of time. Logic Comput. **11** (2001) 25–40
8. Schindler, R.D.: P $\neq$ NP for infinite time Turing machines. Monatshefte für Mathematik **139** (2003) 335–340
9. Deolalikar, V., Hamkins, J.D., Schindler, R.D.: $P \neq \text{NP} \cap \text{co-NP}$ for infinite time turing machines. submitted

Combinatorial Models of Gene Assembly

Tero Harju

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

1 Introduction

We consider formal recombination operations presented in terms of string, graphs and permutations. These operations are faithful to the molecular operations of gene assembly introduced by Prescott, Ehrenfeucht, and Rozenberg [17]. The results mentioned here on the formal operations are mostly stated in the recent book [8], where one also finds the original references.

Ciliates are complex unicellular organisms with a unique nuclear dimorphism: they possess two kinds of nuclei (see reviews by Prescott [14, 15]; Jahn and Klobutcher [12]). The active genome (*macronucleus*) is formed from an inactive genome (*micronucleus*). In the process ciliates undergo complex genomic reorganization involving massive parallel DNA elimination, sequence rearrangement, telomere formation, and extensive gene amplification.

The *micronucleus* (MIC) is an inactive storage of genes consisting of long chromosome(s) that code genes in scrambled manner, that is, a gene can occur fragmented in MIC. MIC becomes functional only during sexual reproduction. The *macronucleus* (MAC) contains the active genome. It consists of short, gene size, DNA molecules; see Prescott et al. [16]. The genes in MAC are unfragmented.

A species of ciliates often has many identical MICs and MACs. E.g., in Sterkiella nova the MAC genome has about 1 000 copies of some 20 000 different chromosomes. Each chromosome has usually only one gene. In the formation of MAC from MIC much of the DNA is lost. E.g., in Stylonychia lemnae 98% of the DNA in the MIC genome is eliminated in this process.

In the micronuclear version, a gene can be dispersed in many parts, *macronuclear destined segments* (MDSs). These parts are dispersed by non-coding segments, *internally eliminated segments* (IESs). Not only can the order of the MDSs be changed in the MIC, but also some of the MDSs may have been inverted. During gene assembly the genes are defragmented: the order of the parts (MDSs) and their direction is recovered. This process can be spectacular: e.g., the whole MIC of Sterkiella nova has more than 100 000 IESs and they are all excised and equally many MACs are spliced in their final order to form the functional genes.

In the MIC the MDSs are permuted, possibly inverted, and they are separated by IESs. They assemble either to *linear* or *circular* genes as illustrated in Fig. 1.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 188–195, 2005.

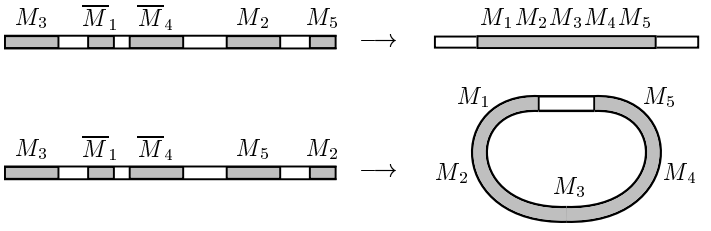


Fig. 1. Two micronuclear genes; MDSs M_1 and M_4 are inverted

At the final stage both of these cases are excised to *linear genes* and telomeres are added at the ends. In this way, the genes are rearranged in the *orthodox order* $M_1 - M_2 - M_3 - M_4 - M_5$.

The MDSs are spliced through 'pointers': the ith MDS (enumerated in the order in MAC) has the form $M_i = (\pi_i, B_i, \pi_{i+1})$, where π_i and π_{i+1} are *pointers* and B_i is the *body*. Thus MDSs in MIC have a linked list structure. The MDSs M_1 and M_n have the forms (b, B_1, π_2) and (π_n, B_n, e), where b and e are special notations for 'beginning' and 'ending'. When the MDSs M_i and M_{i+1} are spliced they form a structure $(\pi_i, B_i\pi_{i+1}B_{i+1}, \pi_{i+2})$, and π_{i+1} ceases to act as a pointer. Note that one copy of π_{i+1} is part of the gene! After all splicings, the MAC gene forms $(b, B_1\pi_2B_2\pi_3 \dots \pi_nB_n, e)$.

2 Models

Signed permutations. Let Σ^* denote the set of all strings over an alphabet Σ, and let $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$ be a disjoint copy of Σ together with $\overline{\overline{a}} = a$. It is convenient also to use notations $a = a^+$ and $\overline{a} = a^-$ for each letter $a \in \Sigma \cup \overline{\Sigma}$. A string $s = a_1^{e_1}a_2^{e_2} \dots a_n^{e_n}$ over $\Sigma \cup \overline{\Sigma}$ is a *signed string* (over Σ). The *inversion* of s is $\overline{s} = a_n^{-e_n}\ a_{n-1}^{-e_{n-1}}\ \dots\ a_1^{-e_1}$. A signed string α is a *signed permutation*, if it has exactly one occurrence from $\{a, \overline{a}\}$ for each $a \in \Sigma$. We say that α is *uniformly signed* if all letters in α have the same sign, that is, $\alpha \in \Sigma^*$ or $\alpha \in \overline{\Sigma}^*$.

Consider a gene that has MDSs $M_1, M_2, \dots, M_n$ in the MIC enumerated in their order in MAC. Such a gene can be represented as a signed string

$$M_{r_1}^{e_1} I_1 M_{r_2}^{e_2} I_2 \dots I_{n-1} M_{r_n}^{e_n},$$

where $M_{r_1}, M_{r_2}, \dots, M_{r_n}$ is a permutation of $M_1, M_2, \dots, M_n$, and each I_i denotes an IES.

The IESs are excised in the process of forming MAC, therefore we can them from the representation of the MIC gene, which then becomes as $M_{r_1}^{e_1} M_{r_2}^{e_2} \dots M_{r_n}^{e_n}$ or simpler still: a signed permutation $r_1^{e_1} r_2^{e_2} \dots r_n^{e_n}$ of $\{1, 2, \dots, n\}$.

The MAC version of the gene (before telomere addition) has presentation $i(i+1)\dots n1\dots(i-1)$ or $\overline{i}(\overline{i-1})\dots\overline{1}\overline{n}\dots(\overline{i+1})$. For $i > 1$, these strings represent *circular molecules*.

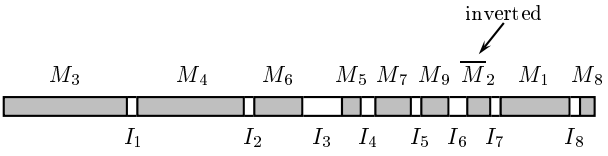


Fig. 2. The MDS/IES structure of actin gene in Sterkiella nova

Example 1. Consider the actin gene in Sterkiella nova, represented in Fig. 2. The micronuclear gene is represented by the string $M_3M_4M_6M_5M_7M_9\overline{M}_2M_1M_8$ which then gives the signed permutation $\alpha = 3\,4\,6\,5\,7\,9\,\overline{2}\,1\,8$. □

Let $\omega = 12\ldots n$ be the *identity* on $[1,n]$. A conjugate $\widetilde{\omega}$ of ω or of the inversion $\overline{\omega}$ is an *assembled permutation*. Hence *for signed permutations, assembling means sorting up to conjugation and inversion.*

A related problem concerns *sorting by inversions*, where the operation is more general: $xyz \;\mapsto\; x\bar{y}z$. Reversals can sort (even to ω) all signed permutations. There is a fast polynomial time algorithm for finding the minimum number of inversions needed to sort a signed permutation; Hannenhalli and Pevzner [11]. Note that the problem of sorting *unsigned* permutations by reversals is NP-hard; Caprara [7].

Legal strings. Let $[k,n]$ denote the interval $\{k,k+1,\ldots,n\}$. A signed string s is *legal*, if every letter x in s has exactly two occurrences from $\{x,\overline{x}\}$. Each signed permutation on $[1,n]$ (MIC gene) can be mapped to a legal string over $[2,n]$ as follows

$$\begin{aligned} p &\;\mapsto\; p(p+1)\,, & \overline{p} &\;\mapsto\; (\overline{p+1})\overline{p}\,,\\ 1 &\;\mapsto\; 2\,, & \overline{1} &\;\mapsto\; \overline{2}\,,\\ n &\;\mapsto\; n\,, & \overline{n} &\;\mapsto\; \overline{n}\,. \end{aligned}$$

A letter $p \in \Sigma$ is *positive* in s, if both p and $\overline{p}$ occur in s; otherwise p is *negative* in s.

Example 2. The MIC version of the actin gene in Sterkiella nova has the presentation $M_3M_4M_6M_5M_7M_9\overline{M}_2M_1M_8$ (see Fig. 2). The corresponding signed permutation gives the legal string: $3\,4\,6\,5\,7\,9\,\overline{2}\,1\,8 \;\mapsto\; 3\,4\,4\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9$. □

Overlap graphs. A *signed graph* γ consits of a set V of vertices, each signed by $+$ or $-$ (positive/negative), and a set $E \subseteq \{\{x,y\} \mid x,y \in V, x \neq y\}$ of edges. (In literature, also known as 'marked graphs'; Zaslavsky [18].)

Let s be a legal string over an alphabet Σ. Then the *overlap graph* γ_s of s is the signed graph with Σ as its vertex set such that a vertex $p \in \Sigma$ has a sign $+$, if p is *positive* in s, Also, $\{p,q\}$ is an edge if the intervals of their occurrences overlap in s: $\ldots p\ldots q\ldots p\ldots q\ldots$ or $\ldots q\ldots p\ldots q\ldots p\ldots$; see Fig. 3.

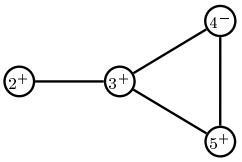


Fig. 3. The overlap graph of $s = 3\,4\,\overline{5}\,2\,\overline{3}\,\overline{2}\,4\,5$

Overlap graphs are also known as *(signed) circle graphs* or *(signed) interleaving graphs*; Bouchet [4, 6], de Fraysseix [10]. The latter are well known from their connection to the Gauss problem on intersecting plane curves. Circle graphs have a geometric presentation as chord diagrams, which are auseful in knot theory and the theory of braids; Birman and Trapp [3].

Let $s = a_1 a_2 \dots a_k$ be a legal string over an alphabet Σ. It is clear that the *reversal* $s^R = a_k a_{k-1} \dots a_1$ as well as the *complement* $s^C = \overline{a}_1 \overline{a}_2 \dots \overline{a}_k$ have the same overlap graph as s, and hence so does the inversion $\overline{s} = (s^R)^C$ of s. Also, all *conjugates* (vu and uv) have the same overlap graph. For two signed strings s and s' over Σ, denote $s \approx s'$, if s is obtained from a conjugate of s' by a composition of the operations of reversal and complementation.

Theorem 3. *Let s and s' be legal strings that are images of signed permutations. Then s and s' have the same overlap graph if and only if $s \approx s'$.*

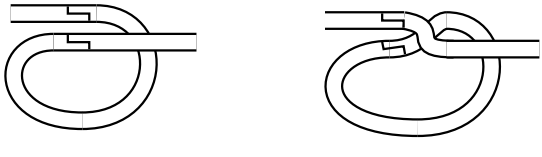
The problem in the general setting is much more difficult.

Problem. Give necessary and sufficient conditions for general legal strings s and s' to have the same overlap graph.

3 The Assembly Rules: ld, hi, dlad

The basic biological question for gene assembly is the following: How is MAC formed from MIC? Kari and Landweber [13] took a computational approach by introducing an intermolecular model on strings, and then Prescott et al. [17] proposed an intramolecular model with three rearrangement operations: ld, hi and dlad. For strings and graphs, these operations are *reductive*: the letters and vertices are removed when an operation is applied to them. For signed permutations, the operations are *nonreductive*, that is, they modify the permutations without removing any parts.

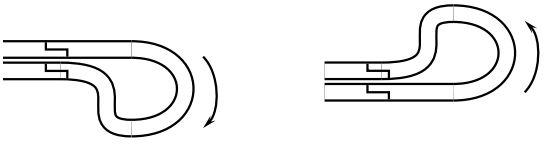
ld-*rule: (loop, direct repeat)-excision.* The molecular operation ld is applied to a molecule with a direct repeat pattern $(- \pi - \pi -)$ of a pointer such that the occurrences of π are separated by one IES only or they are at the two opposite ends of the part of the molecule containing the MDSs of the gene.



The ld-rule yields two molecules: a linear and a circular one; the circular molecule consists either of the whole gene or one IES only.

(1) For legal strings, ld-rule removes a factor pp: for each (signed) p, $uppv \to uv$ or $puvp \to uv$.
(2) For signed graphs, ld-rule removes an isolated negative vertex.
(3) In the model of signed permutations, this operation is not used, since an excision of an IES does not change the order or direction of the MDSs, and hence this operation does not manifest itself in the sorting operations of permutations.

hi-*rule: (hairpin, inverted)-excision.* The operation hi is applied to a molecule with an *inverted repeat pattern* $(-\,\pi\,-\,\overline{\pi}\,-)$ of a pointer. This operation inverts a part of the molecule.



(1) For legal strings, hi-rule is defined as follows: for each p, $upx\overline{p}v \to u\overline{x}v$.
(2) For signed graphs, hi-rule complements the neighbourhood $N(p)$ of a positive vertex p, (and p is removed). The signs in $N(p)$ are changed, and the edges are changed to nonedges and vice versa. This is a generalization of *local complementation* of unsigned graphs; Bouchet [5], Fon-Der-Flaas [9].

If we allow both ld-rules and hi-rules for graphs, then Anderson et al. [1] showed.

Theorem 4. *A signed graph γ can be reduced to the empty graph by* ld*-rules and* hi*-rules if and only if every connected component of γ has a positive vertex.*

(3) For signed permutations, hi_p, with $p \in [1, n]$, is one of the following rules; see Fig. 4 for the first case.

$$\begin{aligned} p\,\delta\,(\overline{p+1}) &\longrightarrow p\,(p+1)\,\overline{\delta}\,, \\ \overline{p}\,\delta\,(p+1) &\longrightarrow \overline{\delta}\,p\,(p+1)\,, \end{aligned}$$

and the dual rules of these obtained by inverting the strings:

$$\begin{aligned} (p+1)\,\delta\,\overline{p} &\longrightarrow \overline{\delta}\,(\overline{p+1})\,\overline{p}\,, \\ (\overline{p+1})\,\delta\,p &\longrightarrow (\overline{p+1})\,\overline{p}\,\overline{\delta}\,. \end{aligned}$$

Example 5. We write $\alpha \xrightarrow{i} \alpha'$, if α' is obtained from α by applying hi_i. We have that $\overline{2}\,3\,1 \xrightarrow{2} 2\,3\,1$ and $\overline{2}\,3\,1 \xrightarrow{1} \overline{2}\,\overline{1}\,\overline{3}$. This case corresponds to a cyclic molecule. □

Note that if hi_p is applicable to a signed permutation α, then p and $p+1$ have opposite signs in α and they have the same sign in the resulting signed permutation $\mathsf{hi}_p(\alpha)$.

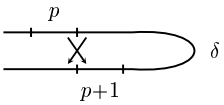
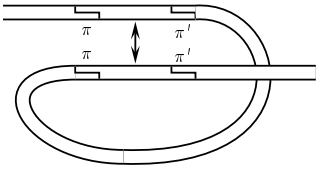


Fig. 4. The case of hi for the order $p - \overline{p+1}$

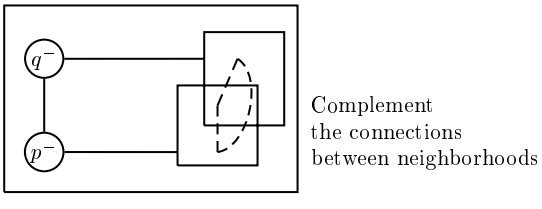
Lemma 6. *Let α be a signed permutation on $[1, n]$. Then there is a composition $\varphi = \mathsf{hi}_{p_k} \mathsf{hi}_{p_{k-1}} \ldots \mathsf{hi}_{p_1}$ for different $p_1, \ldots, p_k$ such that $\varphi(\alpha)$ is uniformly signed.*

dlad*: (double loop, alternating direct repeat)-excision.* The operation dlad is applied to a molecule with an *alternating direct repeat pattern* $(- \pi - \pi' - \pi - \pi' -)$ for pointers π, π'. The operation transposes the substrings between π and π':



(1) For legal strings, dlad-rule has the following effect: for each p and q, $upxqwpyqv \rightarrow uywxv$.

(2) For signed graphs, dlad-rule complements the connections between $N(p)$ and $N(q)$ of adjacent negative p and q (which are removed). This operation corresponds to the *pivot operation* for general graphs; see [2].



(3) For signed permutations, we have first the following '4-rules' $\mathsf{dlad}_{p,q}$, with $p, q \in [1, n]$.

$$p\,\delta_1\,q\,\delta_2\,(p+1)\,\delta_3\,(q+1) \longrightarrow\ p\,(p+1)\,\delta_3\,\delta_2\,\delta_1\,q\,(q+1)$$

and the conjugate rules:

$$\begin{aligned}
p\,\delta_1\,(q+1)\,\delta_2\,(p+1)\,\delta_3\,q &\longrightarrow\ p\,(p+1)\,\delta_3\,q\,(q+1)\,\delta_2\,\delta_1\\
(q+1)\,\delta_1\,(p+1)\,\delta_2\,q\,\delta_3\,p &\longrightarrow\ \delta_3\,p\,(p+1)\,\delta_2\,q\,(q+1)\,\delta_1\\
(p+1)\,\delta_1\,q\,\delta_2\,p\,\delta_3\,(q+1) &\longrightarrow\ \delta_3\,\delta_2\,p\,(p+1)\,\delta_1\,q\,(q+1)
\end{aligned}$$

In these it can be $q = p + 1$:

$$(p+2)\,\delta_1\,(p+1)\,\delta_2\,p \longrightarrow\ \delta_2\,p\,(p+1)\,(p+2)\,\delta_1$$

and the conjugate rules:

$$p\,\delta_1\,(p+2)\,\delta_2\,(p+1) \longrightarrow\ p\,(p+1)\,(p+2)\,\delta_2\,\delta_1$$
$$(p+1)\,\delta_1\,p\,\delta_2\,(p+2) \longrightarrow\ \delta_2\,\delta_1\,p\,(p+1)\,(p+2)$$

Also, we have the duals rules obtained by inverting both sides of the rules.

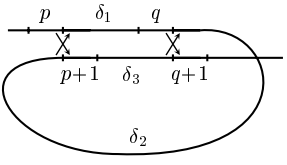


Fig. 5. The case of dlad for the order $p - q - (p+1) - (q+1)$

Theorem 7. *Each unsigned permutation can be sorted to a conjugate of ω by* dlad*-rules.*

Universality. The operations for strings, graphs and permutations have (almost) unique correspondence to each other with respect to sequences of their compositions; [8].

A composition φ of operations hi and dlad (and ld in the case of strings and graphs) is said to be *successful* for a signed permutation α, if $\varphi(\alpha)$ is assembled (a conjugate of ω or $\overline{\omega}$). That is, if it sorts α up to taking inversion and conjugates.

Theorem 8. *Every signed permutation α has a successful composition.*

Using hi-rules every legal string s can be reduced to a string with only negative letters, and by the dlad-rules, we terminate in a string with no overlapping letters. Hence

Theorem 9. *For each legal string s, there is a composition $\varphi = \varphi_3\varphi_2\varphi_1$ such that $\varphi(s)$ is the empty string and φ_1 consists of* hi*-rules, φ_2 of* dlad*-rules, and φ_3 of* ld*-rules.*

Similarly, we have

Theorem 10. *For each signed graph γ, there is a composition $\varphi = \varphi_3\varphi_2\varphi_1$ such that $\varphi(\gamma)$ is the empty graph and φ_1 consists of* hi*-rules, φ_2 of* dlad*-rules, and φ_3 of* ld*-rules.*

It should be noted that a signed permutation can have many successful compositions, but all these give basically the same result. In particular, circularity is invariant, that is, if one composition assembles a signed permutation α to the identity permutation ω, then so do all successful compositions. This result is stated in the following theorem; see [8].

Theorem 11. *Let φ and φ' be two successful compositions of a signed permutation α. If $\varphi(\alpha) = \omega$, then also $\varphi'(\alpha) = \omega$.*

References

1. Anderson, E., Chrobak, M., Noga, J., Sgall, J., and Woeginger, G., Solution of a problem in DNA computing. *Theoret. Comput. Sci.* **287** (2002), 387 – 391.
2. Arratia, R., Bollobáas, B., and Sorkin, G. B., The interlace polynomial: a new graph polynomial. *Proc. Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM, New York, 2000, pp. 237 – 245.
3. Birman, J. S. and Trapp, R., Braided chord diagrams. *J. Knot Theory and its Ramifications* **7** (1998), 1 – 22.
4. Bouchet, A., Circle graphs. *Combinatorica* **7** (1987), 243 – 254.
5. Bouchet, A., Transforming trees by successive local complementations. *J. Graph Theory* **12** (1988), 195 – 207.
6. Bouchet, A., Circle graph obstructions. *J. Combin. Theory Ser B* **60** (1994), 107 – 144.
7. Caprara, A., Sorting by reversals is difficult. *1st Annual Int. Conf. on Comput. Molecular Biology*, (S. Istrail, P. Pevzner, and M. Waterman, eds.), 1997, 75 – 83.
8. Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G., *Computation in Living Cells. Gene Assembly in Ciliates*, Springer-Verlag, 2003.
9. D. Fon-der-Flaas, On local complementations of graphs. In: Combinatorics (Eger, 1987), Colloq. Math. Soc. János Bolyai **52**, North-Holland, Amsterdam, 1988, pp. 257 – 266.
10. de Fraysseix, H., A characterization of circle graphs. *European J. Combin.* **5** (1984), 223 – 238.
11. Hannenhalli, S. and Pevzner, P. A., Transforming cabbage into turnip, *Proc. 27th ACM Symp. on Theory of Computing* (1995), pp. 178 – 189.
12. Jahn, C. L. and Klobutcher, L. A., Genome remodeling in ciliated protozoa. *Ann. Rev. Microbiol.* **56** (2000), 489 – 520.
13. Landweber, L. F. and Kari, L., The evolution of cellular computing: nature's solution to a computational problem. *Proc. 4th DIMACS meeting on DNA based computers*, Philadelphia, PA, pp. 3 – 15 (1998).
14. Prescott, D. M., The evolutionary scrambling and developmental unscambling of germline genes in hypotrichous ciliates. *Nucleic Acids Res.* **27**(5) (1999), 1243 – 1250.
15. Prescott, D. M., Genome gymnastics: unique modes of DNA evolution and processing in ciliates. *Nat Rev Genet.* **1**(3) (2000), 191 – 198.
16. Prescott, D. M., Bostock, C. J., Murti, K. G., Lauth, M. R., Gamow, E., DNA in ciliated protozoa. I. Electron microscopy and sedimentation analysis of macronuclear and micronuclear DNA of *Stylonychia mytilus*. *Chromosoma* **34** (1971), 355 – 366.
17. Prescott, D. M., Ehrenfeucht, A., and Rozenberg, G., Molecular operations for DNA processing in hypotrichous ciliates. *European J Protistology* **37** (2001), 241 – 260.
18. Zaslavsky, T., Bibliography of Signed and Gain Graphs. Electronic J. Combin., Dynamic Surveys, 1999.
`http://www.combinatorics.org/Surveys/`

Symmetric Enumeration Reducibility

Charles M. Harris

Department of Pure Mathematics,
University of Leeds,
Leeds LS2 9JT, England

Abstract. Symmetric Enumeration reducibility ($\leq_{\mathrm{se}}$) is a subrelation of Enumeration reducibility ($\leq_{\mathrm{e}}$) in which both the positive and negative information content of sets is compared. In contrast with Turing reducibility ($\leq_{\mathrm{T}}$) however, the positive and negative parts of this relation are separate. A basic classification of $\leq_{\mathrm{se}}$ in terms of standard reducibilities is carried out and it is shown that the natural embedding of the Turing degrees into the Enumeration degrees easily translates to this context. A generalisation of the relativised Arithmetical Hierarchy is achieved by replacing the relation *c.e. in* by $\leq_{\mathrm{e}}$ and $\leq_{\mathrm{T}}$ by $\leq_{\mathrm{se}}$ in the underlying framework of the latter.

1 Introduction

The structure of the relativised Arithmetical Hierarchy is dictated by the relation *computably enumerable in* (*c.e. in*) in the sense that a set belongs to level $n+1$ of the hierarchy if either it or its complement is *c.e. in* some set belonging to level n. This of course means that each level is the union of a positive (Σ) class and a negative (Π) class. The intersection (Δ) of these classes comprises, by definition, those sets B such that B is *Turing reducible* to (i.e. $\overline{B}$ and B are both *c.e. in*) a set in the antecedent level.

A natural question to ask is whether one can refine the framework of the relativised Arithmetical Hierarchy to the context of Enumeration Reducibility. In other words, can one define a similar hierarchy in which the levels are dictated by the relation *enumeration reducible to* ($\leq_{\mathrm{e}}$) and which is identical to the Arithmetical Hierarchy when relativised to graphs of characteristic functions (i.e. Turing embedded sets, see Section 4). One approach to this question, is to find an appropriate relation to define the intersection (Δ) levels of the structure (in the place of $\leq_{\mathrm{T}}$). This is the initial motivation behind the introduction of *Symmetric Enumeration Reducibility* since the latter allows us to define a hierarchy which is a generalisation of the relativised Arithmetical Hierarchy in the required sense.

2 Preliminaries

Background Notation. We let $\mathbb{N}$ denote the set of natural numbers and $A, B, \ldots$ denote subsets of $\mathbb{N}$. Lower case letters $n, x, \ldots$ and $f, g, \ldots$ represent

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 196–208, 2005.

numbers and functions (from $\mathbb{N}$ to $\mathbb{N}$) respectively, whereas $\mathbf{A}, \mathbf{B}, \ldots$ represent classes of sets. $\overline{A}$ denotes the complement of A. The set $\{ n \cdot x + m \mid x \in A \}$ is written $nA + m$ and $2A \cup 2B{+}1$ is written $A \oplus B$. We use $\langle\ ,\ \rangle$ to denote the standard diagonal coding function defined by $\langle x, y\rangle = 1/2(x^2{+}y^2{+}2xy{+}3x{+}y)$. The characteristic function of A is written c_A, and for any function f, its graph is written $\mathbb{F}$ (and so $\mathbb{C}_A$ stands for the graph of c_A). We assume the availability of effective enumerations of (oracle) Turing machines $\varphi_0, \varphi_1, \ldots$, computably enumerable (c.e) sets $W_0, W_1, \ldots$, and finite sets $D_0, D_1, \ldots$ Note that, to simplify notation, we usually use D, D' etc. to denote both the finite sets themselves and their indices. For example if i, j are the indices of D, D' then $\langle D, D'\rangle$ is shorthand for $\langle i, j\rangle$.

Basic Reducibilities. We assume the standard multitape Turing machine model for computing partial functions and we suppose an oracle Turing machine to be equipped with a function oracle. We say that the set A is *Turing reducible* to the set B ($A \leq_{\mathrm{T}} B$) if there is an oracle machine φ that computes c_A when equipped with oracle c_B (written $c_A \simeq \varphi^B$). A is said to be *computably enumerable in* B (A *c.e. in* B) if, A is the range of some function f computable in B or, equivalently, if $A = \{ x \mid \varphi^B(x) \downarrow \}$ for some oracle Turing machine φ. $\mathcal{K}_B$ denotes the set $\{ x \mid \varphi^B_x(x) \downarrow \}$ and $\mathcal{K}^{(n)}_B$ denotes the iterated form of $\mathcal{K}_B$: $\mathcal{K}^{(0)}_B = B$ and $\mathcal{K}^{(n+1)}_B = \mathcal{K}_{\mathcal{K}^{(n)}_B}$. $\mathcal{K}$ and $\mathcal{K}^{(n)}$ are shorthand for $\mathcal{K}_\emptyset$ and $\mathcal{K}^{(n)}_\emptyset$. For Turing reductions we use $Q(\varphi, x, B)$ to denote the set of oracle queries made by φ^B on input x. Likewise we use $Q^+(\varphi, x, B)$ to denote the set of positive queries of the computation and $Q^-(\varphi, x, B)$ to denote the set of negative queries. Note that, for fixed x (and φ) these sets are c.e. in B and, if $\varphi^B(x) \downarrow$ they are computable in B. We say that A is *many one reducible* to B ($A \leq_{\mathrm{m}} B$) if there is a computable function f such that $A = f^{-1}(B)$. Furthermore, if f is one-one, A is said to be *one-one reducible* to B ($A \leq_1 B$). We say that A is *enumeration reducible* to B ($A \leq_{\mathrm{e}} B$) if there exists a computably enumerable set W such that, for all x,

$$x \in A \quad \text{iff} \quad \exists D(\, \langle x, D\rangle \in W \ \&\ D \subseteq B\,)$$

and in this case we also say that $A \leq_{\mathrm{e}} B$ *via* W. Similarly, the *e-th enumeration operator* Φ_e is defined such that, for any set A,

$$\Phi_e(B) \ = \ \{ x \mid \exists D(\, \langle x, D\rangle \in W_e \ \&\ D \subseteq B\,) \} \ .$$

A is said to be *positive reducible* to B, ($A \leq_p B$) if there exists a computable function f such that, for all $x \geq 0$, $x \in A \Leftrightarrow \exists y(y \in D_{f(x)} \ \&\ D_y \subseteq B)$. We say that A is *wtt-reducible* to B ($A \leq_{wtt} B$), if there exists a Turing machine φ and computable function f such that $c_A \simeq \varphi^B$ and such that for all $x \geq 0$, $Q(\varphi_e, x, B) \subseteq \{0, \ldots, f(x)\}$.

$deg_{\mathrm{r}}(A)$ denotes the *degree* of A under the reducibility $\leq_{\mathrm{r}}$, i.e. the class $\{ B \mid B \equiv_{\mathrm{r}} A \}$. We use $\boldsymbol{a}_{\mathrm{r}}, \boldsymbol{b}_{\mathrm{r}}, \ldots$ to denote the degrees derived according to this definition and $\langle \mathcal{D}_{\mathrm{r}}, \leq\rangle$ to denote the corresponding *degree structure* (with $\leq$ the ordering induced by $\leq_{\mathrm{r}}$). Subscripts are dropped if the context is clear. A is said to be *r-hard* for a class $\mathbf{C}$ if $X \leq_{\mathrm{r}} A$ for all X in $\mathbf{C}$ and A is said to

be *r-complete* for **C** if A also belongs to **C**. We use the shorthand **Comp**(A), **Enum**(A) and **Ce**(A) to denote the classes $\{ E \mid E \mathcal{R} A \}$ such that (respectively) $\mathcal{R}$ is $\leq_{\mathrm{T}}$, $\leq_{\mathrm{e}}$ or "*c.e. in*". Accordingly, we use **Comp** and **Ce** to denote the classes of computable and c.e. sets. Also we will employ the abbreviations r-reduction, r-degree etc. when appropriate.

String Notation. A string is a partial function $\sigma : \mathbb{N} \to \{0,1\}$ with finite domain. λ denotes the empty string and $|\sigma|$ the length of σ (i.e. the cardinality of its domain). For $(s,i) \in \{(+,1),(-,0)\}$, we use σ^s to denote the set $\{ n \mid \sigma(n)\downarrow = i \}$ and $(\sigma\restriction A)^s$ to denote the set $\{ n \mid n \in A \ \& \ \sigma(n)\downarrow = i \}$ (and so $\sigma^s = (\sigma\restriction\mathbb{N})^s$ for $s \in \{+,-\}$). If the domain of σ is an initial segment of $\mathbb{N}$, σ is said to be an *initial segment*. Note that this means that, if $|\sigma| = n+1$ the domain of σ is $\{0,\ldots,n\}$. We use the shorthand $\sigma' = \sigma^\frown(i)$ to denote the extension of σ of length $|\sigma|+1$, such that $\sigma'(|\sigma|) = i$.

3 Introduction to Symmetric Enumeration Reducibility

Enumeration reducibility compares the positive information content of two sets. Symmetric enumeration reducibility, as we will see, compares both positive and negative information content. We will now introduce this reducibility and consider how it relates to other standard reducibilities. Firstly, however we draw the reader's attention to the fact that Alan Selman defined this reducibility in Section 4 of [5] and exhibited some of its basic properties. In particular Selman noted the inclusion $\leq_{\mathrm{m}} \subseteq \leq_{\mathrm{se}} \subseteq \leq_{\mathrm{T}}$ and proved Theorem 6 (below) relative to the pair $(\leq_{\mathrm{tt}}, \leq_{\mathrm{se}})$.

Definition 1. *For any sets A and B, A is defined to be* symmetric enumeration reducible *to B ($A \leq_{\mathrm{se}} B$) if $A \leq_{\mathrm{e}} B$ and $\overline{A} \leq_{\mathrm{e}} \overline{B}$.*

Notation. For any set A, **s-Enum**(A) denotes the class $\{ E \mid E \leq_{\mathrm{se}} A \}$.

Note 2. Clearly $\leq_{\mathrm{se}}$ inherits reflexivity and transitivity from $\leq_{\mathrm{e}}$. It thus gives rise to a degree structure $(\langle \mathcal{D}_{\mathrm{se}}, \leq \rangle)$. The least upper bound of any two degrees $\boldsymbol{a}_{\mathrm{se}}, \boldsymbol{b}_{\mathrm{se}}$ (written $\boldsymbol{a}_{\mathrm{se}} \cup \boldsymbol{b}_{\mathrm{se}}$) always exists: it is the degree of $A \oplus B$ for any $A \in \boldsymbol{a}_{\mathrm{se}}$ and $B \in \boldsymbol{b}_{\mathrm{se}}$. Therefore $\langle \mathcal{D}_{\mathrm{se}}, \leq \rangle$ is an upper semi lattice. The zero element ($\mathbf{0}_{\mathrm{se}}$) of $\langle \mathcal{D}_{\mathrm{se}}, \leq \rangle$ is **Comp**. Each of these properties is easily checked.

Lemma 3. *For any sets A and B, if $A \leq_{\mathrm{se}} B$ then $A \leq_{\mathrm{e}} B$ and $A \leq_{\mathrm{T}} B$. In other words,*

$$\leq_{\mathrm{se}} \quad \subseteq \quad \leq_{\mathrm{e}} \bigcap \leq_{\mathrm{T}} .$$

Moreover, this inclusion is proper.

Proof. Since $\leq_{\mathrm{se}}$ is a subrelation of $\leq_{\mathrm{e}}$ by definition, in order to prove the inclusion it suffices to note that, for any sets A and B, $\overline{A} \leq_{\mathrm{e}} \overline{B}$ implies that $\overline{A}$ *c.e. in* B. Also, $\mathbb{C}_{\mathcal{K}} \leq_{\mathrm{r}} \overline{\mathcal{K}}$ for $\mathrm{r} \in \{\mathrm{e},\mathrm{T}\}$ whereas $\mathbb{C}_{\mathcal{K}} \not\leq_{\mathrm{se}} \overline{\mathcal{K}}$ (since this would imply $\overline{\mathcal{K}} \leq_{\mathrm{e}} \mathcal{K}$). Thus the inclusion is proper. □

Theorem 4. $\leq_p \subseteq \leq_{\mathrm{se}}$.

Proof. Let A and B be any sets such that $A \leq_p B$ and suppose that f is a computable function that witnesses this reduction in the sense that, for all $x \geq 0$, $x \in A \Leftrightarrow \exists y(y \in D_{f(x)} \ \&\ D_y \subseteq B)$. Now define the two c.e. (in fact computable) sets W and $\widehat{W}$ as follows:

$$\begin{aligned} W &= \{\, \langle x, y\rangle \mid y \in D_{f(x)} \,\} \\ \widehat{W} &= \{\, \langle x, y\rangle \mid D_{f(x)} = \emptyset \ \&\ y = f(x) \,\} \\ &\quad \bigcup \{\, \langle x, y\rangle \mid D_{f(x)} \neq \emptyset \ \&\ \forall z(z \in D_{f(x)} \Rightarrow D_z \cap D_y \neq \emptyset) \,\} \ . \end{aligned}$$

Then $A \leq_{\mathrm{e}} B$ via W and $\overline{A} \leq_{\mathrm{e}} \overline{B}$ via $\widehat{W}$. □

Note 5. Theorem 4 implies that all conjunctive and disjunctive subreducibilities of $\leq_{\mathrm{T}}$ are contained in $\leq_{\mathrm{se}}$ and, in particular, that $\leq_1 \subseteq \leq_{\mathrm{m}} \subseteq \leq_{\mathrm{se}}$.

Theorem 6. *It is neither the case that* $\leq_{wtt} \subseteq \leq_{\mathrm{se}}$ *nor the case that* $\leq_{\mathrm{se}} \subseteq \leq_{wtt}$.

Proof. The first inequality is witnessed by $\mathcal{K}$ in that $\overline{\mathcal{K}} \leq_{wtt} \mathcal{K}$ (and in fact $\overline{\mathcal{K}} \leq_{\mathrm{btt}(1)} \mathcal{K}$) whereas $\overline{\mathcal{K}} \not\leq_{\mathrm{e}} \mathcal{K}$. The second inequality is proved by a straightforward diagonalisation construction that we sketch below. Note that we assume a fixed (effective) enumeration of partial computable functions $f_0, f_1 \ldots$ in addition to the enumeration of oracle Turing machines $\varphi_0, \varphi_1, \ldots$ stipulated by Section 2. We construct sets A and B by finite initial segments $\{\alpha_n\}_{n\geq 0}$ and $\{\beta_n\}_{n\geq 0}$, such that $A = \bigcup \{\, \alpha_n^+ \mid n \geq 0 \,\}$ and $B = \bigcup \{\, \beta_n^+ \mid n \geq 0 \,\}$. In order that $A \leq_{\mathrm{se}} B$ the construction ensures that, for all $x \geq 0$,

$$\begin{aligned} x \in A &\quad \text{iff} \quad \exists y(\, \langle 2x, y\rangle \in B \,) \\ x \in \overline{A} &\quad \text{iff} \quad \exists y(\, \langle 2x{+}1, y\rangle \in \overline{B} \,) \ . \end{aligned}$$

To enable this to happen, we impose the constraint that, for all $s \geq 0$,

$$\beta_{s+1}^+ - \beta_s^+ \quad = \quad \{\ \langle 2x+1, y\rangle \ \mid\ |\beta_s| \leq \langle 2x+1, y\rangle < |\beta_{s+1}| \ \} \qquad (A\text{-}coding)$$

except for the number $|\beta_{s+1}| - 1$ (defined to be "n_s" below) which is used for the diagonalisation at stage $(s+1)$.

The construction. At each stage $(s+1)$ we diagonalise against oracle Turing machine φ_e and partial computable function f_d, where $s = \langle e, d\rangle$. We define

$$\widehat{B}_s \quad := \quad \beta_s^+ \ \bigcup \ \{\, \langle 2x{+}1, y\rangle \mid \langle 2x{+}1, y\rangle \geq |\beta_s| \,\}$$

and we define m_s to be $max\{|\beta_s|, f_d(s){+}1\}$ if $f_d(s)\!\downarrow$ or to be $|\beta_s|$ otherwise. If either $f_d(s)\!\uparrow$ or it is not the case that $\varphi_e^{\widehat{B}_s}\!\downarrow = 1$, then we insert s into A, we set $n_s := \langle 2s, m\rangle$ with m the least number such that $\langle 2s, m\rangle \geq m_s$, and we insert n_s into B. Otherwise—in the case that both computations converge and

$\varphi_e(s) = 1$—we put s into $\overline{A}$, we set $n_s := \langle 2s+1, m' \rangle$ with m' the least number such that $\langle 2s+1, m' \rangle \geq m_s$, and we insert n_s into $\overline{B}$. α_{s+1} and β_{s+1} are defined to be the resulting extensions of α_s and β_s of length $s{+}1$ and $n_s{+}1$ respectively, and such that β_{s+1} conforms to the constraint (*A-coding*) above.

Note that, at the end of the construction, if $s \in A$ then there is a unique number $\langle 2s, m \rangle$ in B and if $s \notin A$ then there is no such number in B. Likewise if $s \in \overline{A}$ then there is a unique number $\langle 2s+1, m' \rangle$ in $\overline{B}$ and if $s \notin \overline{A}$ then there is no such number in $\overline{B}$. Also, the choice of n_s and m_s at each stage $(s+1)$ ensures that the diagonalisation at this stage is preserved for the resulting sets A and B. □

4 Embedding the Turing Degrees

The isomorphic embedding ι_{e} of the Turing degrees $(\langle \mathcal{D}_{\mathrm{T}}, \leq \rangle)$ into the Enumeration degrees $(\langle \mathcal{D}_{\mathrm{e}}, \leq \rangle)$ induced by the map $X \mapsto \mathbb{C}_X$ is essentially an embedding into $\langle \mathcal{D}_{\mathrm{se}}, \leq \rangle$. Moreover the range of this embedding contains *gaps* similar to those appearing in the range of ι_e. These results are presented below. We begin with an easy but useful Lemma.

Lemma 7. *For any set A the following equivalences hold:*
(a) $\mathbb{C}_A \equiv_{\mathrm{se}} A \oplus \overline{A}$ **(b)** $\mathbb{C}_A \equiv_{\mathrm{se}} \overline{\mathbb{C}_A}$ **(c)** $\mathbb{C}_A \equiv_{\mathrm{se}} \mathbb{C}_{\overline{A}}$.

Notation. We say that a set A is *characteristic* if $A = B \oplus \overline{B}$ for some set B. For the sake of simplicity, and in view of Lemma 7, we sometimes prefer to work with a *characteristic* set $(X \oplus \overline{X})$ rather than with the corresponding *characteristic function graph* $(\mathbb{C}_X)$.

Definition 8. *An e-degree is said to be* total *if it contains the graph of a total (or, equivalently, characteristic) function . A se-degree is said to be* characteristic *if it contains the graph of a characteristic function (or, equivalently, a characteristic set).*

Proposition 9. *For any se-degree $\boldsymbol{a}$ the following are equivalent:*
(a) *$\boldsymbol{a}$ is characteristic.*
(b) *For all A in $\boldsymbol{a}$, $A \equiv_{\mathrm{se}} \overline{A}$.*

Proof. Apply Lemma 7 and use the transitivity of $\leq_{\mathrm{se}}$. □

Note 10. $\mathbf{0}_{\mathrm{se}}$ is characteristic.

Lemma 11. *Every total e-degree contains exactly one characteristic se-degree.*

Proof. Suppose that $B, C \in \boldsymbol{a}_{\mathrm{e}}$ and that $B \equiv_{\mathrm{se}} \overline{B}$ and $C \equiv_{\mathrm{se}} \overline{C}$. This means that $\mathbb{C}_B \equiv_{\mathrm{e}} \mathbb{C}_C$, and by applying Lemma 7, it follows that $\mathbb{C}_B \equiv_{\mathrm{se}} \mathbb{C}_C$. Hence $B \equiv_{\mathrm{se}} C$. □

Lemma 12. *For any sets A and B, A c.e. in B iff $A \leq_{\mathrm{e}} \mathbb{C}_B$.*

Proof. Suppose that A and B are sets such that $A = W_i^B$ for some $i \geq 0$. Then

$$W_{e_i} := \{ \langle x, D \oplus D' \rangle \mid \varphi_i^D(x) \downarrow \ \& \ Q^+(\varphi_i, x, D) = D \ \& \ Q^-(\varphi_i, x, D) = D' \}$$

is a c.e. set and $A \leq_{\mathrm{e}} B \oplus \overline{B}$ via W_{e_i}. The opposite implication is obvious. □

Lemma 13. *For any sets A and B,*

$$A \leq_{\mathrm{T}} B \quad \textit{iff} \quad A \leq_{\mathrm{se}} \mathbb{C}_B \quad \textit{iff} \quad \mathbb{C}_A \leq_{\mathrm{se}} \mathbb{C}_B \ .$$

Proof. Apply Lemma 12 in conjunction with Lemma 7.

Corollary 14. *The embedding ι_{se} of the Turing degrees into the se-degrees induced by the map $X \mapsto \mathbb{C}_X$ is structure preserving (i.e. isomorphic).*

Definition 15. *An se-degree $\boldsymbol{a}$ is said to be* quasi-minimal *if $\boldsymbol{a} > \boldsymbol{0}$ and $\forall \boldsymbol{d}(\, \boldsymbol{d} < \boldsymbol{a}$ & $\boldsymbol{d}$ characteristic $\Rightarrow \boldsymbol{d} = \boldsymbol{0}\,)$.*

Theorem 16. *For any se-degree $\boldsymbol{b}$ there exists a degree $\boldsymbol{a}$ such that $\boldsymbol{b} < \boldsymbol{a}$ and such that, for any characteristic degree $\boldsymbol{c}$, if $\boldsymbol{c} \leq \boldsymbol{a}$ then $\boldsymbol{c} \leq \boldsymbol{b}$.*

Proof. The proof is a straightforward modification of the corresponding result relative to $\langle \mathcal{D}_{\mathrm{e}}, \leq \rangle$ due to Medvedev ([4]). Indeed, suppose that B is any set. Then it suffices to construct a set A such that $B \leq_{\mathrm{se}} A$ and such that A satisfies the following requirements:

$$\begin{aligned} R_{3e} \quad &: \quad A \neq \Phi_e(B) \\ R_{3e+1} \ &: \ \Phi_e(A) \text{ characteristic } \Rightarrow \ \Phi_e(A) \leq_{\mathrm{e}} B \\ R_{3e+2} \ &: \ \Phi_e(\overline{A}) \text{ characteristic } \Rightarrow \ \Phi_e(\overline{A}) \leq_{\mathrm{e}} \overline{B} \end{aligned}$$

We ensure that $B \leq_{\mathrm{se}} A$ by encoding B into A in the following manner:

$$\forall x(\, x \in B \quad \text{iff} \quad 2x \in A\,) \qquad (B\textit{-coding})$$

Notation. We say that an *initial segment* σ is *B-compatible* if, for all x such that $2x < |\sigma|$, $x \in B$ iff $2x \in \sigma^+$.

The construction. The set A is constructed by finite initial segments $\{\sigma_n\}_{n \geq 0}$, such that $A = \bigcup \{\, \sigma_n^+ \mid n \geq 0 \,\}$.
Stage (0) $\sigma_0 = \lambda$
Stage (s+1) σ_s has already been defined.
• If $s = 3e$ then, letting $n_s := |\sigma_s|$, we satisfy R_{3e} by defining

$$\sigma_{s+1} \quad := \quad \begin{cases} \sigma_s{}^\frown(\, 1 - \Phi_e(A)(n_s)\,) & \text{if } n_s \text{ is odd} \\ \sigma_s{}^\frown(\, B(n_s/2)\,)^\frown(\, 1 - \Phi_e(A)(n_s{+}1)\,) & \text{if } n_s \text{ is even.} \end{cases}$$

- If $s = 3e+1$ then we try to satisfy R_{3e+1} vacuously by searching for a B-compatible initial segment $\sigma \supseteq \sigma_s$ such that, for some n: $2n, 2n{+}1 \in \Phi_e(\sigma^+)$. If this search is successful, choose the least such σ and set $\sigma_{s+1} := \sigma$. Otherwise set $\sigma_{s+1} := \sigma_s$.
- If $s = 3e+2$ then we try to satisfy R_{3e+2} vacuously by searching for a B-compatible initial segment $\sigma \supseteq \sigma_s$ such that, for some n: $2n, 2n{+}1 \in \Phi_e(\sigma^-)$. If this search is successful, choose the least such σ and set $\sigma_{s+1} := \sigma$. Otherwise set $\sigma_{s+1} := \sigma_s$.

Analysis of the construction. The construction obviously ensures that the constraint (*B-coding*) holds, which means that $B \leq_{\rm se} A$. Also the requirements $\{R_{3e}\}_{e\geq 0}$ prevent $A \leq_{\rm se} B$ and hence $B <_{\rm se} A$. So suppose that there exists a set E such that $E \oplus \overline{E} \leq_{\rm se} A$. Thus by definition, $\Phi_i(A) = E \oplus \overline{E}$ and $\Phi_j(\overline{A}) = \overline{E} \oplus E$ for some $i, j \geq 0$. Now set $s := 3i{+}1$ and $t := 3j + 2$, and define

$$\begin{aligned} P_s &:= \{\, n \mid (\exists \sigma \supseteq \sigma_s)(\, n \in \Phi_i(\sigma^+) \;\&\; (\sigma{\restriction}2\mathbb{N})^+ \subseteq B \oplus \emptyset \,)\,\} \\ N_t &:= \{\, n \mid (\exists \sigma \supseteq \sigma_t)(\, n \in \Phi_j(\sigma^-) \;\&\; (\sigma{\restriction}2\mathbb{N})^- \subseteq \overline{B} \oplus \emptyset \,)\,\} \;. \end{aligned}$$

Clearly $P_s \leq_{\rm e} B$ and $N_t \leq_{\rm e} \overline{B}$ and also $\Phi_i(A) \subseteq P_s$ and $\Phi_j(\overline{A}) \subseteq N_t$. So now suppose that $N_t \not\subseteq \Phi_j(\overline{A})$. Wlog choose $2n{+}1 \in N_t - \Phi_j(\overline{A})$. Thus there exists $\beta \supseteq \sigma_t$ such that $2n{+}1 \in \Phi_j(\beta^-)$ and $(\beta{\restriction}2\mathbb{N})^- \subseteq \overline{B} \oplus \emptyset$. Also, by hypothesis (that $\Phi_j(\overline{A})$ is characteristic), there exists B-compatible $\alpha \supseteq \sigma_t$ such that $2n \in \Phi_j(\alpha^-)$. Define initial segment γ of length $max\{\,|\alpha)|, |\beta|\,\}$ such that, for all $m < |\gamma|$,

$$\gamma(m) \;=\; \begin{cases} 0 & \text{if } \alpha(m){\downarrow} = 0 \;\vee\; \beta(m){\downarrow} = 0 \;\vee\; c_{B\oplus\mathbb{N}}(m) = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Then γ is a B-compatible extension of σ_t and $2n, 2n{+}1 \in \Phi_j(\gamma^-)$. Thus at stage $(t{+}1)$ the construction would prevent $\Phi_j(\overline{A})$ from being characteristic in contradiction with the hypothesis. $P_s \subseteq \Phi_i(A)$ is proved in a similar way. □

Corollary 17. *There exists a quasi-minimal se-degree.*

Corollary 18. *For any quasi-minimal se-degree* $\boldsymbol{b}$ *there exists a quasi-minimal se-degree* $\boldsymbol{a}$ *such that* $\boldsymbol{b} < \boldsymbol{a}$.

5 The Jump Operator

We now consider the problem of defining the *jump operator* with respect to se-reducibility. By analogy with the *Turing jump* we will require that such an operator be derived from a map that sends any set A to a set A' that is ordered strictly above A by $\leq_{\rm se}$ and that, in addition, possesses certain hardness properties (relative to A). We begin with a reminder of some standard results in the study of enumeration reducibility.

Notation. For any set A, K_A denotes the set $\{ x \mid x \in \Phi_x(A) \}$ and J_A denotes the set $K_A \oplus \overline{K_A}$. Similarly $J_A^{(k)}$ denotes the iterated form of J_A defined by: $J_A^{(0)} = A$ and $J_A^{(k+1)} = J_{J_A^{(k)}}$.

Lemma 19. *For any set A, K_A is 1-complete for* $\mathbf{Enum}(A)$.

Lemma 20. *For any sets A and B: $A \leq_{\mathrm{e}} B$ iff $A \leq_1 K_B$ iff $K_A \leq_1 K_B$.*

Note 21. A jump operator on the enumeration degrees is defined by Cooper and McEvoy in ([1]) as the function induced by $X \mapsto J_X$. We employ the term *e-jump* to refer to this operator. We use $\boldsymbol{a}'_{\mathrm{e}}$ to denote the e-jump of $\boldsymbol{a}_{\mathrm{e}}$. The iterated e-jump of $\boldsymbol{a}_{\mathrm{e}}$ is written $\boldsymbol{a}_{\mathrm{e}}^{(k)}$ (and is defined by $\boldsymbol{a}_{\mathrm{e}}^{(0)} = \boldsymbol{a}_{\mathrm{e}}$ and $\boldsymbol{a}_{\mathrm{e}}^{(k+1)} = (\boldsymbol{a}_{\mathrm{e}}^{(k)})'$).

Notation. For any set A, H_A denotes the set $K_A \oplus K_{\overline{A}}$ and $H_A^{(k)}$ denotes the iterated form of H_A defined by: $H_A^{(0)} = A$ and $H_A^{(k+1)} = H_{H_A^{(k)}}$.

Lemma 22. *For any sets A and B, if $A \leq_{\mathrm{se}} B$ then $H_A \leq_1 H_B$.*

Proof. Let A and B be any sets such that $A \leq_{\mathrm{se}} B$. Then, by definition $A \leq_{\mathrm{e}} B$ and $\overline{A} \leq_{\mathrm{e}} \overline{B}$. Now apply Lemma 20. □

Lemma 23. *For any set A: $A <_{\mathrm{se}} H_A$.*

Proof. Let A be any set. Then by Lemma 19 we know that $A \leq_{\mathrm{se}} H_A$. Also notice that $H_A \leq_{\mathrm{se}} A$ would imply $\overline{K_{\overline{A}}} \leq_{\mathrm{e}} \overline{A}$ from which we easily derive a contradiction. □

Lemma 24. *For any set A, H_A is 1-hard for* $\mathbf{Enum}(A)$. *Moreover, if $deg_{\mathrm{se}}(A)$ is characteristic, then H_A is 1-complete for* $\mathbf{Enum}(A)$.

Proof. Let A be any set. Then Lemma 19 implies that H_A is 1-hard for $\mathbf{Enum}(A)$. Now suppose that $deg_{\mathrm{se}}(A)$ is characteristic. Then $A \equiv_{\mathrm{se}} \overline{A}$ by Proposition 9 and so $K_{\overline{A}} \equiv_1 K_A$ by Lemma 20. However this means that $H_A \equiv_1 K_A$. Therefore H_A is 1-complete for $\mathbf{Enum}(A)$ by Lemma 19. □

Definition 25. *Let $\boldsymbol{a}_{\mathrm{se}}$ be any se-degree. The se-jump of $\boldsymbol{a}_{\mathrm{se}}$ (written $\boldsymbol{a}'_{\mathrm{se}}$) is defined to be $deg_{\mathrm{se}}(H_A)$ for any A in $\boldsymbol{a}_{\mathrm{se}}$. The iterated se-jump of $\boldsymbol{a}_{\mathrm{se}}$ is written $\boldsymbol{a}_{\mathrm{se}}^{(k)}$ and is defined by: $\boldsymbol{a}_{\mathrm{se}}^{(0)} = \boldsymbol{a}_{\mathrm{se}}$ and $\boldsymbol{a}_{\mathrm{se}}^{(k+1)} = (\boldsymbol{a}_{\mathrm{se}}^{(k)})'$.*

Proposition 26. *Let $\boldsymbol{a}_{\mathrm{se}}$ be any se-degree, let $\boldsymbol{a}_{\mathrm{e}}$ be the e-degree of which it is a subclass, and let $\boldsymbol{d}_{\mathrm{e}}$ be the e-degree that contains $\boldsymbol{a}_{\mathrm{se}}^{(2)}$; then $\boldsymbol{a}_{\mathrm{e}} < \boldsymbol{d}_{\mathrm{e}}$. In other words the double se-jump is strictly increasing relative to the relation induced by $\leq_{\mathrm{e}}$ over $\langle \mathcal{D}_{\mathrm{se}}, \leq \rangle$.*

Proof. Suppose that $\boldsymbol{a}_{\mathrm{se}} \subseteq \boldsymbol{a}_{\mathrm{e}}$ and pick any A in $\boldsymbol{a}_{\mathrm{se}}$. Then H_A is 1-hard for $\mathbf{Enum}(A)$ by Lemma 24 and this implies that $\boldsymbol{b}_{\mathrm{se}} \leq \boldsymbol{a}_{\mathrm{se}}^{(1)}$ for any se-degree $\boldsymbol{b}_{\mathrm{se}} \subseteq \boldsymbol{a}_{\mathrm{e}}$. Now, by Lemma 23, $\boldsymbol{a}_{\mathrm{se}}^{(1)} < \boldsymbol{a}_{\mathrm{se}}^{(2)}$ and so $\boldsymbol{a}_{\mathrm{se}}^{(2)} \not\subseteq \boldsymbol{a}_{\mathrm{e}}$. □

6 The Symmetric Enumeration Hierarchy

We now proceed to formulate a generalisation of the relativised Arithmetical Hierarchy based on $\leq_{\mathrm{e}}$ and $\leq_{\mathrm{se}}$ (in place of "*c.e. in*" and $\leq_{\mathrm{T}}$) in such a way that the two hierarchies are identical when relativised to any set A whose se-degree is characteristic. Note that this work develops on methods used by Kevin McEvoy in [2] Chapter 4 to define a similar hierarchy. We start by presenting an alternative definition of the relativised Arithmetical Hierarchy. The equivalence of the latter with the standard definition (see for example [6] Chapter 4) is easily checked.

Notation. For any classes $\mathbf{C}$ and $\mathbf{D}$ we use $\mathbf{C} \oplus \mathbf{D}$ as shorthand for the class $\{ A \oplus B \mid A \in \mathbf{C} \ \& \ B \in \mathbf{D} \}$. Also, for $\mathbf{R} \in \{ \mathbf{Ce}, \mathbf{Comp}, \mathbf{Enum}, \mathbf{s\text{-}Enum} \}$, $\mathbf{R}(\mathbf{C})$ denotes the class $\bigcup \{ \mathbf{R}(E) \mid E \in \mathbf{C} \}$.

Definition 27. *Let A be any set. The* Arithmetical Hierarchy relativised to A *is defined to be...* $\{ \Sigma^A_n, \Pi^A_n, \Delta^A_n \ : \ n \geq 0 \}$
where $\Sigma^A_0 = \Pi^A_0 = \Delta^A_0 = \mathbf{Comp}(A)$ *and, for* $n \geq 0$,
$\Sigma^A_{n+1} = \mathbf{Ce}(\Sigma^A_n \oplus \Pi^A_n)$, $\Pi^A_{n+1} = co\text{-}\Sigma^A_{n+1}$ *and* $\Delta^A_{n+1} = \mathbf{Comp}(\Sigma^A_n \oplus \Pi^A_n)$.
The Arithmetical Hierarchy $\{ \Sigma_n, \Pi_n, \Delta_n : n \geq 0 \}$ *is the special case of this definition with* $A = \emptyset$.

Note 28. $\mathbf{Comp}(A) = \mathbf{Comp}(\{A\} \oplus \{A\})$, so if we define $\Sigma^A_{-1} = \Pi^A_{-1} = \Delta^A_{-1}$ to be $\{A\}$, we obtain $\Delta^A_0 = \mathbf{Comp}(\Sigma^A_{-1} \oplus \Pi^A_{-1})$ in conformity with the definition of Δ^A_{n+1}.

Proposition 29 ([3]). *For any set A, $\Sigma^A_{n+1} = \mathbf{Enum}(J^{(n)}_{\mathbb{C}_A})$, for all $n \geq 0$.*

Definition 30. *Let A be any set. The* SE-hierarchy relativised to A *(written SE(A)- hierarchy) is defined to be...* $\{ \Sigma^{SE,A}_n, \Pi^{SE,A}_n, \Delta^{SE,A}_n \ : \ n \geq 0 \}$
where $\Sigma^{SE,A}_0 = \Pi^{SE,A}_0 = \Delta^{SE,A}_0 = \mathbf{s\text{-}Enum}(A)$
and, for $n \geq 0$, $\Sigma^{SE,A}_{n+1} = \mathbf{Enum}(\Sigma^{SE,A}_n \oplus \Pi^{SE,A}_n)$, $\Pi^{SE,A}_{n+1} = co\text{-}\Sigma^{SE,A}_{n+1}$
and $\Delta^{SE,A}_{n+1} = \mathbf{s\text{-}Enum}(\Sigma^{SE,A}_n \oplus \Pi^{SE,A}_n)$.
The SE-hierarchy $\{ \Sigma^{SE}_n, \Pi^{SE}_n, \Delta^{SE}_n : n \geq 0 \}$ *is the special case of this definition with* $A = \emptyset$.

Note 31. $\mathbf{s\text{-}Enum}(A) = \mathbf{s\text{-}Enum}(\{A\} \oplus \{A\})$ and so in this case also (see Note 28), if we define $\Sigma^{SE,A}_{-1} = \Pi^{SE,A}_{-1} = \Delta^{SE,A}_{-1}$ to be $\{A\}$ we obtain $\Delta^{SE,A}_0 = \mathbf{s\text{-}Enum}(\Sigma^{SE,A}_{-1} \oplus \Pi^{SE,A}_{-1})$. Notice also that obviously, for all $n \geq 0$, $\Sigma^{SE,A}_n \bigcup \Pi^{SE,A}_n \subseteq \Delta^{SE,A}_{n+1}$ and that $\Sigma^{SE,A}_{n+1}$ and $\Delta^{SE,A}_n$ are closed under $\leq_{\mathrm{e}}$ and $\leq_{\mathrm{se}}$ respectively (by transitivity).

Proposition 32. *Let A be any set. Then for all $n \geq 0$, $\Sigma^{SE,A}_{n+1} = \mathbf{Enum}(J^{(n)}_A)$ and $\Delta^{SE,A}_{n+1} = \mathbf{s\text{-}Enum}(J^{(n)}_A)$.*

Proof. It suffices to show that $J_A^{(n)}$ is e-complete for $\Sigma_{n+1}^{SE,A}$ and se-complete for $\Delta_{n+1}^{SE,A}$ (for all $n \geq 0$), since the latter are closed under $\leq_{\mathrm{e}}$ and $\leq_{\mathrm{se}}$ respectively. The case $(\boldsymbol{n = 0})$ is straightforward, so we assume that $(\boldsymbol{n > 0})$.

- Suppose that $B \in \Sigma_{n+1}^{SE,A}$. Then there exist sets $C \in \Sigma_n^{SE,A}$ and $D \in \Pi_n^{SE,A}$ such that $B \leq_{\mathrm{e}} C \oplus D$. By the Induction Hypothesis, $C \leq_{\mathrm{e}} J_A^{(n-1)}$ and $\overline{D} \leq_{\mathrm{e}} J_A^{(n-1)}$. Hence by Lemma 20, $C \leq_1 K_{J_A^{(n-1)}}$ and $\overline{D} \leq_1 K_{J_A^{(n-1)}}$. However this implies that $B \leq_{\mathrm{e}} C \oplus D \leq_1 K_{J_A^{(n-1)}} \oplus \overline{K}_{J_A^{(n-1)}} =_{def} J_A^{(n)}$. Thus $B \leq_{\mathrm{e}} J_A^{(n)}$. Also, by the Induction Hypothesis, and the closure of $\Sigma_n^{SE,A}$ under $\leq_{\mathrm{e}}$ (using $K_{J_A^{(n-1)}} \leq_{\mathrm{e}} J_A^{(n-1)}$) we obtain $J_A^{(n)} =_{def} K_{J_A^{(n-1)}} \oplus \overline{K}_{J_A^{(n-1)}} \in \Sigma_n^{SE,A} \oplus \Pi_n^{SE,A}$. It follows that $J_A^{(n)} \in \Delta_{n+1}^{SE,A} \subseteq \Sigma_{n+1}^{SE,A}$.
- Suppose that $B \in \Delta_{n+1}^{SE,A}$. Then $B \leq_{\mathrm{e}} C \oplus D$ and $\overline{B} \leq_{\mathrm{e}} \overline{C} \oplus \overline{D} \equiv_1 \overline{D} \oplus \overline{C}$, for some $C \in \Sigma_n^{SE,A}$ and $D \in \Pi_n^{SE,A}$. Thus $B, \overline{B} \leq_{\mathrm{e}} J_A^{(n)}$ by the first part of this proof. It follows that $\overline{B} \leq_{\mathrm{e}} \overline{J_A^{(n)}}$ since $J_A^{(n)}$ is characteristic (using Lemma 7); i.e. $B \leq_{\mathrm{se}} J_A^{(n)}$. Moreover, $J_A^{(n)} \in \Delta_{n+1}^{SE,A}$ by the first part of this proof. □

Note 33. It follows from Lemma 19 and Proposition 32 that, for all $n \geq 0$, $K_{J_A^{(n)}}$ is 1-complete for $\Sigma_{n+1}^{SE,A}$.

Proposition 34. *Let A be any set such that $deg_{\mathrm{se}}(A)$ is characteristic. Then, for all $n \geq 0$, $\Delta_n^{SE,A} = \Sigma_n^{SE,A} \bigcap \Pi_n^{SE,A}$.*

Proof. Let A be any set of characteristic se-degree. We argue by cases for $n \geq 0$. The case $(\boldsymbol{n = 0})$ holds by definition. For $(\boldsymbol{n > 0})$ we use the fact that $J_A^{(n-1)} \equiv_{\mathrm{se}} \overline{J_A^{(n-1)}}$ for all n (by Lemma 7 for $n > 1$ and by hypothesis for $n = 1$). Accordingly, by Proposition 32, $B \in \Delta_n^{SE,A}$ iff $B \leq_{\mathrm{e}} J_A^{(n-1)}$ & $\overline{B} \leq_{\mathrm{e}} \overline{J_A^{(n-1)}}$ iff $B \leq_{\mathrm{e}} J_A^{(n-1)}$ & $\overline{B} \leq_{\mathrm{e}} J_A^{(n-1)}$ iff $B \in \Sigma_n^{SE,A} \bigcap \Pi_n^{SE,A}$. □

Note 35. The proof of Proposition 34 clearly also implies that, for *any* set A, $\Delta_n^{SE,A} = \Sigma_n^{SE,A} \bigcap \Pi_n^{SE,A}$ for all $n \neq 1$.

Theorem 36. *Suppose that $\Gamma \in \{\Sigma, \Pi, \Delta\}$ and let be A be any set. Then, for all $n \geq 0$, $\Gamma_n^A = \Gamma_n^{SE,\mathbb{C}_A}$.*

Proof. Let A be any set. We reason by cases for $n \geq 0$.
$(\boldsymbol{n = 0})$ An easy application of Proposition 13.
$(\boldsymbol{n > 0})$ $\Sigma_n^A = \mathbf{Enum}(J_{\mathbb{C}_A}^{(n-1)}) = \Sigma_n^{SE,\mathbb{C}_A}$ by Propositions 29 and 32 and so the case $\Gamma = \Sigma$ holds. The case $\Gamma = \Pi$ then follows by definition and the case $\Gamma = \Delta$ follows from the other two cases by applying Proposition 34. □

Theorem 37. *For any set A, if $deg_{\mathrm{se}}(A)$ is characteristic then the Arithmetical Hierarchy relativised to A and the SE(A)-hierarchy are identical.*

Proof. Suppose that $\Gamma \in \{\Sigma, \Pi, \Delta\}$ and let be A be any set with characteristic se-degree. By Theorem 36, it suffices to show that $\Gamma_n^{SE,A} = \Gamma_n^{SE,\mathbb{C}_A}$ for all $n \geq 0$. We reason by induction.
$(n = 0)$ Since $deg_{\rm se}(A)$ is characteristic, $A \equiv_{\rm se} \mathbb{C}_A$ by Proposition 9 (and Lemma 7) and so $\Gamma_0^{SE,A} = \textbf{s-Enum}(A) = \textbf{s-Enum}(\mathbb{C}_A) = \Gamma_0^{SE,\mathbb{C}_A}$.
$(n > 0)$ Note that the identity $\Sigma_n^{SE,A} = \Sigma_n^{SE,\mathbb{C}_A}$ is, by definition, equivalent to $\textbf{Enum}(\Sigma_{n-1}^{SE,A} \oplus \Pi_{n-1}^{SE,A}) = \textbf{Enum}(\Sigma_{n-1}^{SE,\mathbb{C}_A} \oplus \Pi_{n-1}^{SE,\mathbb{C}_A})$ which holds by the Induction Hypothesis (i.e. that $\Sigma_{n-1}^{SE,A} = \Sigma_{n-1}^{SE,\mathbb{C}_A}$). This proves the case $\Gamma = \Sigma$. The case $\Gamma = \Pi$ then follows by definition and the case $\Gamma = \Delta$ follows from the other two cases by applying Proposition 34. □

Corollary 38. *The Arithmetical Hierarchy is identical to the SE-hierarchy.*

Note 39. $\overline{A} \leq_{\rm e} A$ implies that $A \equiv_{\rm e} \mathbb{C}_A$ and so it follows, by application of Lemma 12, that $\textbf{Ce}(A) = \textbf{Enum}(A)$. Thus by modifying the proof of Theorem 37, it can be shown that $\Gamma_n^A = \Gamma_n^{SE,A}$ holds for $\Gamma \in \{\Sigma, \Pi\}$ and $n \geq 1$, and for $\Gamma = \Delta$ and $n \geq 2$, whenever $\overline{A} \leq_{\rm e} A$.

In addition to the above results it is easy to show (by induction) that, for $\Gamma \in \{\Sigma, \Pi, \Delta\}$, and for any set A

$$\Gamma_n \quad \subseteq \quad \Gamma_n^{SE,A} \quad \subseteq \quad \Gamma_n^A \ . \tag{1}$$

However, despite the similarities between the hierarchies it should be emphasised that, for any set A, the SE(A)-hierarchy distinguishes between A and $\mathbb{C}_A$ whenever A is not of characteristic se-degree (i.e. $\overline{A} \not\equiv_{\rm se} A$) whereas for the relativised Arithmetical hierarchy, A and $\mathbb{C}_A$ are equivalent. In particular, if $\overline{A} \not\equiv_{\rm se} A$ then $\mathbb{C}_A \in \Delta_n^A - \Delta_n^{SE,A}$ for $n \leq 1$ and if (the stronger condition) $\overline{A} \not\leq_{\rm e} A$ holds, then $\mathbb{C}_A \in \Sigma_1^A - \Sigma_1^{SE,A}$. This leads to the obvious question of whether the difference ever propogates up the hierarchy and if so, what conditions have to be fulfilled by A for this to happen. For example, supposing that $\Gamma \in \{\Sigma, \Pi, \Delta\}$, does $\overline{A} \not\leq_{\rm e} A$ imply that $\Gamma_n^A \neq \Gamma_n^{SE,A}$ for all $n \geq 0$? We proceed by giving a partial answer to this question.

Note 40. For any set A, if $\overline{A} \leq_{\rm e} A$ then $K_A \leq_{\rm e} \overline{K}_A$ (since (a) $\overline{A} \leq_{\rm e} A$ iff $\overline{A} \leq_1 K_A$ iff $A \leq_1 \overline{K}_A$ and (b) $K_A \leq_{\rm e} A$).

Lemma 41. *For all $n \geq 0$, $J_{\mathcal{K}}^{(n)} \in \Sigma_{n+1}$.*

Proof. By induction on $n \geq 0$. Note that the Case **$(n = 0)$** is just the fact that $\mathcal{K} \in \Sigma_1$. Case **$(n = 1)$** holds because $J_{\mathcal{K}}^{(1)} =_{def} K_{\mathcal{K}} \oplus \overline{K}_{\mathcal{K}} \leq_{\rm e} \overline{K}_{\mathcal{K}} \in \Pi_1$ (which implies that $J_{\mathcal{K}}^{(1)} \in \Sigma_2$). For the Case **$(n + 2)$** note that $\overline{J_{\mathcal{K}}^{(n+1)}} \leq_{\rm e} J_{\mathcal{K}}^{(n+1)}$ since $J_{\mathcal{K}}^{(n+1)}$ is characteristic, and so $J_{\mathcal{K}}^{(n+2)} =_{def} K_{J_{\mathcal{K}}^{(n+1)}} \oplus \overline{K}_{J_{\mathcal{K}}^{(n+1)}} \leq_{\rm e} \overline{K}_{J_{\mathcal{K}}^{(n+1)}}$, by Note 40. However $J_{\mathcal{K}}^{(n+1)} \in \Sigma_{n+2}$ by the Induction Hypothesis and hence $K_{J_{\mathcal{K}}^{(n+1)}} \in \Sigma_{n+2}$ (since $K_{J_{\mathcal{K}}^{(n+1)}} \leq_{\rm e} J_{\mathcal{K}}^{(n+1)}$), which implies that $\overline{K}_{J_{\mathcal{K}}^{(n+1)}} \in \Pi_{n+2}$. Therefore $J_{\mathcal{K}}^{(n+2)} \in \Sigma_{n+3}$. □

Proposition 42. *Suppose that* $\Gamma \in \{\Sigma, \Pi, \Delta\}$. *There exists a set* A *such that* $\Gamma_n^{SE,A} \neq \Gamma_n^A$ *for all* $n \geq 0$.

Proof. Let $A = \mathcal{K}$. Note firstly that $\mathcal{K}^{(n+2)}$ and $K_{J_{\mathcal{K}}^{(n)}}$ are 1-complete for $\Sigma_{n+1}^{\mathcal{K}}$ and $\Sigma_{n+1}^{SE,\mathcal{K}}$ respectively for all $n \geq 0$. Choose any n and suppose that $\Sigma_{n+1}^{\mathcal{K}} = \Sigma_{n+1}^{SE,\mathcal{K}}$. This would mean that $\mathcal{K}^{(n+2)} \equiv_1 K_{J_{\mathcal{K}}^{(n)}}$. Now, by Lemma 41, $J_{\mathcal{K}}^{(n)} \in \Sigma_{n+1}$ and so $K_{J_{\mathcal{K}}^{(n)}} \in \Sigma_{n+1}$ (since $K_{J_{\mathcal{K}}^{(n)}} \leq_{\mathrm{e}} J_{\mathcal{K}}^{(n)}$). Thus $\mathcal{K}^{(n+2)} \leq_1 \mathcal{K}^{(n+1)}$ (contradiction). This proves the cases $\Gamma \in \{\Sigma, \Pi, \Delta\}$ for $n \geq 2$ (using Note 35) and $\Gamma \in \{\Sigma, \Pi\}$ for $n = 1$. Also $\mathbb{C}_{\mathcal{K}} \in \Delta_n^{\mathcal{K}} - \Delta_n^{SE,\mathcal{K}}$ for $n \leq 1$ and so the proof is complete. □

A further point to be underlined is that the methodology inherent to the SE-hierarchy relates to enumeration reducibility (and indeed symmetric enumeration reducibility) in much the same way in which that of the Arithmetical Hierarchy relates to Turing reducibility. Our last result (Theorem 45) is an example of this relationship and of the manner in which the SE-hierarchy might contribute to certain aspects of the theory of enumeration reducibility.

Definition 43. *An e-degree* $\boldsymbol{a}_{\mathrm{e}} \leq \mathbf{0}_{\mathrm{e}}^{(1)}$ *is defined to be* low_n *if* $\boldsymbol{a}_{\mathrm{e}}^{(n)} = \mathbf{0}_{\mathrm{e}}^{(n)}$ *and* high_n *if* $\boldsymbol{a}_{\mathrm{e}}^{(n)} = \mathbf{0}_{\mathrm{e}}^{(n+1)}$, *for all* $n \geq 0$ *(the case* $n = 0$ *being trivial). A set* $A \leq_{\mathrm{e}} J$ *is said to be* e-low_n *(*e-high_n*) if* $deg_{\mathrm{e}}(A)$ *is* low_n *(*high_n*). If* $n = 1$ *we abbreviate these terms to* low, high *etc.*

Lemma 44. *For all* $n \geq 0$, $K_{J^{(n)}} \equiv_1 \mathcal{K}^{(n+1)}$.

Proof. For all $n \geq 0$, $\Sigma_{n+1}^{SE} = \Sigma_{n+1}$ by Corollary 38. Hence it suffices to note that $K_{J^{(n)}}$ is 1-complete for Σ_{n+1}^{SE} whereas $\mathcal{K}^{(n+1)}$ is 1-complete for Σ_{n+1}. □

Theorem 45. *For any set* $A \leq_{\mathrm{e}} J$ *and* $n \geq 1$*:*
(a) *A is e-low*$_n$ *iff* $\Sigma_n^{SE,A} \subseteq \Pi_{n+1}$,
(b) *A is e-high*$_n$ *iff* $\Pi_{n+1} \subseteq \Sigma_{n+1}^{SE,A}$ *iff* $\overline{\mathcal{K}^{(n+1)}} \leq_{\mathrm{e}} J_A^{(n)}$.

Proof. Assume that $n \geq 1$ and that $A \leq_{\mathrm{e}} J$.

A is e-low$_n$ iff $J_A^{(n)} \leq_{\mathrm{e}} J^{(n)}$ (which means that $\Sigma_{n+1}^{SE,A} \subseteq \Sigma_{n+1}^{SE}$) iff $K_{J_A^{(n-1)}} \leq_{\mathrm{T}} K_{J^{(n-1)}}$ by an easy modification of Lemma 13, iff $K_{J_A^{(n-1)}} \leq_{\mathrm{T}} \mathcal{K}^{(n)}$ by Lemma 44, iff $\Sigma_n^{SE,A} \subseteq \Delta_{n+1}$ by Note 33, iff $\Sigma_n^{SE,A} \subseteq \Pi_{n+1}$ since $\Sigma_n^{SE,A} \subseteq \Sigma_{n+1}^{SE,A} \subseteq \Sigma_{n+1}^{SE}$ (see above) and $\Sigma_{n+1}^{SE} = \Sigma_{n+1}$ by Corollary 38.

A is e-high$_n$ iff $J^{(n+1)} \leq_{\mathrm{e}} J_A^{(n)}$ iff $J^{(n+1)} \leq_{\mathrm{se}} J_A^{(n)}$ since $J^{(n+1)}$ and $J_A^{(n)}$ are characteristic, iff $K_{J^{(n)}} \leq_{\mathrm{se}} J_A^{(n)}$ by Lemma 13, iff $\mathcal{K}^{(n+1)} \leq_{\mathrm{se}} J_A^{(n)}$ by Lemma 44, iff $\Sigma_{n+1} \subseteq \Delta_{n+1}^{SE,A}$ iff $\Sigma_{n+1} \subseteq \Pi_{n+1}^{SE,A}$ (as $\Sigma_{n+1} \subseteq \Sigma_{n+1}^{SE,A}$ by Equation 1) iff $\Pi_{n+1} \subseteq \Sigma_{n+1}^{SE,A}$ iff $\overline{\mathcal{K}^{(n+1)}} \leq_{\mathrm{e}} J_A^{(n)}$ since $\overline{\mathcal{K}^{(n+1)}}$ is 1-complete for Π_{n+1}. □

References

1. Cooper S.B. and McEvoy K. On Minimal Pairs of Enumeration Degrees. *Journal of Symbolic Logic*, **50** (4), 983-1001, 1985.
2. McEvoy K. *The Structure of the Enumeration Degrees*. PhD thesis, The University of Leeds, UK, October 1984.
3. McEvoy K. Jumps of Quasi-minimal Enumeration Degrees. *Journal of Symbolic Logic* **50** (4), 839-848, 1985.
4. Medvedev Y. T. On Non-isomorphic Recursively Enumerable Sets. *Doklady Academii Nauk*, **102**, 211-214, 1955.
5. Selman A. L. Arithmetical Reducibilities II. *Zeitshrift Math. Logik Grundlagen Math.*, **18**, 83-92, 1972.
6. Soare R.I. *Recursively Enumerable Sets and Degrees*. Springer, Berlin, 1987.

Computability-Theoretic and Proof-Theoretic Aspects of Vaughtian Model Theory

Denis R. Hirschfeldt*

Department of Mathematics,
The University of Chicago,
5734 S. University Avenue,
Chicago, IL 60637, USA
drh@math.uchicago.edu

I will discuss some recent results in the analysis of the computability-theoretic and proof-theoretic content of Vaughtian model theory, that is, the study of special models such as prime, saturated, and homogeneous models, and associated results such as the omitting types theorem. This is a research program dating back to the 1970's (see [4]), but which has recently picked up steam with the application of finer computability-theoretic tools. Here are two examples.

Goncharov-Nurtazin [1] and Millar [2] showed that there is a complete decidable theory T such that all the types of T are computable, but T has no decidable prime model. However, as I will show, for any noncomputable D, such a theory T must have a D-decidable prime model. This fact yields as a corollary the existence of a noncomputably presentable structure with D-computable presentations for all noncomputable D, a result due to Slaman [5] and to Wehner [6].

Given a complete decidable atomic theory T, a natural way to try to obtain a decidable prime model of T is to effectively omit the nonprincipal types of T. And indeed, Millar [3] showed that if S is a uniformly computable set of complete types of T, then there is a decidable model of T omitting all the nonprincipal types in S. Unfortunately, in the same way that we cannot effectively list all the computable functions, we may not be able to effectively list the computable complete types of T. On the other hand, we can list the computable partial types of T, so it becomes interesting to ask for which D is it the case that there is always a D-decidable model of T omitting all the nonprincipal types in a uniformly computable family of partial types of T. I will show that this is the case if and only if D has hyperimmune degree.

References

1. S. S. Goncharov and A. T. Nurtazin, Constructive models of complete decidable theories, Algebra and Logic 12 (1973) 67–77.
2. T. S. Millar, Foundations of recursive model theory, Ann. Math. Logic 13 (1978) 45–72.

* Partially supported by NSF grant DMS-02-00465.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 209–210, 2005.

3. T. S. Millar, Omitting types, type spectrum, and decidability, J. Symbolic Logic 48 (1983) 171–181.
4. V. S. Harizanov, Pure computable model theory, in *Handbook of Recursive Mathematics* (Yu. L. Ershov, S. S. Goncharov, A. Nerode, and J. B. Remmel, eds.), Stud. Logic Found. Math. 138–139 (1998), Elsevier Science, Amsterdam, 3–114.
5. T. A. Slaman, Relative to any nonrecursive set, Proc. Amer. Math. Soc. 126 (1998) 2117–2122.
6. S. Wehner, Enumerations, countable structures, and Turing degrees, Proc. Amer. Math. Soc. 126 (1998) 2131–2139.

Finite Trees as Ordinals

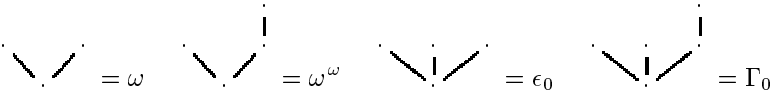
Herman Ruge Jervell

Institutt for lingvistiske og nordiske studier,
Universitetet i Oslo Henrik Wergelands hus,
Niels Henrik Abels vei 36,
0313 Oslo, Norway

Abstract. Finite trees are given a well ordering in such a way that there is a 1-1 correspondence between finite trees and an initial segment of the ordinals. The ordinal ϵ_0 is the supremum of all binary trees. We get the (fixpoint free) n-ary Veblen hierarchy as tree functions and the supremum of all trees is the small Veblen ordinal $\phi_{\Omega^\omega}(0)$.

1

We work with ordinary finite trees with immediate subtrees ordered sequentially from left to right and with no labels at the nodes. Here we just call them trees. The smallest tree is $\cdot$ and we draw the trees with the root at the bottom. Here are four examples where we have indicated the corresponding ordinals in the ordering defined below



We write $\langle \mathbf{A} \rangle$ for the finite sequence of immediate subtrees of the tree $\mathbf{A}$, and $\langle \cdot \rangle$ is the empty sequence. Equality between trees is the usual equality. Given that we already know the ordering of some trees we let

$\mathbf{A} \leq \langle \mathbf{B} \rangle$ **:** There is an immediate subtree $\mathbf{B}_i$ of $\mathbf{B}$ such that either $\mathbf{A} < \mathbf{B}_i$ or $\mathbf{A} = \mathbf{B}_i$

$\langle \mathbf{A} \rangle < \mathbf{B}$ **:** For all immediate subtrees $\mathbf{A}_j$ of $\mathbf{A}$ we have $\mathbf{A}_j < \mathbf{B}$

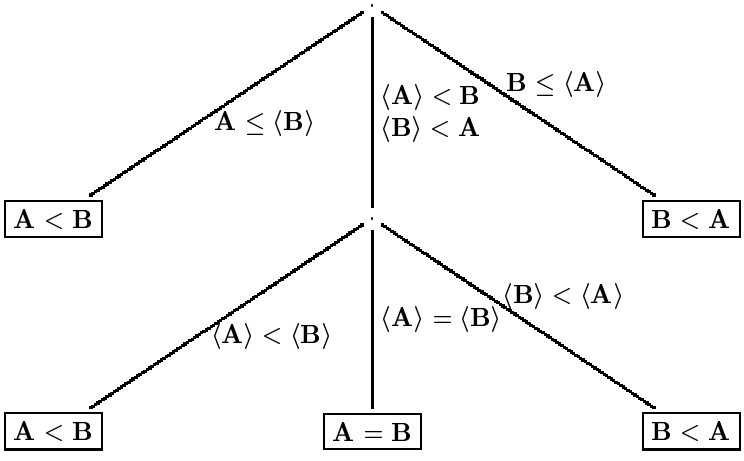
$\langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle$ **:** The inverse lexicographical ordering of the immediate subtrees — we first check which sequence have smallest length, and if they have equal length we look at the rightmost immediate subtree where they differ

We are now ready to define the ordering of trees by recursion over the immediate subtrees.

$$\boxed{\mathbf{A} < \mathbf{B} \;\Leftrightarrow\; \mathbf{A} \leq \langle \mathbf{B} \rangle \;\vee\; (\langle \mathbf{A} \rangle < \mathbf{B} \wedge \langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle)}$$

We must prove that this defines an ordering. The following decision tree shows (by induction) that we have a total relation

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 211–220, 2005.



To get that this total relation is a total ordering we prove that it is transitive. As before the argument is by induction over the heights of the trees. So assume we have

$$\mathbf{A} < \mathbf{B} < \mathbf{C}$$

and want to prove by induction over the sum of heights of the three trees that we get $\mathbf{A} < \mathbf{C}$. This is done by cases

- $\mathbf{B} \leq \langle \mathbf{C} \rangle$: Then $\mathbf{A} < \mathbf{B} \leq \langle \mathbf{C} \rangle$, and by induction $\mathbf{A} \leq \langle \mathbf{C} \rangle$ and $\mathbf{A} < \mathbf{C}$
- $\langle \mathbf{B} \rangle < \langle \mathbf{C} \rangle$ and $\langle \mathbf{B} \rangle < \mathbf{C}$: Then
  - $\mathbf{A} \leq \langle \mathbf{B} \rangle$: Then $\mathbf{A} \leq \langle \mathbf{B} \rangle < \mathbf{C}$ and by induction $\mathbf{A} < \mathbf{C}$
  - $\langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle$ and $\langle \mathbf{A} \rangle < \mathbf{B}$: Then we have $\mathbf{A} < \mathbf{C}$ from
    - $\langle \mathbf{A} \rangle < \langle \mathbf{B} \rangle < \langle \mathbf{C} \rangle$ which gives by induction $\langle \mathbf{A} \rangle < \langle \mathbf{C} \rangle$
    - $\langle \mathbf{A} \rangle < \mathbf{B} < \mathbf{C}$ which gives by induction $\langle \mathbf{A} \rangle < \mathbf{C}$

Theorem 1. *The ordering between finite trees is a total ordering where the equality is the usual equality between trees.*

By induction over the build up of trees we prove

Theorem 2. *Let $\boldsymbol{T}(x)$ be a tree where x indicates a place where we can substitute trees. Then*

$$\boldsymbol{A} < \boldsymbol{B} \;\Leftrightarrow\; \boldsymbol{T}(\boldsymbol{A}) < \boldsymbol{T}(\boldsymbol{B})$$

Furthermore by induction over the build up

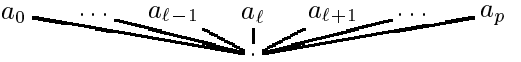
Theorem 3. *If tree $\boldsymbol{S}$ can be embedded in tree $\boldsymbol{T}$, then $\boldsymbol{S} \leq \boldsymbol{T}$.*

2

We note here that we have the following:

$0 =$. $\quad 1 =$! $\quad \omega =$ 

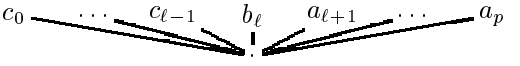
But we need to develop more of the theory to come to these and other calculations. First we give approximations of a tree from below. Given a tree **A** with immediate subtrees



The immediate subtrees a_i of **A** are smaller.
Now assume

- $b_l < a_l$
- $c_i <$ **A** for all $i < l$

Then the following tree is less than **A**



This can be rephrased that for $b_l < a_l$ the function which to $x_0, \ldots, x_{\ell-1}$ gives

is closed under **A**. We also get that for $s < p$ that **A** is closed under the function which to $x_0, \ldots, x_s$ gives

In the usual theory of ordinal notations we use fundamental sequences as a way to approach ordinals from below. [2] For a given tree **A** we call

Fundamental subtrees of A : The immediate subtrees of **A**.
Fundamental functions of A : The two types of functions above.
Fundamental set of A : The set of trees generated by the fundamental functions starting with the fundamental subtrees.
Elementary fundamental function of A : We first get unary functions by letting all variables except the rightmost be 0. Then use all such unary functions of the first type. If there are no functions of the first type use the one of the second type with largest branching.

Elementary fundamental set of A : The set of trees generated by the elementary fundamental functions starting with the fundamental subtrees.

We denote the fundamental set of $\mathbf{A}$ with $\mathcal{F}(\mathbf{A})$ and we shall write it as

$$[S, \ldots, T|F, \ldots, G]$$

where we have displayed the fundamental subtrees $S, \ldots, T$ and the fundamental functions $F, \ldots, G$. Similarly for the elementary fundamental set $\mathcal{H}(\mathbf{A})$

We have the following:

$$\mathcal{F}(\,.\,) = \emptyset$$

$$\mathcal{F}(\;\vdots\;) = [\cdot\,|\,]$$

$$\mathcal{F}(\;\diagdown\!\diagup\;) = [\cdot\,|\,x]$$

$$\mathcal{F}(\;\diagdown\!\diagup\!\vdots\;) = [\cdot, \vdots\,|\,x, y\;\diagdown\!\diagup\;]$$

$$\mathcal{F}(\;\diagdown\!\vdots\!\diagup\;) = [\cdot\,|\,x, y\;\diagdown\!\diagup\;z]$$

$$\mathcal{H}(\;\diagdown\!\vdots\!\diagup\;) = [\cdot\,|\;\diagdown\!\diagup\;x]$$

Here x, y, z are variables used for describing fundamental functions.

The following theorem shows the importance of the fundamental sets.

Theorem 4. *For any tree* $\boldsymbol{A}$*:*

$$\boldsymbol{B} < \boldsymbol{A} \Leftrightarrow \exists \boldsymbol{C} \in \mathcal{F}(\boldsymbol{A}).\boldsymbol{C} \geq \boldsymbol{B}$$

We prove this by induction over the height of $\mathbf{B}$. It is trivial for height 0. So assume it proved for smaller heights than the height of $\mathbf{B}$. The direction $\Leftarrow$ is obvious. We assume $\mathbf{B} < \mathbf{A}$ and divide up into cases:

$\mathbf{B} \leq \langle \mathbf{A} \rangle$ **:** But then $\mathbf{B}$ is less than or equal to one of the fundamental subtrees of $\mathbf{A}$.

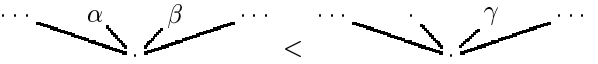
$\langle \mathbf{B} \rangle < \mathbf{A} \wedge \langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ **:** By induction — to each immediate subtree $\mathbf{B}_i$ there is an $\mathbf{C}_i \in \mathcal{F}(\mathbf{A})$ with $\mathbf{C}_i \geq \mathbf{B}_i$. Depending on how we prove $\langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ we get a fundamental function which we can apply to some of the $\mathbf{C}_i$'s to get a $\mathbf{C} \in \mathcal{F}(\mathbf{A})$ with $\mathbf{C} \geq \mathbf{B}$

And the theorem is proved. We can also use the elementary fundamental set

Theorem 5. *For any tree* $\boldsymbol{A}$*:*

$$\boldsymbol{B} < \boldsymbol{A} \Leftrightarrow \exists \boldsymbol{C} \in \mathcal{H}(\boldsymbol{A}).\boldsymbol{C} \geq \boldsymbol{B}$$

We only need to note that



where $\gamma > \max(\alpha, \beta)$ and that the result of of an application of the second type of fundamental function can be embedded into an application of the first type.

We have also

Theorem 6. *Assume that all trees less than or equal to the fundamental subtrees of* $\boldsymbol{A}$ *is contained in* $\mathcal{F}(\boldsymbol{A})$*, then*

$$\boldsymbol{B} < \boldsymbol{A} \Leftrightarrow \boldsymbol{B} \in \mathcal{F}(\boldsymbol{A})$$

The proof follows the lines above. We have induction over the height of $\mathbf{B}$ and get to the cases

$\mathbf{B} \leq \langle \mathbf{A} \rangle$ **:** Then by assumption $\mathbf{B} \in \mathcal{F}(\mathbf{A})$.

$\langle \mathbf{B} \rangle < \mathbf{A} \wedge \langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ **:** By induction — for each immediate subtree $\mathbf{B}_i$ we have $\mathbf{B}_i \in \mathcal{F}(\mathbf{A})$. Depending on how we prove $\langle \mathbf{B} \rangle < \langle \mathbf{A} \rangle$ we get a fundamental function which we can apply to the $\mathbf{B}_i$'s to get $\mathbf{B} \in \mathcal{F}(\mathbf{A})$.

We call a fundamental set which is an initial segment of the ordinals for *full*. The fundamental sets mentioned above are full.

Theorem 7. $\overset{\alpha}{\vdots} = \alpha + 1$

We can prove this by simple induction over trees. We prove

$$\beta < \overset{\alpha}{\vdots} \Leftrightarrow \beta \leq \alpha$$

or we can use that the ordering is a wellordering and then induction over α noting that

$$\mathcal{F}(\overset{\alpha}{\vdots}) = [\alpha \mid \overset{\alpha-}{\vdots}]$$

We are now getting a clearer picture of the ordering. The trees can be divided into layers — we let $\mathcal{T}_i$ be the trees with at most i-branchings. We then get that $\mathcal{T}_1$ is majorised by

and this tree is the least in $\mathcal{T} - \mathcal{T}_1$. The $\mathcal{T}_2$ is majorised by

and this tree is the least in $\mathcal{T} - \mathcal{T}_2$. The $\mathcal{T}_3$ are majorised by

and this tree is the least in $\mathcal{T} - \mathcal{T}_3$. And so on.

3

So far the properties could be proved by simple inductions over the heights. Below we shall prove that the trees are well ordered. This requires a stronger method of proof. Let $\mathcal{T}$ be the initial segment of trees which are well ordered. We then prove

Theorem 8. *Assume $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_k$ are well ordered. Then so is $\boldsymbol{S}$ given by*

Let $\mathbf{A} < \mathbf{S}$ be as low as possible and not well ordered. We shall show that $\mathbf{A}$ is well ordered. This is done by the inverse lexicographical ordering of $\mathbf{S}_1, \ldots, \mathbf{S}_k$ where the field of the inverse lexicographical ordering is the set $\mathcal{T}$ of well ordered trees. We now have the following cases

$\mathbf{A} \leq \langle \mathbf{S} \rangle$ **:** Then $\mathbf{A} \leq \mathbf{S}_i$ and is therefore well ordered.

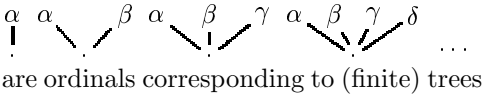
$\langle \mathbf{A} \rangle < \mathbf{S} \wedge \langle \mathbf{A} \rangle < \langle \mathbf{S} \rangle$ **:** Then all immediate subtrees of $\mathbf{A}$ are less than $\mathbf{S}$ and since $\mathbf{A}$ is lowest, then all the immediate subtrees of $\mathbf{A}$ are well ordered. But the immediate subtrees of $\mathbf{A}$ comes before the immediate subtrees of $\mathbf{S}$ in the inverse lexicographical ordering of well ordered sets and with field $\mathcal{T}$. We conclude that $\mathbf{A}$ is well ordered.

The proof is finished. Observe that this proof can be formalized and proved within the theory of inductive definitions. This means — in proof theoretical terms — that the trees give an initial segment of the ordinals, and this segment is below the Howard ordinal. Using that the ordering respects embedding we can lower this estimate considerably to the small Veblen ordinal. [3]

Theorem 9. *The ordering is a well order.*

For the rest of the paper when we talk about ordinals, we mean ordinals from the initial segment given by the (finite) trees. With our orderings on the trees we have obtained an easy translation between ordinals and trees. Note that this is an improvement over the usual theories of ordinal notations. Then we can have multiple representations of an ordinal and must worry how to pick a notation for an ordinal.

The function below are well defined:



where $\alpha, \beta, \gamma, \delta, \ldots$ are ordinals corresponding to (finite) trees. The first function is the successor, but the other functions have not been characterized so far.

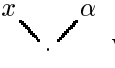
4

Here we want to describe the trees $\mathcal{T}_2$ with at most binary branching. We know that the supremum of it is

We shall prove that this is in fact the ordinal ϵ_0. We can describe $\mathcal{T}_2$ as the set of ordinals/trees given by

Start: ·

Successor: The function f_+ given by $x \mapsto$ 

α**-jump:** The function f_α given by $x \mapsto$ 

where α is in $\mathcal{T}_2$

Furthermore we have a 1-1 correspondence between the terms given above and the ordinals in $\mathcal{T}_2$. This can be used to describe the ordinals less than

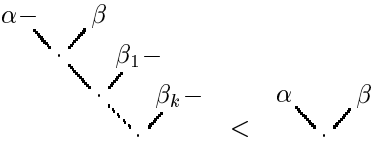
The ordinals are given by terms built up from f_+ and f_0. For simplicity we just write down the sequence of indices. So we have a finite sequence of $+$ and 0. The empty sequence correspond to 0, and the finite ordinals are $+$,$++$, $+++$, $++++$, ... The tree ordering correspond exactly to the lexicographical ordering of the sequences where we let the 0's be signs separating the $+$'s. The sequence $0++0+00+++0++$ correspond to the sequence $\langle 0,2,1,0,3,2\rangle$ and this is again given by the ordinal

$$\omega^5 + \omega^4 \cdot 2 + \omega^3 \cdot 1 + \omega^2 \cdot 0 + \omega^1 \cdot 3 + 2$$

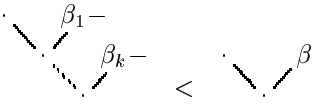
We get

Theorem 10. $\omega^\omega =$ 

It is no surprise that we get connections to the lexicographical ordering in the ordering of binary trees. Let us note that we have



and



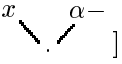
where $\alpha- < \alpha$ and $\beta_i- < \beta$. These are the crucial properties for the lexicographical ordering.

We now want to characterize the function

$$g(\alpha) = \quad$$

The fundamental set is

$$[\cdot, \alpha | x \mapsto x + 1, x \mapsto \quad]$$



where $\alpha-$ runs over the ordinals $< \alpha$. Furthermore the fundamental set gives all the ordinals less than $g(\alpha)$. We get that the ordinals less than $g(\alpha)$ are those that are built up from f_+ and the $f_{\alpha-}$ where $\alpha- < \alpha$. We can write them as finite sequences of $+$ and the ordinals $\alpha-$. As before we have a lexicographical ordering where we have as separating sign the largest ordinal in the finite sequence and $+$ is the least element. Hence

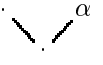
$$g(\alpha + 1) = g(\alpha)^\omega$$

We get a recursion equation for $g(\alpha)$. We have the same recursion equation in

$$\omega^{(\omega^{\alpha+1})} = (\omega^{\omega^\alpha})^\omega$$

and both function behaves continuously at limits and have the same start. Therefore

Theorem 11. *For all* $\alpha \in \mathcal{T}_2$



$= \omega^{(\omega^\alpha)}$

The ordinal $g(\alpha)$ majorises all ordinals built up from f_+ and $f_{\alpha-}$ where $\alpha- < \alpha$. Hence

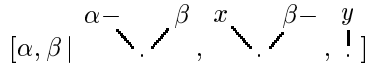
Theorem 12.  $= \epsilon_0$

5

Consider now the function

$= \psi(\alpha, \beta)$

It has as fundamental set



where $\alpha-$ runs over the ordinals $< \alpha$, $\beta-$ runs over the ordinals $< \beta$ and x and y are variables indicating functions. The fundamental set gives a recursion equation for the function $\psi(\alpha, \beta)$. We note that it is the least ordinal such that

- $\psi(\alpha, \beta) > \psi(\alpha-, \beta)$
- $\psi(\alpha, \beta) > \alpha$
- $\psi(\alpha, \beta) > \beta$
- $\psi(\alpha, \beta)$ is a limit ordinal
- $\psi(\alpha, \beta)$ is a fixpoint for all $x \mapsto \psi(x, \beta-)$

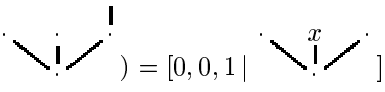
But this is the fix point free Veblen hierarchy starting with $x \mapsto \omega \cdot x$. This hierarchy does not grow so fast. The first critical point is ϵ_0.

Consider now the function

We first observe that $\phi(\alpha, 0)$ gives a fixpoint free enumeration of the ϵ-numbers. This is immediately seen from its fundamental set. But then we get $\phi(\alpha, \beta)$ is the fixpoint free Veblen hierarchy starting with the ϵ-numbers. This is almost the same as the usual Veblen hierarchy [2] where the start is the function ω^α enumerating the multiplicative principal numbers. The first critical number is of course the fixpoint of $x \mapsto \phi(0, x)$ which gives

Theorem 13. 

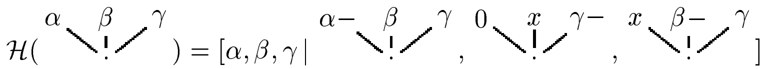
Its elementary fundamental set is



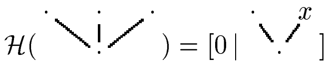
To go further along this line we need the Veblen hierarchy generalized to n-ary functions as defined by Kurt Schütte [4] based on work by Wilhelm Ackermann[1]. Assume we have the ordinary binary Veblen function $\phi(\alpha, \beta)$. We get the ternary Veblen function by

$$\begin{aligned} \phi(\alpha, \beta, 0) &= \phi(\alpha, \beta) \\ \gamma > 0 : \phi(\alpha, 0, \gamma) &= \text{ the } \alpha \text{ common fixpoint of all } \phi(0, x, \gamma-) \\ \beta, \gamma > 0 : \phi(\alpha, \beta, \gamma) &= \text{ the } \alpha \text{ common fixpoint of all } \phi(x, \beta-, \gamma) \end{aligned}$$

And we recognize these cases in the elementary fundamental set (for the case when one of the subtrees is different from 0)



and



So here we have a fixpoint free version of the ternary Veblen hierarchy.

Theorem 14. *The ordinal*  *is the ordinal of the ternary Veblen hierarchy.*

The same argument is lifted up.

Theorem 15. *The ordinal of the trees with n-ary branching where $n > 2$ is the ordinal of the n-ary Veblen hierarchy.*

Theorem 16. *The ordinal of all finite trees is the small Veblen ordinal $\phi_{\Omega^\omega}0$.*

The small Veblen ordinal $\phi_{\Omega^\omega}0$ is the ordinal of finitary Veblen functions. It is also the ordinal connected with Kruskals theorem. [3]

References

1. Wilhelm Ackermann. Konstruktiver Aufbau eines Abschnitts der zweiten Cantorsche Zahlenklasse. *Mathematische Zeitschrift*, 53:403–413, 1951.
2. Wolfram Pohlers. *Proof Theory : An Introduction.* Number 1407 in Lecture Notes in Mathematics. Springer, Berlin, 1989.
3. Michael Rathjen and Andreas Weiermann. Proof-theoretic investigations on Kruskal's theorem. *Annals of Pure and Applied Logic*, 60:49–88, 1993.
4. Kurt Schütte. Kennzeichnung von Ordinalzahlen durch rekursiv erklärte Funktionen. *Mathematische Annalen*, 127:15–32, 1954.

On the Problems of Definability in the Enumeration Degrees

Iskander Sh. Kalimullin*

Kazan State University,
Kazan, Russia
Iskander.Kalimullin@ksu.ru

Informally, the enumeration degree $\deg_e(A)$ (or e-degree) of a set A of natural numbers is a class of sets which have the same enumeration difficulty (see [2] for the exact definition).

We say that an e-degree is *total* if it contains the set $X \oplus \overline{X}$ for some $X \subseteq \omega$. The class **TOT** of total e-degrees forms a subsemilattice in the upper semilattice $\mathbf{D}_e$ of all e-degrees and are isomorphic to the Turing degrees under the natural embedding $\iota : \deg_T(X) \mapsto \deg_e(X \oplus \overline{X})$.

There are several open problems of definabilty in the enumeration degrees. In particular, at 1967 Rogers [2] posed the problem on definabilty of the class of total degrees in the upper semilattice $\mathbf{D}_e$. It is interesting to note that if we replace in this problem the class **TOT** by the class $\mathbf{TOT}(\geq \mathbf{0}'_e)$ of all total enumeration enumeration degrees above the e-degree $\mathbf{0}'_e$ of the complement of a creative set then the obtained problem has a positive solution.

Theorem 1. *([1]) The class* $\mathbf{TOT}(\geq \mathbf{0}'_e)$ *is definable by the formula*

$$\Psi(\mathbf{x}) = (\exists \mathbf{a} > \mathbf{0}_e)(\exists \mathbf{b} > \mathbf{0}_e)[\mathbf{x} = \mathbf{a} \cup \mathbf{b} \ \&\ (\forall \mathbf{z})[(\mathbf{a} \cup \mathbf{z}) \cap (\mathbf{b} \cup \mathbf{z}) = \mathbf{z}]] \ \&$$
$$(\forall \mathbf{c} > \mathbf{0}_e)(\forall \mathbf{d} > \mathbf{0}_e)(\forall \mathbf{e} > \mathbf{0}_e)[(\forall \mathbf{z})[(\mathbf{c} \cup \mathbf{z}) \cap (\mathbf{d} \cup \mathbf{z}) = (\mathbf{c} \cup \mathbf{z}) \cap (\mathbf{e} \cup \mathbf{z}) = \mathbf{z}] \rightarrow$$
$$\mathbf{d} \cup \mathbf{x} = \mathbf{e} \cup \mathbf{x}].$$

The proof of this theorem essentially uses properties of the e-degrees from the next definition.

Definition 2. *For an e-degree* $\mathbf{u}$ *a pair of e-degrees* $\mathbf{u}$ *and* $\mathbf{b}$ *is* $\mathbf{u}$*-e-ideal if there are sets* $A \in \mathbf{a}$, $B \in \mathbf{b}$ *and* U, $\deg_e(U) \leq \mathbf{u}$, *such that* $A \times B \subseteq U$ *and* $\overline{A} \times \overline{B} \subseteq \overline{U}$.

In fact, this notion is definable in the enumeration degrees ([1]): a pair of e-degrees $\mathbf{a}$ and $\mathbf{b}$ is $\mathbf{u}$-e-ideal iff $(\forall \mathbf{z} \geq \mathbf{u})[(\mathbf{a} \cup \mathbf{z}) \cap (\mathbf{b} \cup \mathbf{z}) = \mathbf{z}]$. Thus, the first conjunct of the formula $\Psi(\mathbf{x})$ from Theorem 1 simply says that $\mathbf{x}$ splits into a $\mathbf{0}_e$-e-ideal pair of e-degrees $\mathbf{a} > \mathbf{0}_e$ and $\mathbf{b} > \mathbf{0}_e$ (the second conjunct is the order-theoretical definition of the predicate $\mathbf{x} \geq \mathbf{0}'_e$).

By results from [1] each half of a $\mathbf{u}$-e-ideal pair $\mathbf{a} > \mathbf{u}$ and $\mathbf{b} > \mathbf{u}$ is $\mathbf{u}$-quasiminimal and hence can not be total (a e-degree $\mathbf{x} > \mathbf{u}$ is $\mathbf{u}$-quasiminimal

* Supported by grants of RFBR 02-01-00169.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 221–222, 2005.

if $\mathbf{y} \leq \mathbf{u}$ for all total e-degrees $\mathbf{y} \leq \mathbf{x}$). Thus the class defined by the formula $(\exists \mathbf{a} > \mathbf{0}_e)(\forall \mathbf{z})[(\mathbf{a} \cup \mathbf{z}) \cap (\mathbf{x} \cup \mathbf{z}) = \mathbf{z}]$ is a definable subclass of the non-total degrees. Of course, this class is not equal to the class of all non-total e-degrees since not every non-total e-degree is $\mathbf{0}_e$-quasiminimal.

The following theorem shows that at least each nontotal e-degree below $\mathbf{0}'$ is $\mathbf{u}$-quasiminimal for some $\mathbf{u}$ and this can help to solve the problem for the total degrees below $\mathbf{0}'_e$.

Theorem 3. *For every non-total e-degree* $\mathbf{x} \leq \mathbf{0}'_e$ *there are an e-degree* $\mathbf{u} < \mathbf{x}$ *such that* $\mathbf{x}$ *is* $\mathbf{u}$*-quasiminimal.*

It is not known yet does this theorem holds for all e-degrees $\mathbf{x}$. If yes and, moreover, the e-degree $\mathbf{u}$ can be chosen such that for some $\mathbf{a} > \mathbf{u}$ the pair of e-degrees $\mathbf{a}$ and $\mathbf{x}$ is $\mathbf{u}$-e-ideal, then we get a definability of the total e-degrees

More generally, we can consider the following definable class containing the class of total e-degrees.

Definition 4. *An e-degree* $\mathbf{x}$ *is pseudo-total if for all e-degrees* $\mathbf{u}$ *and* $\mathbf{a}$ *such that the pair of e-degrees* $\mathbf{a}$ *and* $\mathbf{x}$ *is* $\mathbf{u}$*-e-ideal we have either* $\mathbf{a} \leq \mathbf{u}$*, or* $\mathbf{x} \leq \mathbf{u}$*.*

It is easy to check that this class is closed under sups like the class of total e-degree and this presents an indirect argument in favour of coincidence of these two classes.

In the remaining part of the talk we will study definability properties for some other classes of enumeration degrees.

References

1. Kalimullin, I. Sh. *Definability of the jump operator in the enumeration degrees*, J. Math. Logic., vol. 3, no. 2 (2003), 257–267.
2. H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.

Computing a Model of Set Theory

Peter Koepke

Mathematisches Institut,
Rheinische Friedrich-Wilhelms-Universität Bonn,
Beringstraße 1,
53113 Bonn, Germany

Abstract. We define the notion of *ordinal computability* by generalizing standard TURING computability on tapes of length ω to computations on tapes of arbitrary ordinal length. The generalized TURING machine is able to compute a recursive bounded truth predicate on the ordinals. The class of sets of ordinals which can be read off the truth predicate satisfies a natural theory SO. SO is the theory of the sets of ordinals in a model of the ZERMELO-FRAENKEL axioms ZFC. Hence a set of ordinals is ordinal computable from ordinal parameters if and only if it is an element of GÖDEL's constructible universe L.

1 Introduction

A standard TURING computation may be visualized as a time-like sequence of elementary *read-write-move* operations carried out by one or more "heads" on "tapes". The sequence of actions is determined by the initial tape contents and by a finite TURING *program*. We may assume that TURING machines act on tapes whose cells are indexed by the set ω $(= \mathbb{N})$ of *natural numbers* $0, 1, \ldots$ and contain 0's or 1's.

		SPACE									
		0	1	2	3	4	5	6	7	...	...
	0	<u>1</u>	0	0	1	1	1	0	0	0	0
	1	0	<u>0</u>	0	1	1	1	0	0		
T	2	0	0	<u>0</u>	1	1	1	0	0		
I	3	0	<u>0</u>	1	1	1	1	0	0		
M	4	0	1	<u>1</u>	1	1	1	0	0		
E	:										
	n	1	1	1	1	<u>0</u>	1	1	1		
	$n+1$	1	1	1	1	1	<u>1</u>	1	1		
	:										

A standard TURING *computation. Head positions are indicated by* <u>underlining</u>.

An obvious generalization from the perspective of transfinite ordinal theory is to extend TURING calculations to tapes whose cells are indexed by the class Ord

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 223–232, 2005.

of all *ordinal numbers*. At *limit* ordinals we define the tape contents, program states and head positions by appropriate limit operations which may be viewed as *inferior limits*.

		Ordinal Space															
		0	1	2	3	4	5	6	7	...	...	ω	...	α	...	κ	...
O	0	$\underline{1}$	1	0	1	0	0	1	1	...	...	1	...	1	0	0	0
r	1	0	$\underline{1}$	0	1	0	0	1	1			1					
d	2	0	0	$\underline{0}$	1	0	0	1	1			1					
i	3	0	$\underline{0}$	0	1	0	0	1	1			1					
n	4	0	0	$\underline{0}$	0	0	0	1	1			1					
a	:																
l	n	1	1	1	1	$\underline{0}$	1	0	1			1					
	$n+1$	1	1	1	1	1	$\underline{1}$	0	1			1					
T	⋮	⋮	⋮	⋮	⋮	⋮											
i	ω	0	0	1	0	0	0	1	1	...	...	$\underline{1}$					
m	$\omega+1$	$\underline{0}$	0	1	0	0	0	1	1			0					
e	:																
	θ	1	0	0	1	1	1	1	0	...	...	...	...	$\underline{0}$	...	...	...
⋮	⋮			⋮			⋮			⋮	⋮						
	⋮																

An ordinal computation.

This notion of *ordinal computability* obviously extends TURING computability. By the CHURCH-TURING thesis many operations on natural numbers are ordinal computable. The *ordinal arithmetical* operations (addition, multiplication, exponentiation) and other basic operations on ordinals are also ordinal computable.

Using GÖDEL's pairing function $G : \mathrm{Ord} \times \mathrm{Ord} \to \mathrm{Ord}$ one can view each ordinal α as a first-order sentence with constant symbols for ordinals $< \alpha$. One can then define a recursive truth predicate $T : \mathrm{Ord} \to \{0, 1\}$ by:

$$T(G(\alpha, \beta)) = 1 \text{ iff } (\alpha, <, G \cap \alpha^3, T \restriction \alpha) \vDash \alpha[\beta].$$

This recursion can be carried out by an ordinal TURING machine. For ordinals μ and α the function T codes the set

$$T(\mu, \alpha) = \{\beta < \mu | T(G(\alpha, \beta)) = 1\}.$$

The class

$$\mathcal{S} = \{T(\mu, \alpha) | \mu, \alpha \in \mathrm{Ord}\}$$

is the class of sets of ordinals of a transitive proper class model of set theory. Since the ordinal TURING computations can be carried out in the $\subseteq$-smallest such model, namely GÖDEL's model L of *constructible sets*, we obtain our main result characterizing ordinal computability:

Theorem 1. *A set $x \subseteq \mathrm{Ord}$ is ordinal computable from finitely many ordinal parameters if and only if $x \in L$.*

This theorem may be viewed as an analogue of the CHURCH-TURING thesis: ordinal computability defines a natural and absolute class of sets, and it is stable with respect to technical variations in its definition. This work was inspired by the *infinite time* TURING machines introduced by JOEL D. HAMKINS, JEFF KIDDER and ANDY LEWIS [3]. A more comprehensive technical account of ordinal computability, also indicating set theoretical applications is given in [5].

2 Ordinal Turing Machines

Ordinal TURING machines are defined in close analogy with standard TURING machines. At *successor ordinals* we use standard TURING steps. The behaviour at *limit ordinals* will be defined by simple limit operations.

Definition 2.

a) *A **command** is a 5-tuple $C{=}(s, c, c', m, s')$ where $s, s' \in \omega$ and $c, c', m \in \{0, 1\}$; the natural number s is the **state** of the command C. The intention of the command C is that if the machine is in state s and reads the symbol c under its read-write head, then it writes the symbol c', moves the head left if $m = 0$ or right if $m = 1$, and goes into state s'. States correspond to the "line numbers" of some programming languages.*
b) *A **program** is a finite set P of commands satisfying the following structural conditions:*
   i. *If $(s, c, c', m, s') \in P$ then there is $(s, d, d', n, t') \in P$ with $c \neq d$; thus in state s the machine can react to reading a "0" as well as to reading a "1".*
   ii. *If $(s, c, c', m, s') \in P$ and $(s, c, c'', m', s'') \in P$ then $c' = c'', m = m', s' = s''$; this means that the course of the computation is completely determined by the sequence of program states and the initial cell contents.*
c) *For a program P let*

$$\mathrm{states}(P) = \{s | (s, c, c', m, s') \in P\}$$

be the set of program states.

Definition 3. *Let P be a program. A triple*

$$S : \theta \to \omega, H : \theta \to \mathrm{Ord}, T : \theta \to ({}^{\mathrm{Ord}}2)$$

*is an **ordinal computation** by P if the following hold:*

a) *θ is a successor ordinal or $\theta = \mathrm{Ord}$; θ is the **length** of the computation.*
b) *$S(0) = H(0) = 0$; the machine starts in state 0 with head position 0.*

c) *If $t < \theta$ and $S(t) \notin \mathrm{state}(P)$ then $\theta = t+1$; the machine* ***stops*** *if the machine state is not a program state of P.*

d) *If $t < \theta$ and $S(t) \in \mathrm{state}(P)$ then $t+1 < \theta$; choose the unique command $(s, c, c', m, s') \in P$ with $S(t) = s$ and $T(t)_{H(t)} = c$; this command is executed as follows:*

$$T(t+1)_\xi = \begin{cases} c', \text{ if } \xi = H(t); \\ T(t)_\xi \text{ , else;} \end{cases}$$
$$S(t+1) = s';$$
$$H(t+1) = \begin{cases} H(t)+1, \text{ if } m = 1; \\ H(t)-1, \text{ if } m = 0 \text{ and } H(t) \text{ is a successor ordinal;} \\ 0, \text{ else.} \end{cases}$$

e) *If $t < \theta$ is a limit ordinal, the machine constellation at t is determined by taking inferior limits:*

$$\forall \xi \in \mathrm{Ord}\, T(t)_\xi = \liminf_{r \to t} T(r)_\xi;$$
$$S(t) = \liminf_{r \to t} S(r);$$
$$H(t) = \liminf_{s \to t, S(s) = S(t)} H(s).$$

The computation is obviously recursively determined by the initial tape contents $T(0)$ and the program P. We call it the ***ordinal computation by*** *P* ***with input*** *$T(0)$. If the computation stops, $\theta = \beta + 1$ is a successor ordinal and $T(\beta)$ is the final tape content. In this case we say that P* ***computes*** *$T(\beta)$ from $T(0)$ and write $P : T(0) \mapsto T(\beta)$.*

This interpretation of programs yields associated notions of computability.

Definition 4. *A partial function $F :^{\mathrm{Ord}} 2 \rightharpoonup^{\mathrm{Ord}} 2$ is* ***ordinal computable*** *if there is a program P such that $P : T \mapsto F(T)$ for every $T \in \mathrm{dom}(F)$.*

By coding, the notion of ordinal computability can be extended to other domains. We can e.g. *code* an ordinal $\delta \in \mathrm{Ord}$ by the characteristic function $\chi_{\{\delta\}} : \mathrm{Ord} \to 2$, $\chi_{\{\delta\}}(\xi) = 1$ iff $\xi = \delta$, and define:

Definition 5. *A partial function $F : \mathrm{Ord} \rightharpoonup \mathrm{Ord}$ is* ***ordinal computable*** *if the function $\chi_{\{\delta\}} \mapsto \chi_{\{F(\delta)\}}$ is ordinal computable.*

We also consider computations involving finitely many ordinal *parameters*.

Definition 6. *A subset $x \subseteq \mathrm{Ord}$ is* ***ordinal computable from finitely many ordinal parameters*** *if there a finite subset $z \subseteq \mathrm{Ord}$ and a program P such that $P : \chi_z \mapsto \chi_x$.*

3 Ordinal Algorithms

The intended computations will deal with ordinals and sequences of ordinals. The simplest way of representing the ordinal $\alpha \in \mathrm{Ord}$ in an ordinal machine is by a tape whose content is the characteristic function of $\{\alpha\}$:

$$\chi_{\{\alpha\}} : \mathrm{Ord} \to 2,\ \chi_{\{\alpha\}}(\xi) = 1 \text{ iff } \xi = \alpha.$$

A basic task is to *find* or *identify* this ordinal α: initially the head is in position 0, it then moves to the right until it stops exactly at position α. This is achieved by the following program:

$$P = \{(0,0,0,1,0),(0,1,1,1,1),(1,0,0,0,2),(1,1,1,0,2)\}.$$

The program is in state 0 until it reads a 1, then it goes one cell to the right, one cell to the left, and stops because 2 is not a program state. Informally the algorithm may be written as

```
Find_Ordinal:
0   if head = 1 then STOP otherwise moveright, go to 0
```

It will be convenient to work with several tapes side-by-side instead of just one. One can simulate an n-tape machine on a 1-tape machine. The contents $(T^i_\xi | \xi \in \mathrm{Ord})$ of the i-th tape are successively written into the cells of tape T indexed by ordinals $2n\xi + 2i$:

$$T_{2n\xi+2i} = T^i_\xi.$$

The head position H^i on the i-th tape is simulated by writing 1's into an initial segment of length H^i of cells with indices of the form $2n\xi + 2i + 1$:

$$T_{2n\xi+2i+1} = \begin{cases} 1, \text{ if } \xi < H^i; \\ 0, \text{ else.} \end{cases}$$

So two tapes with contents $a_0a_1a_2a_3a_4\ldots$ and $b_0b_1b_2b_3b_4\ldots$ and head positions 3 and 1 respectively are coded as

$$T = a_01b_01a_11b_10a_21b_20a_30b_30a_40b_40\ldots\ldots.$$

There are canonical but tedious translations from programs for n-tape machines into corresponding programs for 1-tape machines. One can assume that one or more of the machine tapes serve as standard TURING tapes on which ordinary TURING recursive functions are computed.

Basic operations on ordinals are ordinal computable. The GÖDEL pairing function for ordinals is defined recursively by

$$\begin{aligned} G(\alpha,\beta) = \{G(\alpha',\beta') | \ & \max(\alpha',\beta') < \max(\alpha,\beta) \text{ or} \\ & (\max(\alpha',\beta') = \max(\alpha,\beta) \text{ and } \alpha' < \alpha) \text{ or} \\ & (\max(\alpha',\beta') = \max(\alpha,\beta) \text{ and } \alpha' = \alpha \text{ and } \beta' < \beta)\}. \end{aligned}$$

We sketch an algorithm for computing $\gamma = G(\alpha, \beta)$ which can be implemented straightforwardly on a TURING machine with several tapes, each holding one of the variables.

```
Goedel_Pairing:
0  alpha':=0
1  beta':=0
2  eta:=0
3  flag:=0
3  gamma:=0
4  if alpha=alpha' and beta=beta' then print gamma, stop fi
5  if alpha'=eta and and beta'=eta and flag=0 then
      alpha'=0, flag:=1, go to 4 fi
6  if alpha'=eta and and beta'=eta and flag=1 then
      eta:=eta+1, alpha'=eta, beta'=0, gamma:=gamma+1, go to 4 fi
7  if beta'<eta and flag=0 then
      beta':=beta'+1, gamma:=gamma+1, go to 4 fi
8  if alpha'<eta and flag=1 then
      alpha':=alpha'+1, gamma:=gamma+1, go to 4 fi
```

Observe that at limit times this algorithm will always cycle to command 4. The inverse functions G_0 and G_1 satisfying

$$\forall \gamma \gamma = G(G_0(\gamma), G_1(\gamma))$$

are also ordinal computable. To compute $G_0(\gamma)$ compute $G(\alpha, \beta)$ for $\alpha, \beta < \gamma$ until you find α, β with $G(\alpha, \beta) = \gamma$; then set $G_0(\gamma) = \alpha$.

4 A Recursive Truth Predicate

The GÖDEL pairing function G allows to code a finite sequence $\alpha_0, \ldots, \alpha_{n-1}$ of ordinals as a single ordinal

$$\alpha = G(\ldots G(G(\alpha_0, \alpha_1), \alpha_2) \ldots).$$

The usual operations on finite sequences like concatenation, cutting at a certain length, substitution, etc. are ordinal computable using the GÖDEL functions G, G_0, G_1. We can thus code terms and formulas of a first-order language by single ordinals in an ordinal computable way.

We introduce a language L_T suitable for structures of the form

$$(\alpha, <, G \cap \alpha^3, f)$$

where $G \cap \alpha^3$ is viewed as a ternary relation and $f : \alpha \to \alpha$ is a unary function. The language has variables $v_n = G(0, n)$ for $n < \omega$ and constant symbols $c_\xi = G(1, \xi)$ for $\xi \in \mathrm{Ord}$; the symbol c_ξ will be interpreted as the ordinal ξ. Terms are defined recursively: variables and constant symbols are terms; if t is a Term then $G(2, t)$ is a term as well which stands for $f(t)$. Atomic formulas are of the forms

- $G(3, G(t_1, t_2))$ where t_1, t_2 are terms; this stands for the equality $t_1 = t_2$;
- $G(4, G(t_1, t_2))$ where t_1, t_2 are terms; this stands for the inequality $t_1 < t_2$;
- $G(5, G(G(t_1, t_2), t_3))$ where t_1, t_2, t_3 are terms; this stands for the relation $t_3 = G(t_1, t_2)$.

L_T-Formulas are defined recursively: atomic formulas are formulas; if φ and ψ are formulas then the following are formulas as well:

- $G(6, \varphi)$; this stands for the negation $\neg\varphi$;
- $G(7, G(\varphi, \psi))$; this stands for the conjunction $(\varphi \wedge \psi)$;
- $G(8, G(v_n, \varphi))$ where v_n is a variable; this stands for the existential quantification $\exists v_n \varphi$.

Then the satisfaction relation

$$(\alpha, <, G \cap \alpha^3, f) \vDash \varphi[b]$$

for φ an L_T-formula and b an assignment of values in α can be defined as usual. If the function f is ordinal computable then this property is ordinal computable, since the recursive TARSKI truth definition can be carried out by an ordinal TURING machine.

Define the truth predicate $T : \mathrm{Ord} \to \{0, 1\}$ recursively by

$$T(\alpha) = 1 \text{ iff } (\alpha, <, G \cap \alpha^3, T \restriction \alpha) \vDash G_0(\alpha)[G_1(\alpha)].$$

The assignments $\alpha \mapsto T(\alpha)$ can be enumerated successively by an ordinal TURING machine. Hence T is ordinal computable. We shall see shortly that T is a strong predicate which codes a model of set theory.

5 The Theory SO of Sets of Ordinals

It is well-known that a model of Zermelo-Fraenkel set theory with the axiom of choice (ZFC) is determined by its sets of ordinals [4], Theorem 13.28. We define a natural theory SO which axiomatizes the sets of ordinals in a model of ZFC. This theory was first defined and examined in [6].

The theory SO is two-sorted: *ordinals* are taken as atomic objects, the other sort corresponds to *sets of ordinals*. Let L_{SO} be the language

$$L_{\mathrm{SO}} := \{\mathrm{Ord}, \mathrm{SOrd}, <, =, \in, g\}$$

where Ord and SOrd are unary predicate symbols, $<$, $=$ and $\in$ are binary predicatesymbols and g is a two-place function. To simplify notation, we use lower case greek letters to range over elements of Ord and lower case roman letters to range over elements of SOrd.

1. Well-ordering axiom:
   $\forall\alpha, \beta, \gamma(\neg\alpha < \alpha \wedge (\alpha < \beta \wedge \beta < \gamma \to \alpha < \gamma) \wedge$
   $(\alpha < \beta \vee \alpha = \beta \vee \beta < \alpha)) \wedge$
   $\forall a(\exists\alpha(\alpha \in a) \to \exists\alpha(\alpha \in a \wedge \forall\beta(\beta < \alpha \to \neg\beta \in a)))$;

2. Axiom of infinity (existence of a limit ordinal):
   $\exists\alpha(\exists\beta(\beta<\alpha)\wedge\forall\beta(\beta<\alpha\rightarrow\exists\gamma(\beta<\gamma\wedge\gamma<\alpha)))$;
3. Axiom of extensionality: $\forall a,b(\forall\alpha(\alpha\in a\leftrightarrow\alpha\in b)\rightarrow a=b)$;
4. Initial segment axiom: $\forall\alpha\exists a\forall\beta(\beta<\alpha\leftrightarrow\beta\in a)$;
5. Boundedness axiom: $\forall a\exists\alpha\forall\beta(\beta\in a\rightarrow\beta<\alpha)$;
6. Pairing axiom (Gödel Pairing Function):
   $\forall\alpha,\beta,\gamma(g(\beta,\gamma)\leq\alpha\leftrightarrow\forall\delta,\epsilon((\delta,\epsilon)<^*(\beta,\gamma)\rightarrow g(\delta,\epsilon)<\alpha))$.
   Here $(\alpha,\beta)<^*(\gamma,\delta)$ stands for
   $\exists\eta,\theta(\eta=\max(\alpha,\beta)\wedge\theta=\max(\gamma,\delta)\wedge(\eta<\theta\vee$
   $(\eta=\theta\wedge\alpha<\gamma)\vee(\eta=\theta\wedge\alpha=\gamma\wedge\beta<\delta)))$,
   where $\gamma=\max(\alpha,\beta)$ abbreviates $(\alpha>\beta\wedge\gamma=\alpha)\vee(\alpha\leq\beta\wedge\gamma=\beta)$;
7. g is onto: $\forall\alpha\exists\beta,\gamma(\alpha=g(\beta,\gamma))$;
8. Axiom schema of separation: For all L_{SO}-formulae $\phi(\alpha,P_1,\ldots,P_n)$ postulate:
   $\forall P_1,\ldots,P_n\forall a\exists b\forall\alpha(\alpha\in b\leftrightarrow\alpha\in a\wedge\phi(\alpha,P_1,\ldots,P_n))$;
9. Axiom schema of replacement: For all L_{SO}-formulae $\phi(\alpha,\beta,P_1,\ldots,P_n)$ postulate:
   $\forall P_1,\ldots,P_n(\forall\xi,\zeta_1,\zeta_2(\phi(\xi,\zeta_1,P_1,\ldots,P_n)\wedge\phi(\xi,\zeta_2,P_1,\ldots,P_n)\rightarrow\zeta_1=\zeta_2)\rightarrow$
   $\forall a\exists b\forall\zeta(\zeta\in b\leftrightarrow\exists\xi\in a\phi(\xi,\zeta,P_1,\ldots,P_n)))$;
10. Powerset axiom:
    $\forall a\exists b(\forall z(\exists\alpha(\alpha\in z)\wedge\forall\alpha(\alpha\in z\rightarrow\alpha\in a)\rightarrow\exists^{=1}\xi\forall\beta(\beta\in z\leftrightarrow g(\beta,\xi)\in b)))$.

6 T Codes a Model of SO

The truth predicate T contains information about a large class of sets of ordinals.

Definition 7. *For ordinals μ and α define*

$$T(\mu,\alpha)=\{\beta<\mu|T(G(\alpha,\beta))=1\}.$$

Set

$$\mathcal{S}=\{T(\mu,\alpha)|\mu,\alpha\in\mathrm{Ord}\}.$$

Theorem 8. $(\mathrm{Ord},\mathcal{S},<,=,\in,G)$ *is a model of the theory* SO.

Proof. The axioms (1)-(7) are obvious. The proofs of axiom schemas (8) and (9) rest on a Levy-type reflection principle. For $\theta\in\mathrm{Ord}$ define

$$\mathcal{S}_\theta=\{T(\mu,\alpha)|\mu,\alpha\in\theta\}.$$

Then for any L_{SO}-formula $\varphi(v_0,\ldots,v_{n-1})$ and $\eta\in\mathrm{Ord}$ there is some limit ordinal $\theta>\eta$ such that

$$\forall\xi_0,\ldots,\xi_{n-1}\in\theta((\mathrm{Ord},\mathcal{S},<,=,\in,G)\vDash\varphi[\xi_0,\ldots,\xi_{n-1}]\text{ iff}$$

$$(\theta,\mathcal{S}_\theta,<,=,\in,G)\vDash\varphi[\xi_0,\ldots,\xi_{n-1}]).$$

Since all elements of $\mathcal{S}_\theta$ can be defined from the truth function T and ordinals $< \theta$, the right-hand side can be evaluated in the structure $(\theta, <, G \cap \theta^3, T)$ by an L_T-formula φ^*which can be recursively computed from φ. Hence

$$\forall \xi_0, \ldots, \xi_{n-1} \in \theta((\mathrm{Ord}, \mathcal{S}, <, =, \in, G) \vDash \varphi[\xi_0, \ldots, \xi_{n-1}] \text{ iff}$$

$$(\theta, <, G \cap \theta^3, T) \vDash \varphi^*[\xi_0, \ldots, \xi_{n-1}]).$$

So sets witnessing axioms (8) and (9) can be defined over $(\theta, <, G \cap \theta^3, T)$ and are thus elements of $\mathcal{S}$.

The powerset axiom can be shown by a similar reflection argument.

7 Ordinal Computability Corresponds to Constructibility

Kurt Gödel [2] defined the inner model L of *constructible sets* as the union of a hierarchy of levels L_α:

$$L = \bigcup_{\alpha \in \mathrm{Ord}} L_\alpha$$

where the hierarchy is defined by: $L_0 = \emptyset$, $L_\delta = \bigcup_{\alpha<\delta} L_\alpha$ for limit ordinals δ, and $L_{\alpha+1}$ =the set of all sets which are first-order definable in the structure $(L_\alpha, \in)$. The model L is the $\subseteq$-smallest inner model of set theory. The standard reference for the theory of the model L is the monograph [1].

The following main result provides a characterization of ordinal computability which does not depend on any specific machine model or coding of language:

Theorem 9. *A set x of ordinals is ordinal computable from a finite set of ordinal parameters if and only if it is an element of the constructible universe L.*

Proof. Let $x \subseteq \mathrm{Ord}$ be ordinal computable by the program P from the finite set $\{\alpha_0, \ldots, \alpha_{k-1}\}$ of ordinal parameters: $P : \chi_{\{\alpha_0, \ldots, \alpha_{k-1}\}} \mapsto \chi_x$. By the simple nature of the computation procedure the same computation can be carried out inside the inner model L:

$$(L, \in) \vDash P : \chi_{\{\alpha_0, \ldots, \alpha_{k-1}\}} \mapsto \chi_x.$$

Hence $\chi_X \in L$ and $x \in L$.

Conversely consider $x \in L$. Since $(\mathrm{Ord}, \mathcal{S}, <, =, \in, G)$ is a model of the theory SO there is an inner model M of set theory such that

$$\mathcal{S} = \{z \subseteq \mathrm{Ord} \,|\, z \in M\}.$$

Since L is the $\subseteq$-smallest inner model, $L \subseteq M$. Hence $x \in M$ and $x \in \mathcal{S}$. Let $x = T(\mu, \alpha)$. By the computability of the truth predicate, x is ordinal computable from the parameters μ and α.

References

1. Keith Devlin. *Constructibility*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1984.
2. Kurt Gödel. *The Consistency of the Continuum Hypothesis*, volume 3 of *Ann. of Math. Studies*. Princeton University Press, Princeton, 1940.
3. Joel David Hamkins and Andy Lewis. Infinite Time Turing Machines. *J. Symbolic Logic*, 65(2):567–604, 2000.
4. Thomas Jech. *Set Theory. The Third Millennium Edition*. Springer Monographs in Mathematics. Springer-Verlag, 2003.
5. Peter Koepke. Turing Computations on Ordinals. Submitted to the *Bulletin of Symbolic Logic*. Preprint math.LO/0502264 at the e-print archive arXiv.org.
6. Peter Koepke and Martin Koerwien. The Theory of Sets of Ordinals. Preprint math.LO/0502265 at the e-print archive arXiv.org.

Proof Mining in Functional Analysis

Ulrich Kohlenbach

Department of Mathematics,
Darmstadt University of Technology,
Schlossgartenstraße 7,
D-64289 Darmstadt, Germany
kohlenbach@mathematik.tu-darmstadt.de
http://www.mathematik.tu-darmstadt.de/∼kohlenbach/

In recent years (though much influenced by writings of G. Kreisel going back to the 50's as well as subsequent work by H. Luckhardt and others) an applied form of proof theory systematically evolved which is also called 'Proof Mining' ([10], see also [1]). It is concerned with transformations of prima facie ineffective proofs into proofs from which certain quantitative computational information as well as new qualitative information can be read off which was not visible beforehand. We will present general logical metatheorems ([3, 6]) which guarantee a priorily for large classes of theorems and proofs in analysis the extractability of effective bounds which are independent from parameters in general classes of metric, hyperbolic and normed spaces if certain local boundedness conditions are satisfied. Unless separability assumptions on the spaces involved are used in a given proof, the independence results from parameters only need metric bounds but no compactness. Obviously, certain restrictions on the logical form of the theorems to be proved as well as on the axioms to be used in the proofs are necessary. These restrictions in turn depend on the language of the formal systems used as well as the representation of the relevant mathematical objects such as general function spaces. The correctness of the results, moreover, depends in subtle ways on the amount of extensionality properties used in the proof which has a direct analytic counterpart in terms of uniform continuity conditions.

We discuss a number of new qualitative existence results in the area of nonlinear functional analysis which follow from the metatheorems but so far did not have a functional analytic proof. Applying the extraction algorithm provided by the proofs of the metatheorems to these results then yields explicit quantitative versions and at the same time direct proofs which no longer rely on the logical metatheorems themselves ([2, 4, 5, 8, 9, 7]).

References

1. Berger, U., Buchholz, H., Schwichtenberg, H., Refined program extraction from proofs. Ann. Pure Appl. Logic **114**, pp. 3-25 (2002).
2. Gerhardy, P., A quantitative version of Kirk's fixed point theorem for asymptotic contraction. Submitted.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 233–234, 2005.

3. Gerhardy, P., Kohlenbach, U., General logical metatheorems for functional analysis. Draft 29pp. (2005).
4. Kohlenbach, U., A quantitative version of a theorem due to Borwein-Reich-Shafrir. Numer. Funct. Anal. and Optimiz. **22**, pp. 641-656 (2001).
5. Kohlenbach, U., Uniform asymptotic regularity for Mann iterates. J. Math. Anal. Appl. **279**, pp. 531-544 (2003).
6. Kohlenbach, U., Some logical metatheorems with applications in functional analysis. Trans. Amer. Math. Soc. vol. 357, no. 1, pp. 89-128 (2005).
7. Kohlenbach, U., Some computational aspects of metric fixed point theory. To appear in: Nonlinear Analysis.
8. Kohlenbach, U., Lambov, B., Bounds on iterations of asymptotically quasi-nonexpansive mappings. In: Falset, J.G., Fuster, E.L., Sims, B. (eds.), Proc. International Conference on Fixed Point Theory and Applications, Valencia 2003, pp. 143-172, Yokohama Publishers (2004)
9. Kohlenbach, U., Leuştean, L., Mann iterates of directionally nonexpansive mappings in hyperbolic spaces. Abstr. Appl. Anal. vol. 2003, no.8, pp. 449-477 (2003).
10. Kohlenbach, U., Oliva, P., Proof mining: a systematic way of analysing proofs in mathematics. Proc. Steklov Inst. Math. **242**, pp. 1-29 (2003).

Towards Computability of Higher Type Continuous Data*

Margarita Korovina[1,2] and Oleg Kudinov[2]

[1] Lehrgebiet Theoretische Informatik I,
Berechenbarkeit und Logik, Fachbereich Informatik,
FernUniversität in Hagen, Informatikzentrum,
58084 Hagen, Germany
korovina@brics.dk
http://www.brics.dk/~korovina
[2] Sobolev Institute of Mathematics,
Siberian Branch of the Russian Academy of Sciences,
4 Acad. Koptyug avenue,
630090 Novosibirsk, Russia,
Sobolev Institute of Mathematics, Novosibirsk, Russia
kud@math.nsc.ru

Abstract. This paper extends the logical approach to computable analysis via Σ–definability to higher type continuous data such as functionals and operators. We employ definability theory to introduce computability of functionals from arbitrary domain to the real numbers. We show how this concept works in particular cases.

1 Introduction

In this paper we address the problem of computability of higher type continuous data such as real-valued functionals. This topic is motivated by problems from Physics and Engineering concerning infinite computational processes evaluating on continuous data types. Formalisation of such problems involve abstract mathematical objects of complex structures such as real numbers and real-valued functions. Such continuous data is usually represented by approximations to them [15]. The main difficulty here is that different representations lead to different, and often non-equivalent, concepts of computability. In this case it is desirable to have a notion of computability of higher type objects which does not rely upon representations. It turns out that such a notion is difficult to give even for particular cases such as real-valued functions and functionals.

* The authors would like to thank Klaus Weihrauch and Konstantin Korovin for useful discussions.
The first author was supported by the DFG grant N: We 843/17-1 "Berechenbare Analysis".
The first and the second authors were partially supported by the DFG grant N:436RUS113/638, Grant Scientific School N:2112.2003.1.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 235–241, 2005.

In this paper we extend the logical approach to computability on the reals, which was first proposed in [6], to deal with functionals from arbitrary structures to the reals. The main results of this paper are as follows. We introduce a notion of computability of higher type functionals defined on arbitrary structures, called majorant-computability. This notion is based on logical definability in extensions of the structures by hereditarily finite sets. One of the main features of this notion is that it is independent from concrete representations of the elements of structures. We also give a semantic characterisation of this notion of computability for some concrete structures such as continuous functions on reals. Such characterisation allows us to show that our notion of computability on such structures captures the same class of computable objects as the notion of computability from computable analysis (see [15]).

2 Basic Definitions and Notions

According to general concepts of computable analysis [15], it is natural to consider structures with *languages without equality* [9, 10]. In this paper we consider the standard model of the real numbers $\langle \mathbb{R}, 0, 1, +, \cdot, < \rangle = \langle \mathbb{R}, \sigma_0 \rangle$, denoted also by $\mathbb{R}$, and an abstract model $\langle \mathcal{F}, \sigma_{\mathcal{F}} \rangle$. We assume that the predicate $<$ and all predicates from $\sigma_{\mathcal{F}}$ occur positively in all formulas.

In order to introduce a notion of computability of functionals $f : \mathcal{F} \to \mathbb{R}$ we extend the structure $\mathcal{R} = \mathbb{R} \cup \mathcal{F}$ by the set of hereditarily finite sets $\mathrm{HF}(\mathcal{R})$ which is rich enough for information to be coded and stored. We construct the set of hereditarily finite sets, $\mathrm{HF}(\mathcal{R})$ over a structure $\mathcal{R}$, as follows:

1. $\mathrm{HF}_0(\mathcal{R}) \rightleftharpoons \mathcal{R}$,
2. $\mathrm{HF}_{n+1}(\mathcal{R}) \rightleftharpoons \mathcal{P}_\omega(\mathrm{HF}_n(\mathcal{R})) \cup \mathrm{HF}_n(\mathcal{R})$, where $n \in \omega$ and for every set B, $\mathcal{P}_\omega(B)$ is the set of all finite subsets of B.
3. $\mathrm{HF}(\mathcal{R}) = \bigcup_{m \in \omega} \mathrm{HF}_m(\mathcal{R})$.

We define $\mathbf{HF}(\mathcal{R})$ as the following model: $\mathbf{HF}(\mathcal{R}) \rightleftharpoons \langle \mathrm{HF}(\mathcal{R}), R, F, \sigma_0, \sigma_F, \emptyset, \in \rangle \rightleftharpoons \langle \mathrm{HF}(\mathcal{R}), \sigma \rangle$, where the constant $\emptyset$ stands for the empty set and the binary predicate symbol $\in$ has the set-theoretic interpretation. We also add predicate symbols R for elements of $\mathbb{R}$ and F for elements of $\mathcal{F}$.

For our convenience, we use variables subject to the following conventions:
$r, x, y, z, \alpha, \beta \ldots$ range over $\mathbb{R}$ (reals),
$f_1, f_2, g_1, g_2 \ldots$ range over $\mathcal{F}$ (elements of $\mathcal{F}$),
$K, L, M, N, S, \ldots$ range over $\mathrm{HF}(\mathcal{R})$.
We use the same letters as for variables to denote elements from the corresponding structures.

The set of Δ_0-*formulas* is the closure of the set of atomic formulas under $\wedge, \vee, \neg$, bounded quantifiers $(\exists M \in S)$ and $(\forall M \in S)$, where $(\exists M \in S)\ \Psi$ denotes $\exists M(M \in S \wedge \Psi)$, $(\forall M \in S)\ \Psi$ denotes $\forall M(M \in S \to \Psi)$ and S ranges over sets.

The set of Σ-*formulas* is the closure of the set of Δ_0-formulas under $\wedge$,$\vee$, $(\exists M \in S)$, $(\forall M \in S)$ and $\exists$, where S ranges over sets. We define Π-*formulas*

as negations of Σ-formulas. We are interested in Σ-definability of subsets on $\mathcal{R}^n$ which can be considered as a generalisation of recursive enumerability. The analogy of Σ-definable and recursive enumerable sets is based upon the following fact. If we consider the structure $\mathbf{HF} = \langle \mathrm{HF}(\emptyset), \in \rangle$ with the hereditarily finite sets over $\emptyset$ as its universe and membership as its only relation, then the Σ-definable sets are exactly the recursively enumerable sets. The notion of Σ-definability has a natural meaning also in the structure $\mathbf{HF}(\mathcal{R})$.

Definition 1. *1. A relation $B \subseteq \mathrm{HF}(\mathcal{R})^n$ is Σ-**definable**, if there exists a Σ-formula Φ such that $\bar{M} \in B \leftrightarrow \mathbf{HF}(\mathcal{R}) \models \Phi(\bar{M})$.*

In a similar way, we define the notions of *Π-definable sets.*

3 Higher Type Majorant-Computability

Now we are ready to extend the concept of majorant-computability, which was first proposed for real functions in [6], to functionals $F : \mathcal{F} \to \mathbb{R}$.

Definition 2. *A functional $F : \mathcal{F} \to \mathbb{R}$ is called* **majorant-computable** *if there exists a Σ-formula $\Phi(s, f, y)$ and a Π-formula $\Psi(s, f, y)$ such that the following conditions hold.*

1. *For all $s \in \omega$, $f \in \mathcal{F}$, the formulas $\Phi(s, f, \cdot)$ and $\Psi(s, f, \cdot)$ define nonempty intervals $< \alpha_s, \beta_s >$ and $[\delta_s, \gamma_s]$.*
2. *For all $f \in \mathcal{F}$, the sequences $\{< \alpha_s, \beta_s >\}_{s\in\omega}$ and $\{[\delta_s, \gamma_s]\}_{s\in\omega}$ decrease monotonically and $< \alpha_s, \beta_s > \subseteq [\delta_s, \gamma_s]$ for all $s \in \omega$.*
3. *For all $f \in \mathrm{dom}(F)$, $F(f) = y \leftrightarrow \bigcap_{s\in\omega} < \alpha_s, \beta_s > = \{y\} \leftrightarrow \bigcap_{s\in\omega} [\delta_s, \gamma_s] = \{y\}$ holds; for all $f \notin \mathrm{dom}(F)$, $|| \cap_{s\in\omega} [\delta_s, \gamma_s]|| > 1$.*

The formulas $\Phi(s, \cdot, \cdot)$ and $\Psi(s, \cdot, \cdot)$ define effective sequences $\{\Phi_s\}_{s\in\omega}$ and sequences $\{\Psi_s\}_{s\in\omega}$. The sequence $\{\Phi_s\}_{s\in\omega}$ is called a *sequence of Σ-approximations* for F. The sequence $\{\Psi_s\}_{s\in\omega}$ is called a *sequence of Π-approximations* for F. As we can see, the process which carries out the computation is represented by two effective procedures. These procedures produce Σ-formulas and Π-formulas which define approximations closer and closer to the result.

Below we will write $\varphi_1(f, \cdot) < \varphi_2(f, \cdot)$ if $\mathbf{HF}(\mathcal{R}) \models \varphi_1(f, y) \wedge \varphi_2(f, z) \to y < z$ for all real numbers y, z. The following theorem connects a majorant-computable functional with validity of finite formulas in the set of hereditarily finite sets, $\mathbf{HF}(\mathcal{R})$.

Theorem 3. *For every functional $F : \mathcal{F} \to \mathbb{R}$ the following assertions are equivalent:.*

1. *The functional F is majorant-computable.*
2. *There exists Σ–formulas $\varphi_1(f, y)$, $\varphi_2(f, y)$ such that $\varphi_1(f, \cdot) < \varphi_2(f, \cdot)$ and*

$$F(f) = y \leftrightarrow \forall z_1 \forall z_2 \, (\varphi_1(f, z_1) < y < \varphi_2(f, z_2)) \wedge$$
$$\{z \mid \varphi_1(f, z)\} \cup \{z \mid \varphi_2(f, z)\} = \mathbb{R} \setminus \{y\}.$$

Proof. $\rightarrow$)Let $F : \mathcal{F} \rightarrow \mathbb{R}$ be majorant-computable. By Definition 2, there exists a sequence $\{F_s\}_{s\in\omega}$ of Σ-approximations for F and a sequence $\{\Psi_s\}_{s\in\omega}$ of Π-approximations for F. Let

$$\varphi_1(f, y) \rightleftharpoons (\exists s \in \omega)\,(y \notin [\delta_s, \gamma_s] \wedge (\exists z \in < \alpha_s, \beta_s >)\,(y < z))$$

and

$$\varphi_2(f, y) \rightleftharpoons (\exists s \in \omega)\,(y \notin [\delta_s, \gamma_s] \wedge (\exists z \in < \alpha_s, \beta_s >)\,(y > z))\,.$$

By construction, φ_1 and φ_2 are the sought formulas.
$\leftarrow$) Let φ_1 and φ_2 satisfy the requirements of the theorem. Let us construct approximations in the following way:

$$\Phi_s(f, y) \rightleftharpoons \exists z \exists v\,(\varphi_1(f, z) \wedge \varphi_2(f, v) \wedge y \in (z, v) \wedge v - z < 1/s)\,,$$
$$\Psi_s(f, y) \rightleftharpoons \forall z\,(\varphi_1(f, z) \rightarrow z - y \leq 1/s) \wedge \forall z\,(\varphi_2(f, z) \rightarrow y - z \leq 1/s)\,.$$

4 Majorant-Computability and Computable Analysis

In order to illustrate how the concept of majorant-computability works in some particular cases let us consider computability of functionals $F : C[0, 1] \rightarrow \mathbb{R}$, where $C[0, 1]$ is the set of continuous real functions defined on $[0, 1]$.

For this purpose we define $\mathcal{F} = \langle C[0, 1], E,\ H\rangle$, where predicates E and H have the following meaning:

$$E(f, x_1, x_2, z) \rightleftharpoons f|_{[x_1,x_2]} < z \vee (x_1 < 0) \vee (x_2 > 1)\ \textit{and}$$
$$H(f, x_1, x_2, z) \rightleftharpoons f|_{[x_1,x_2]} > z \vee (x_1 < 0) \vee (x_2 > 1)\,.$$

Below we will use the notations E_f and H_g for $E(f, \cdot, \cdot, \cdot)$ and $H(g, \cdot, \cdot, \cdot)$.

In this case we get the following characterisation of majorant-computable functionals as a straightforward corollary of the theorem 3.

Definition 4. *A partial functional $F : C[a, b] \rightarrow \mathbb{R}$ is said to be* **shared by two Σ-formulas φ_1 and φ_2** *if the following assertions hold. For every $u \in C[a, b]$ and $y \in \mathbb{R}$, $F(f) = y$ holds if and only if*

$$y > z \leftrightarrow \mathbf{HF}(\mathcal{R}) \models \varphi_1(H_f, E_f, x_1, x_2, z)\ \textit{and}$$
$$y < z \leftrightarrow \mathbf{HF}(\mathcal{R}) \models \varphi_2(H_f, E_f, x_1, x_2, z).$$

Corollary 5. *A partial functional $F : C[a, b] \rightarrow \mathbb{R}$ majorant-computable if and only if F is shared by two Σ-formulas.*

The following theorem reveals algorithmic properties of Σ-formulas over $\mathbf{HF}(\mathcal{R})$.

Theorem 6 (Semantic Characterisation of Σ-definability). *A set $B \subseteq \mathcal{R}^n$ is Σ-definable if and only if there exists an effective sequence of quantifier free formulas in the language $\sigma_0 \cup \{E, H\}$, $\{\Phi_s(x)\}_{s\in\omega}$, such that*

$$x \in B \leftrightarrow \mathbf{HF}(\mathcal{R}) \models \bigvee_{s\in\omega} \Phi_s(x).$$

The proof of this theorem is based on Gandy's theorem for abstract structures without equality [9] and the technique developed in [10]. For the complete proof we refer to the full version of this paper [11].

Now we compare majorant-computability and computability in the sense of computable analysis [15]. Informally, a real-valued functional is *computable* if there is a Turing machine which from approximations of arguments produces approximations of the functional value. In the framework of computable analysis [15], approximations are given by representations. Below we use the following appropriate representations. For the real numbers we take the representation ρ_I, where the name of a real number is a sequence of compact intervals with rational endpoints fast converging to it. We use the Cauchy representation $\delta_{\mathrm{C}}^{[0,1]}$ of $C[0,1]$, where the name of a function f is a sequence of rational polygons fast converging to it. Let us recall their formal definitions according to [15].

Definition 7 (Interval representation of $\mathbb{R}$). *Let ν_{Int} be the standard notation of the set of compact intervals with rational endpoints.*

$$\rho_I(p) = x :\Longleftrightarrow \left[\begin{array}{l} \text{there are words } \; u_0, u_1 \ldots \in \mathrm{dom}(\nu_{Int}) \\ \text{such that } \; p = u_0\, \iota\, u_1 \ldots, \\ (\forall k)\left(\nu_{Int}(u_{k+1}) \subseteq \nu_{Int}(u_k) \text{ and } \mathrm{length}(\nu_{Int}(u_k)) \leq 2^{-k}\right) \\ \text{and } \; \{x\} = \nu_{Int}(u_0) \cap \nu_{Int}(u_1) \cap \ldots, \end{array}\right.$$

where ι is a special symbol for separating words.

Definition 8 (Cauchy representation of $C[0,1]$). *A rational polygon is a function $g \in C[0,1]$ for which there are rational numbers $a_0, b_0, \ldots, a_k, b_k$ such that $0 \leq a_0 < a_1 < \ldots < a_k \leq 1$ and for all $x \in A$,*

$$g(x) = b_{i-1} + (x - a_{i-1})(b_i - b_{i-1})/(a_i - a_{i-1}) \quad \text{if} \quad a_{i-1} \leq x \leq a_i$$

for $i = 1, \ldots, k$. Let ν_{Pg} be a standard notation of the set Pg of rational polygons on $[0,1]$. The Cauchy representation $\delta_C^{[0,1]}$ of $C[0,1]$ is defined as follows:

$$\delta_{\mathrm{C}}^{A}(p) = f :\Longleftrightarrow \left[\begin{array}{l} \text{there are words } \; w_0, w_1 \ldots \in \mathrm{dom}(\nu_{\mathrm{Pg}}) \\ \text{such that } p = w_0\, \iota\, w_1 \ldots \\ d(\nu_{\mathrm{Pg}}(w_i), \nu_{\mathrm{Pg}}(w_k)) \leq 2^{-i} \text{ for } i < k \\ \text{and } f = \lim_{i\to\infty} \nu_{\mathrm{Pg}}(w_i)\ . \end{array}\right.$$

If $\delta_C^{[0,1]}(w_0\, \iota\, w_1 \ldots) = f$, then for each n, f is in the ball with center $\nu_{\mathrm{Pg}}(w_n)$ and radius 2^{-n}.

Theorem 9. *The class of majorant-computable real-valued functionals coincides with the class of* $\left(\delta_C^{[0,1]}, \rho_I\right)$*-computable real-valued functionals.*

Proof. Suppose $F : C[0,1] \to \mathbb{R}$ is $\left(\delta_C^{[0,1]}, \rho_I\right)$-computable and $\delta_C^{[0,1]}(p) = f$ and $p = w_0 \, \iota \, w_1 \ldots$. By assumption, there is a Type-2 machine T which given input $p = w_0 \, \iota \, w_1 \ldots \in \mathrm{dom}(\delta_C^{[0,1]})$ prints a list of names of fast converging intervals $[q_m^1, q_m^2]$ such that $F(f) \in [q_m^1, q_m^2]$. It is easy to see that the relations $\nu_{\mathrm{Pg}}(w_n) - 2^{-n} < f$ and $f < \nu_{\mathrm{Pg}}(w_n) + 2^{-n}$ are Σ-definable by formulas with occurrences of E_f and H_f. Let $\nu_{\mathrm{Pg}}(w_n) - 2^{-n} < f$ be defined by $\Phi_n(E_f, H_f)$ and $\nu_{\mathrm{Pg}}(w_n) + 2^{-n} < f$ be defined by $\Psi_n(E_f, H_f)$. Assume that T produces $[q_m^1, q_m^2]$ on nth step. Then the Σ-formula

$$\varphi_1(E_f, H_f, z) \rightleftharpoons \bigvee_{n \in \omega} \left(\bigwedge_{i \leq n} \Phi_i(E_f, H_f) \wedge z < q_m^1 \right)$$

defines the left Dedekind cut of $F(f)$. In the same way, we can construct a Σ-formula $\varphi_1(E_f, H_f, z)$ which defines the right Dedekind cut of $F(f)$.

Let $F : C[0,1] \to \mathbb{R}$ be majorant-computable. We consider the functional domain $\mathcal{I}_f[0,1]$ which has been introduced and shown to be an effective ω-continuous domain in [7]. The set $\{\langle \nu_{\mathrm{Pg}}(w_n) - 2^{-n}, \nu_{\mathrm{Pg}}(w_n) + 2^{-n} \rangle \, | n \in \omega\}$ can be taken as an effective basis of this domain. In the similar way, as in [8], it is possible to show that the formulas φ_1 and φ_2, from the definition 4, induce operator $F^* : \mathcal{I}_f[0,1] \to I$, where I is the interval domain. This, in particular, means that the set

$$\left\{(n,m) | F(\langle \nu_{\mathrm{Pg}}(w_n) - 2^{-n}, \nu_{\mathrm{Pg}}(w_n) + 2^{-n} \rangle) \gg \nu_{\mathrm{Int}}(u_m)\right\}$$

is computable enumerable. Then a program for a strong $\left(\delta_C^{[0,1]}, \rho_I\right)$-realization of F can be programmed straightforwardly.

5 Future Work

In this paper we extend the notion of majorant-computability to real-valued functionals. One of the main features of this notion is that it is independent from concrete representations of the elements of structures. We also have given a semantic characterisation of this notion of computability for some concrete structures such as continuous functions on reals. Such characterisation allows us to show that our notion of computability on such structures captures the same class of computable objects as the notion of computability from computable analysis. In this respect the following direction of research is of special interest: to propose and study reasonable requirements on the universe and the language of an abstract structure without the equality test under which a similar characterisation can be obtained.

References

1. M. Ajtai. First-order definability on finite structures. *Annals of Pure and Applied Logic*, 45:211–225, 1989.
2. J. Barwise. *Admissible sets and Structures.* Springer Verlag, Berlin, 1975.
3. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer Verlag, Berlin, 1996.
4. Yu. L. Ershov. *Definability and computability.* Plenum, New-York, 1996.
5. H. Friedman and K. Ko. Computational complexity of real functions. *Theoretical Computer Science*, 20:323–352, 1992.
6. M. Korovina and O. Kudinov, Characteristic properties of majorant-computability over the reals. In *Proc. of CSL'98*, volume 1584 of *Lecture Notes in Computer Science*, pages 188–204, 1999.
7. M. Korovina, O. Kudinov, Formalisation of Computability of Operators and Real-Valued Functionals via Domain Theory, in Proceedings of CCA-2000, LNCS 2064, 2000, 146-168.
8. M. Korovina and O. Kudinov, Semantic Characterisations of Second-Order Computability over the Real Numbers. LNCS 2142, 2001, pages 160–173.
9. M.V. Korovina, Gandy's Theorem on Abstract Structures without the Equality Test, In Proc. LPAR'03, LNAI 2850, pp. 290–301, 2003
10. M Korovina, Computational aspects of Σ-definability over the real numbers without the equality test. In Proc. CSL'03, LNCS 2803, pp. 330–344
11. M Korovina, O Kudinov, Towards Computability of Higher Type Continuous Data. the full version, `http://www.brics.dk/~korovina/highertypes.pdf`
12. Y. N. Moschovakis. Abstract first order computability I, II. *Transactions of the American Mathematical Society*, 138:427–504, 1969.
13. M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics.* Springer Verlag, Berlin, 1988.
14. J. V. Tucker and J. I. Zucker. Projections of semicomputable relations on astract data types. *International Journal of the Foundations of Computer Science*, 2:267–296, 1991.
15. K. Weihrauch. *Computable Analysis.* Springer Verlag, Berlin, 2000.

The Power of Mobility: Four Membranes Suffice

Shankara Narayanan Krishna

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay,
Powai, Mumbai, India 400 076
krishnas@cse.iitb.ac.in

Abstract. We continue the study of P systems with mobile membranes introduced in [6], which is a variant of P systems with active membranes having none of the features like polarizations, label change and division of non-elementary membranes. This variant was shown to be universal using only the simple operations of endocytosis and exocytosis; moreover, if elementary membrane division is allowed, it is capable of solving hard problems. Here, we investigate the power of the two operations (endocytosis, exocytosis) in more detail: 2 membranes can generate sets of vectors outside $PsMAT$, and four membranes give universality.

1 Introduction

P systems are a class of distributed parallel computing models inspired from the way the living cells process chemical compounds, energy, and information. One of the central operations in cell biology is cell division, and with this inspiration, P systems with active membranes were introduced in [7]. This variant was shown to be computationally universal as well as to be able to solve hard problems. The features used by this variant include the use of polarizations (+, -, 0) and division of non-elementary as well as elementary membranes, giving rise to an exponential workspace. These features are quite powerful, thus making the system powerful. Many attempts have been made to define equivalent systems having none of the above features, but in general, removal of one feature has requested the introduction of other powerful operations [8], [9]. [6] is an attempt in this direction, wherein we introduce a variant of P systems with none of the above mentioned features, but instead use two simple operations : *endocytosis* and *exocytosis*. These operations are different and simpler than the operations considered in [1], [2], [4] and [3].

In this paper, we take a closer look at the power of endocytosis and exocytosis rules. We have investigated the generative capacity of systems with 2 and 4 membranes and understand that endocytosis and exocytosis have a surprising power : universality is obtained with 4 membranes in contrast to achieving the same using label changing, membrane division and membrane dissolution keeping at most 3 membranes all the while [9].

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 242–251, 2005.

2 Some Prerequisites

We refer to [5], [10] for the elements of formal language theory we use here. We only specify that for a string $x \in V^*$ and a symbol $a \in V$, we denote by $|x|$ the length of x and by $|x|_a$ the number of occurrences of the symbol a in the string x. For $w \in V^*$ with $V = \{a_1, \ldots, a_n\}$, we denote by $\Psi_V(w)$ the Parikh vector of w, that is, $\Psi_V(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$; this is extended to languages in a natural way. For a family FL of languages, we denote by $PsFL$ the family of Parikh sets of vectors associated with languages in FL.

A multi set over an alphabet V is represented by a string over V (and by all its permutations) and each string precisely identifies a multi set; the Parikh vector associated with the string indicates the multiplicities of each element of V in the corresponding multi set. For basic elements of membrane computing we refer to [8]; for the state-of-the art of the domain, the reader may consult the bibliography from the web address http://psystems.disco.unimib.it.

We recall the definition of the family MAT. A context-free matrix grammar without appearance checking is a construct $G = (N, T, S, M)$ where N, T are disjoint alphabets of non-terminals and terminals, $S \in N$ is the axiom, and M is a finite set of matrices of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$ of context-free rules. For a string x, a matrix $m : (r_1, \ldots r_n)$ is executed by applying the productions $r_1, \ldots, r_n$ one after the another, following the order in which they appear in the matrix. We write $w \Rightarrow_m z$ if there is a matrix $m : (A_1 \to x_1, \ldots A_n \to x_n)$ in M and the strings $w_1, \ldots w_{n+1}$ in $(N \cup T)^*$ such that $w = w_1, w_{n+1} = z$, and for each $i = 1, 2, \ldots, n$ we have $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$. The language generated by $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. The family of languages generated by context-free matrix grammars is denoted by MAT.

For proving computational universality, we use the notion of a matrix grammar with appearance checking in the *improved strong binary normal form*, introduced in [6]. Such a grammar is a construct $G = (N, T, S, M, F)$, where $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, two distinguished symbols $B^{(1)}, B^{(2)} \in N_2$, and the matrices in M of one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$,
3. $(X \to Y, B^{(j)} \to \#)$, with $X, Y \in N_1, j = 1, 2$,
4. $(X \to a, A \to x)$, with $X \in N_1, A \in N_2, x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists of all the rules $B^{(j)} \to \#$, $j = 1, 2$, appearing in matrices of type 3; # is a trap-symbol, once introduced it is never removed. (Clearly, a matrix of type 4 is used only once, in the last step of a derivation.)

3 P Systems with Mobile Membranes

We now briefly recall P systems with mobile membranes introduced in [6].

A *P system with mobile membranes* is a construct

$$\Pi = (V, H, \mu, w_1, \ldots, w_n, R),$$

where: $n \geq 1$ (the initial *degree* of the system); V is an alphabet (its elements are called *objects*); H is a finite set of *labels* for membranes; μ is a *membrane structure*, consisting of n membranes, labeled (not necessarily in a one-to-one manner) with elements of H; $w_1, w_2, \ldots, w_n$ are strings over V, describing the *multi sets of objects* placed in the n regions of μ, and R is a finite set of *developmental rules*, of the following forms:

(a) $[_m a \rightarrow v]_m$, for $m \in H, a \in V, v \in V^*$; object evolution rules.
(b) $[_h a]_h [_m \,]_m \rightarrow [_m [_h b]_h]_m$, for $h, m \in H, a, b \in V$;
endocytosis rules: an elementary membrane labeled h enters the adjacent membrane labeled m, under the control of object a; the labels h and m remain unchanged during this process, however, the object a may be modified to b during the operation; m is not necessarily an elementary membrane.
(c) $[_m [_h a]_h]_m \rightarrow [_h b]_h [_m \,]_m$, for $h, m \in H, a, b \in V$;
exocytosis: an elementary membrane labeled h is sent out of a membrane labeled m, under the control of object a; the labels of the two membranes remain unchanged, but the object a from membrane h may be modified during this operation; membrane m is not necessarily elementary.
(d) $[_h a]_h \rightarrow [_h b]_h [_h c]_h$, for $h \in H, a, b, c \in V$;
division rules for elementary membranes: in reaction with an object a, the membrane labeled h is divided into two membranes labeled h, with the object a replaced in the two new membranes by possibly new objects.

We do not use rules of type (d) for our investigations in this paper. The rules are applied according to the following principles:

1. All rules are applied in parallel, non-deterministically choosing the rules, the membranes, and the objects, but in such a way that the parallelism is maximal; this means that in each step we apply a set of rules such that no further rule can be added to the set, no further membranes and objects can evolve at the same time.
2. The membrane m from each type (a) – (c) of rules as above is said to be *passive*, while the membrane h is said to be *active*. In any step of a computation, any object and any active membrane can be involved in at most one rule, but the passive membranes are not considered involved in the use of rules (hence they can be used by several rules at the same time as passive membranes); for instance, a rule $[_m a \rightarrow v]_m$, of type (a), is considered to involve only the object a, not also the membrane m.
3. The evolution of objects and membranes takes place in a bottom-up manner. After having a (maximal) set of rules chosen, they are applied starting from the innermost membranes, level by level, up to the skin membrane (all these sub-steps form a unique evolution step, called a *transition* step).
4. When a membrane is moved across another membrane, by endocytosis or exocytosis, its whole contents (its objects) are moved; because of the bottom-up way of using the rules, the inner objects first evolve (if there are rules applicable for them), and then any membrane is moved with the contents as obtained after this inner evolution.

5. If a membrane exits the system (by exocytosis), then its evolution stops, even if there are rules of type (a) which would be applicable to it provided that the membrane would be in the system.
6. All objects and membranes which do not evolve at a given step (for a given choice of rules which is maximal) are passed unchanged to the next configuration of the system.

By using the rules in this way, we get transitions among the configurations of the system. A sequence of transitions is a computation, and a computation is successful if it halts (it reaches a configuration where no rule can be applied). During a computation, membranes can leave the skin membrane (by means of rules of type (c)). The multiplicity vector of the multi set from such a membrane is considered as a result of the computation. Thus, the result of a halting computation consists of all vectors describing the multiplicity of objects from all membranes sent out of the system during the computation; a non-halting computation provides no output. The set of all vectors of natural numbers produced in this way by a system Π is denoted by $Ps(\Pi)$. Each membrane which exits the system gives a vector, hence a computation can produce several vectors, all of them considered in the set $Ps(\Pi)$.

The family of all sets $Ps(\Pi)$ generated by systems of degree atmost n using endocytosis and exocytosis rules, is denoted by $PsMP_n(endo, exo)$. If a type of rules is not used, then we omit its "name" from the list. For instance, if only exocytosis rules are used, we write $PsMP_n(exo)$. It has been shown in [6] that P systems with mobile membranes having n membranes, $n \geq 9$ can compute the family of all Turing computable sets of vectors of natural numbers.

4 The Power of Endocytosis and Exocytosis

We adopt the **convention** that when comparing the sets of vectors generated (accepted) by two devices, we ignore the empty vector. We also follow the shorthand notation $[_i a_1 \to a_2, b_1 \to b_2, \ldots, m_1 \to m_2]_i$ for representing rules $[_i a_1 \to a_2]_i, [_i b_1 \to b_2]_i, \ldots, [_i m_1 \to m_2]_i$ of membrane i in a compact way.

Theorem 1. $PsMP_2(exo) - PsMAT \neq \emptyset$.

Proof. Consider the system $\Pi = (\{a, b\}, \{1, 2\}, [_1[_2\ ]_2]_1, \emptyset, ab, R)$ where R contains the following rules:

$$R = \{[_1[_2 b]_2]_1 \to [_2 b]_2[_1\ ]_1, [_2 b \to b]_2, [_2 a \to aa]_2\}$$

Clearly, $Ps\Pi = \{(2^n, 1) \mid n \geq 1\}$, which is not in $PsMAT$.

□

Theorem 2. $PsMP_4(endo, exo) = PsRE$.

Proof. Consider a matrix grammar $G = (N, T, S, M, F)$ with appearance checking in the improved strong binary normal form ($N = N_1 \cup N_2 \cup \{S, \#\}$), having

n_1 matrices of types 2 and 4 and n_2 matrices of type 3. Let $B^{(1)}$ and $B^{(2)}$ be the two objects in N_2 for which we have rules $B^{(j)} \to \#$ in matrices of M. The matrices of the form $(X \to Y, B^{(j)} \to \#)$ are labeled by m'_i, $1 \le i \le n_2$ with $i \in lab_j$, for $j \in \{1,2\}$, such that lab_1, lab_2, and $lab_0 = \{1, 2, \ldots, n_1\}$ are mutually disjoint sets.

We construct a P system with mobile membranes of degree 4,

$$\Pi = (V, H, \mu, w_1, w_2, w_3, w_4, R)$$

where

$$\begin{aligned}
V &= N_1 \cup N_2 \cup \{Y_{i,j}, a_{i,j}, A_{k,j} \mid 0 \le i \le 2n_1+1, 2 \le k \le 2n_1+1, 1 \le j \le n_1\} \\
&\quad \cup \{\#, Y', Y'', Y''', a', Y^{(1)}, Y^{(1)\prime}, Y^{(1)\prime\prime}, Y^{(2)}, Y^{(2)\prime}, Y^{(2)\prime\prime}, Y^{(2)\prime\prime\prime}, B^{(1)}, B^{(2)}\} \\
&\quad \cup \{a_l, b_m, \langle x\rangle_i, \langle x\rangle \mid 0 \le l \le 7, m \in \{1,3,5,7\}, i \in \{0,1\}, (X \to Y/a, A \to x) \in M\}, \\
H &= \{1,2,3,4\}, \\
\mu &= [_1[_2\,]_2[_3[_4\,]_4]_3]_1, \\
w_2 &= XA, \text{ where } (S \to XA) \text{ is the initial matrix of } G, w_i = \emptyset, i \ne 2.
\end{aligned}$$

and R consists of the following rules:

1. Simulation of matrices $m_j : (X \to Y, A \to x)$ or $m_j : (X \to a, A \to x)$, $2 \le j \le n_1$.
   1. $[_2X \to Y_{2j+1,j}]_2$, $m_j : (X \to Y, A \to x)$, and $[_2X \to a_{2j+1,j}]_2$, $m_j : (X \to a, A \to x)$,
   2. $[_2A]_2[_3\,]_3 \to [_3[_2A_{2j+1,j}]_2]_3, m_j : (X \to Y, A \to x)$ or $m_j : (X \to a, A \to x)$, $[_2\#]_2[_3\,]_3 \to [_3[_2\#]_2]_3$,
   3. $[_2A_{i,k}]_2[_4\,]_4 \to [_4[_2A_{i-1,k}]_2]_4, i \ge 3$,
   4. $[_4[_2A_{i,k}]_2]_4 \to [_2A_{i-1,k}]_2[_4\,]_4, i \ge 3$,
   5. $[_2A_{i,k} \to \#]_2, i \ge 2$,
   6. $[_4[_2A_{2,k}]_2]_4 \to [_2\langle x\rangle_0]_2[_4\,]_4$,
   7. $[_2\langle x\rangle_0 \to \langle x\rangle_1, \langle x\rangle_1 \to \#]_2$,
   8. $[_2Y_{i,j} \to Y_{i-1,j}, a_{i,j} \to a_{i-1,j}, Y_{0,j} \to \#]_2, i \ge 1$,
   9. $[_2Y_{1,j}]_2[_4\,]_4 \to [_4[_2Y']_2]_4$, $m_j : (X \to Y, A \to x)$, $[_2a_{1,j}]_2[_4\,]_4 \to [_4[_2a']_2]_4$, $m_j : (X \to a, A \to x)$,
   10. $[_4[_2\langle x\rangle_1]_2]_4 \to [_2\langle x\rangle]_2[_4\,]_4$,
   11. $[_2Y' \to Y'', a' \to a_0, Y'' \to Y''', \langle x\rangle \to x]_2$,
   12. $[_3[_2Y'']_2]_3 \to [_2Y]_2[_3\,]_3$.

   In the initial configuration, we have the objects X, A corresponding to the initial matrix in membrane 2. To simulate a matrix of the above type (2 or 4), rules 1 and 2 have to be applied in parallel. This results in X being replaced by $Y_{2j+1,j}$ and the endocytosis rule 2 ensures that a single $A \in N_2$

is replaced by $A_{2j+1,j}$. No other $A \in N_2$ can be replaced until membrane 2 comes out of membrane 3 by using the exocytosis rule 12. Inside membrane 3, the rules 8 (for Y) and 3,4 (for A) are used to decrement the indices of Y, A. This is done to check if the indices of Y, A are the same, and in that case to rewrite A according to the matrix m_j. This is continued until $A_{i,j}$ becomes $A_{2,j}$. At this point of time, if the indices are the same, then the index of Y should be $Y_{2,j}$. Observe that since we started out with an odd index for A, when the index reaches 2, membrane 2 should be inside membrane 4. Using rules 6 and 8 for $i = 2$, we get $[_2 w\langle x\rangle_0 w_1 Y_{1,j} w_2]_2, w, w_1, w_2 \in V^*$. At this point of time, membrane 2 will be inside membrane 3. Now, rules 9, 7 should be used, replacing $Y_{1,j}$ by Y' and $\langle x\rangle_0$ by $\langle x\rangle_1$. Rules 10 and 11 should be applied next, replacing Y' by Y''. Once this is done, membrane 2 is inside 3 and can move out of 3 using rule 12. A new simulation is started after membrane 2 exits membrane 3.

Observe that a correct simulation of a type 2 or 4 matrix is obtained if and only if rules are applied in the order as mentioned above. In case the endocytosis rule 15 is used instead of 3,9 and/or the exocytosis rule 21 is used instead of 4,6,10, the object # is introduced in the system. This will prevent any output being sent out of the system, by blocking membrane 2 inside membrane 3 or 4.

Now let's examine what happens if the indices of A, Y are not the same.

Case 1 : Index of Y > Index of A.
In this case, when membrane 2 comes out of membrane 4 using rule 6, the index of Y will not be decremented from 2 to 1. Observe that the indices of A, Y will always have the same parity (if not, there will be no $A_{i,j}$; it would be replaced by a #) . Hence, the index of Y will be decremented from $2j$ to $2j-1$, $j \geq 2$. The smallest possible value of the index is 3 (decremented from 4 to 3). We have the following possible transitions:

(a) Assume that rules 1-12 are the only ones which are applied.

$$[_2\langle x\rangle_0 Y_{3,j}]_2[_4\,]_4 \to [_2\langle x\rangle_1 Y_{2,j}]_2[_4\,]_4 \to [_2\#Y_{1,j}]_2[_4\,]_4$$
$$\to [_4[_2\#Y']_2]_4 \to [_4[_2\#Y'']_2]_4 \to [_4[_2\#Y'''']_2]_4$$

We then have the above sequence of transitions, leading to blocking membrane 2 inside membrane 4.

(b) Assume that rules 1-12, and rule 15 are applied.

$$[_2\langle x\rangle_0 Y_{3,j} B^{(1)}]_2[_4\,]_4 \to [_4[_2\langle x\rangle_1 Y_{2,j}\#]_2]_4 \to [_2\langle x\rangle Y_{1,j}\#]_2[_4\,]_4$$
$$\to [_4[_2 x Y'\#]_2]_4 \to [_4[_2 x Y''\#]_2]_4 \to [_4[_2 x Y''''\#]_2]_4$$

There is another possibility in the second transition of the above sequence, viz., replacing $\langle x\rangle_1$ by #, but that will again mean blocking membrane 2. Similarly, it is possible to replace $Y_{1,j}$ by $Y_{0,j}$ in the third transition, blocking membrane 2 inside membrane 3 or 4.

(c) Assume that rules 1-12, and 21 are used.

$$[_2\langle x\rangle_0 Y_{3,j} B^{(2)}]_2[_4\,]_4 \to [_2\langle x\rangle_1 Y_{2,j} B^{(2)}]_2[_4\,]_4 \to [_2\#Y_{1,j} B^{(2)}]_2[_4\,]_4 \to [_4[_2\#Y' B^{(2)}]_2]_4 \to [_2\#Y''\#]_2[_4\,]_4 \to [_2\#Y\#]_2[_3\,]_3$$

(d) Assume that rules 1-12, 15 and 21 are applied.

$$[_2\langle x\rangle_0 Y_{3,j} B^{(1)} B^{(2)}]_2[_4\,]_4 \to [_4[_2\langle x\rangle_1 Y_{2,j}\#B^{(2)}]_2]_4 \to [_2\#Y_{1,j}\#\#]_2[_4\,]_4 \to [_4[_2\#Y'\#\#]_2]_4 \to [_4[_2\#Y''\#\#]_2]_4 \to [_4[_2\#Y'''\#\#]_2]_4$$

We also have other sequences of transitions; but in all cases, the symbol # is introduced in membrane 2. We have similar cases if we consider $Y_{2j+1,j}$, $j \geq 2$.

Case 2 : Index of Y < Index of A.
In this case, it is possible to have $[_2 A_{2j+1,j} Y_{1,j}]_2$ inside membrane 3. The smallest possible value for $A_{2j+1,j}$ then is $A_{3,j}$. Since the index of Y has reached 1, and since membrane 2 is inside membrane 3, rule 9 can be used. This will replace $Y_{1,j}$ by Y', taking membrane 2 inside 4, but will replace the $A_{3,j}$ by #. There is also the possibility of replacing $Y_{1,j}$ by $Y_{0,j}$ in case rule 15 or rule 3 is used, but the point to note is that if the indices are not the same, or if rules are not applied in the correct order, the blocking symbol # is introduced.

Observe that simulation of a type 4 matrix is on similar lines, except that we have an a in place of Y. During the finishing stages of a type 4 simulation, we use rules 9 and 11 to replace $a_{1,j}$ by a' and the to rewrite it to a_0. We will see later how this is useful for verifying if all simulations are done correctly.

2. Simulation of matrices $m'_i : (X \to Y, B^{(j)} \to \#), j = 1, 2,$ and $1 \leq i \leq n_2$.

13. $[_2 X]_2[_3\,]_3 \to [_3[_2 Y^{(1)}]_2]_3$, $m'_i : (X \to Y, B^{(1)} \to \#)$,
14. $[_2 Y^{(1)} \to Y^{(1)'}, Y^{(1)'} \to Y^{(1)''}]_2$,
15. $[_2 B^{(1)}]_2[_4\,]_4 \to [_4[_2\#]_2]_4$,
16. $[_3[_2 Y^{(1)'}]_2]_3 \to [_2 Y]_2[_3\,]_3$,
17. $[_2 X]_2[_3\,]_3 \to [_3[_2 Y^{(2)}]_2]_3$, $m'_i : (X \to Y, B^{(2)} \to \#)$,
18. $[_2 Y^{(2)}]_2[_4\,]_4 \to [_4[_2 Y^{(2)'}]_2]_4$,
19. $[_2 Y^{(2)'} \to Y^{(2)''}, Y^{(2)''} \to Y^{(2)'''}]_2$,
20. $[_2 Y^{(2)} \to Y^{(2)'''}]_2$,
21. $[_4[_2 B^{(2)}]_2]_4 \to [_2\#]_2[_4\,]_4$,
22. $[_4[_2 Y^{(2)''}]_2]_4 \to [_2 Y'']_2[_4\,]_4$.

The simulation of matrices of type 3 begins by rule 13 or 17. To simulate $(X \to Y, B^{(1)} \to \#)$, rule 13 is used. This is followed by rules 14 and 15 in parallel in case $B^{(1)}$ exists. If no $B^{(1)}$ exists, then rule 16 can be used to send out membrane 2, successfully completing the simulation.

If $B^{(1)}$ exists, we will have membrane 2 inside membrane 4. There is no way for membrane 2 to come out of 4, except in case a $B^{(2)}$ exists. Even then, membrane 2 cannot come out of 3, since the $Y^{(1)'}$ would have evolved to $Y^{(1)''}$, preventing application of rule 16.

To simulate $(X \to Y, B^{(2)} \to \#)$, rule 17 is used. For a correct simulation, this should be followed by rule 18. If instead, one chooses rule 15, then membrane 2 will be blocked inside membrane 3 since $Y^{(2)}$ evolves to $Y^{(2)'''}$. Assuming that 18 was applied, we need to apply rules 19 and 21 (if applicable) in the next step. If rule 21 was not applicable, we are done; rules 22 and 12 can be used, bringing membrane 2 back home. However, if rule 21 was applied, then membrane 2 will be blocked inside membrane 3 or 4 since $Y^{(2)''}$ evolves to $Y^{(2)'''}$.

3. Checking for non-terminals after simulation of a terminal matrix $(X \to a, A \to x)$.

23. $[_2 a_0 \to a_1, a_1 \to b_1]_2$,
24. $[_2 a_1]_2 [_4\,]_4 \to [_4 [_2 a_2]_2]_4$,
25. $[_2 a_2 \to a_3, a_3 \to b_3]_2$,
26. $[_4 [_2 a_3]_2]_4 \to [_2 a_4]_2 [_4\,]_4$,
27. $[_2 a_4 \to a_5, a_5 \to b_5]_2$,
28. $[_3 [_2 a_5]_2]_3 \to [_2 a_6]_2 [_3\,]_3$,
29. $[_2 a_6 \to a_7, a_7 \to b_7]_2$,
30. $[_1 [_2 a_7]_2]_1 \to [_2 a]_2 [_1\,]_1$.

Once the type 4 matrix is simulated, we need to ensure that there are no more $A, B^{(1)}, B^{(2)} \in N_2$'s and no #'s to ensure a correct simulation. If this is indeed the case, then membrane 2 can be sent out as the output of the system. Observe that the $a_{i,j}$ are the counterparts of $Y_{i,j}$'s in a type 4 matrix simulation, and do the same thing, until $a_{1,j}$ is replaced by a' using the endocytosis rule 9. We have already seen that if this was obtained (application of rule 9) not by correct means, then a # will be present in the system.

Assuming that things took place correctly and rule 9 was applied, in the next step, we use rules 10 and 11 $(a' \to a_0)$. In the next step, we replace $\langle x \rangle$ by x and a_0 by a_1 (rule 23). We ultimately have membrane 2 inside membrane 3 with a_1. (This scenario is possible in other wrong ways, but remember that these will introduce a #). In case membrane 2 with a_1 is not inside membrane 3, the computation will halt in the next step replacing a_1 by b_1 with no output.

Lets see what happens when membrane 2 is inside membrane 3 with a_0.

$$
\begin{array}{l}
[_3[_2a_0]_2[_4\ ]_4]_3 \rightarrow [_3[_2a_1]_2[_4\ ]_4]_3 \rightarrow [_3[_4[_2a_2]_2]_4]_3\ (\Rightarrow \text{ no } B^{(1)}) \\
\quad \downarrow \text{ rule 15} \\
[_3[_4[_2a_1\#]_2]_4]_3 \rightarrow [_3[_4[_2b_1\#]_2]_4]_3 \\
\quad \downarrow \text{ rule 21} \\
[_3[_2b_1\#\#]_2[_4\ ]_4]_3
\end{array}
$$

Thus it is possible to get a_2 only if there were no $B^{(1)}$'s, otherwise, rule 15 would be applied when a_0 evolves to a_1. Now let us see what happens with a_2 inside membrane 4.

$$
\begin{array}{l}
[_3[_4[_2a_2]_2]_4]_3 \rightarrow [_3[_4[_2a_3]_2]_4]_3 \rightarrow [_3[_2a_4]_2[_4\ ]_4]_3\ (\Rightarrow \text{ no } B^{(2)}) \\
\quad \downarrow \text{ rule 21} \\
[_3[_2a_3\#]_2[_4\ ]_4]_3 \rightarrow [_3[_2b_3\#]_2[_4\ ]_4]_3 \\
\quad \downarrow \text{ rule 15} \\
[_3[_4[_2b_3\#\#]_2]_4]_3
\end{array}
$$

Thus, it is possible to get a_4 only if there are no $B^{(1)}, B^{(2)}$. Now we need to check only for $A \in N_2, \#$. Finally, we will have membrane 2 adjacent to membrane 3 only if it contains no $A, B^{(1)}, B^{(2)}, \#$. Observe that the only ways for membrane 2 to come out of membrane 3 are via the rules 12, 16 and 28. If the last matrix has been simulated (giving an a'), then rule 28 is the only option.

$$
\begin{array}{l}
[_1[_3[_2a_4]_2[_4\ ]_4]_3]_1 \rightarrow [_1[_3[_2a_5]_2[_4\ ]_4]_3]_1 \rightarrow [_1[_2a_6]_2[_3\ ]_3]_1 \rightarrow [_1[_2a_7]_2[_3\ ]_3]_1\ (\text{no } A, \#) \\
\qquad\qquad\qquad\qquad \downarrow \qquad\qquad\qquad\qquad \downarrow \text{ rule 2} \\
\qquad\qquad\qquad\quad [_3[_2b_5]_2[_4\ ]_4]_3 \qquad [_1[_3[_2a_7\# \text{ or } a_7A_{i,j}]_2]_3]_1 \rightarrow^* [_1[_3[_2b_7\#]_2]_3]_1
\end{array}
$$

Thus, if we have a_7 in membrane 2, and if membrane 2 is inside membrane 1, then it means that all simulations are carried out correctly. Membrane 2 can now be sent out of the system replacing a_7 by a.

□

5 Conclusion

We have established a universality result with 4 membranes, using the operations of endocytosis and exocytosis and conjecture that the result is optimal with respect to the number of membranes and depth. We also conjecture that these operations can be coupled with some basic (already existing) operations to simulate P systems with active membranes, thereby giving rise to an equivalent, but much simpler counterpart for P systems with active membranes. An interesting open problem worthwhile consideration is the power of these systems with 3 membranes.

References

1. A. Alhazov, T.-O. Ishdorj, Membrane operations in P systems with active membranes, *Proc. Second Brainstorming Week on Membrane Computing*, Sevilla, February 2004, TR 01/04 of Research Group on Natural Computing, Sevilla University, 2004, 37–44.
2. G. Bel Enguix, M.D. Jiménez-Lopez, Linguistic membrane systems and applications, in *Applications of Membrane Computing* (G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds.), Springer-Verlag, Berlin, to appear.
3. D. Besozzi, C. Zandron, G. Mauri, N. Sabadini, P systems with gemmation of mobile membranes, *Proc. ICTCS 2001*, Torino, *LNCS* 2202 (A. Restivo, S.R. Della Rocca, L. Roversi, eds.), Springer-Verlag, Berlin, 2001, 136–153.
4. E. Csuhaj-Varjú, A. Di Nola, Gh. Păun, M.J. Peŕez-Jiménez, G. Vaszil, Editing configurations of P systems, manuscript, 2004.
5. J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.
6. S. N. Krishna, Gh. Păun, P systems with mobile membranes, submitted, 2004.
7. Gh. Păun, P systems with active membranes: Attacking NP-complete problems, *Journal of Automata, Languages and Combinatorics*, 6, 1 (2001), 75–90.
8. Gh. Păun, *Computing with Membranes. An Introduction*, Springer-Verlag, Berlin, 2002.
9. Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori, On the power of membrane division in P systems, *Theoretical Computer Science*, 324, 1 (2004), 61–85.
10. G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.

The Small Grzegorczyk Classes and the Typed λ-Calculus

Lars Kristiansen[1,2] and Mathias Barra[2]

[1] Oslo University College, Faculty of Engineering
Cort Adelers gate 30, N-0254 Oslo, Norway
larskri@iu.hio.no
http://www.iu.hio.no/~larskri
[2] Department of Mathematics, University of Oslo
Postboks 1053, Blindern, N-0316 Oslo, Norway
georgba@math.uio.no
http://folk.uio.no/~georgba

1 Introduction

The class $\Delta_0^{\mathbb{N}}$ of rudimentary relations and the small relational Grzegorczyk classes $\mathcal{E}_\star^0$, $\mathcal{E}_\star^1$, $\mathcal{E}_\star^2$ attracted fairly much attention during the latter half of the previous century, e.g. Gandy [6], Paris-Wilkie [20], and numerous others. Yet, the open problems imposed by these classes are still there for new generations to battle. It is well know, and rather obvious, that $\Delta_0^{\mathbb{N}} \subseteq \mathcal{E}_\star^0 \subseteq \mathcal{E}_\star^1 \subseteq \mathcal{E}_\star^2$, but it is not known whether any of the inclusions are strict, indeed it is open if the inclusion $\Delta_0^{\mathbb{N}} \subseteq \mathcal{E}_\star^2$ is strict. It is proved in Bel'tyukov [3] that $\mathcal{E}_\star^1 = \mathcal{E}_\star^2$ implies $\mathcal{E}_\star^0 = \mathcal{E}_\star^2$. Furthermore, we know that $\Delta_0^{\mathbb{N}} = \mathcal{E}_\star^0$ implies $\Delta_0^{\mathbb{N}} = \mathcal{E}_\star^2$ (We do not know if this is proved anywhere in the literature, but it will be a corollary of some of the theorems below.). The open problems can be traced back to Grzegorczyk's initial paper [7] from 1953, and it is fair to say that the problems belong to subrecursion theory, but they are closely related to complexity theory and computer science. Let LINSPACE be the class of number-theoretic relation decidable by a a deterministic Turing machine working in linear space; let NLINSPACE be the corresponding class for a nondeterministic Turing machines; and let ETIME be the class of number-theoretic relation decidable by a a deterministic Turing machine working in exponential. Ritchie [21] proved that LINSPACE $= \mathcal{E}_\star^2$, and of course LINSPACE $\subseteq$ NLINSPACE $\subseteq$ ETIME. Again neither inclusion is known to be strict (It is not even known if the inclusion $\Delta_0^{\mathbb{N}} \subseteq$ ETIME is strict.). These unsolved problems do again relate to other, and perhaps more notorious, problems, e.g. LINSPACE $\neq$ ETIME implies that LOGSPACE is strictly included in P. For more on complexity theory, see Odifreddi [19]; for more the Grzegorczyk classes and the rudimentary relations see Clote [4], Rose [22], Kutylowski [17], Esbelin-More [5].

In this paper we introduce the $\mathcal{L}$-hierarchy $\mathcal{L}^0 \subseteq \mathcal{L}^1 \subseteq \mathcal{L}^2 \subseteq \cdots$ which might shed some new light on the open problems described above. We have $\Delta_0^{\mathbb{N}} \subseteq \mathcal{L}_\star^0 \subseteq \mathcal{E}_\star^0 \subseteq \mathcal{E}_\star^1 \subseteq \mathcal{L}_\star^1 \subseteq \cdots \subseteq \mathcal{L}_\star = \mathcal{E}_\star^2$, and $\mathcal{L}^i = \mathcal{L}^j$ iff $\mathcal{L}_\star^i = \mathcal{L}_\star^j$ for any $i, j \in \mathbb{N}$. Hence, $\mathcal{E}_\star^0 \neq \mathcal{E}_\star^2$ if $\mathcal{L}^i \neq \mathcal{L}^j$ for some $i, j > 0$; and $\Delta_0^{\mathbb{N}} \neq \mathcal{E}_\star^2$ if $\mathcal{L}^0 \neq \mathcal{L}^i$ for some $i > 0$.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 252–262, 2005.

A class in the $\mathcal{L}$-hierarchy is defined by a certain fragment of a typed λ-calculi. No explicit bounds are embodied in the definition. Thus, we have so-called implicit characterisation of $\mathcal{E}^2_\star$ (LINSPACE). To get rid of explicit resource bounds and obtain implicit characterisations of sub-recursive and complexity classes, so-called *ramification techniques* (also known as tiering techniques) have shown to be successful. Numerous examples of ramification can be found in the literature, Simmons [23] may be the first, Bellantoni and Cook's [2] distinction between normal and safe variables is probably the best known. Leivant [18], Beckmann-Weiermann [1], Simmons [24], and Kristiansen-Niggl [13] are other examples. In this paper we achieve implicit characterisations by a qualitatively different technique: *We remove successor-like functions from a standard computability-theoretic framework (typed λ-calculi).* Jones [10, 11] uses the same technique (with functional programming languages). So do Kristiansen-Voda [15] (with functionals of higher types), Kristiansen-Voda [14] (with imperative programming languages) and Kristiansen [12] (with function algebras).

In [16] Kristiansen and Voda show that all the well known deterministic complexity classes can be characterised by fragments of a typed λ-calculi (Gödel's T). It is also sketchy proved that the "type 0 fragment" defining $\mathcal{L}$ captures LINSPACE, and thus $\mathcal{E}^2_\star$. The proof is based on Turing machines. In this paper we make no detours via Turing machines and give a direct and detailed proof of the equality $\mathcal{L}_\star = \mathcal{E}^2_\star$. (The hierarchy $\mathcal{L}^0 \subseteq \mathcal{L}^1 \subseteq \mathcal{L}^2 \subseteq \cdots$ is not introduced in [16].)

2 Preliminaries

We assume that the reader is familiar with the basics of the (typed) λ-calculus, the reader which is not, should consult a standard textbook on the subject, e.g. [9] or [25]. We assume some familiarity with sub-recursive classes and hierarchies. In this section we recall some definitions and results.

We will use some notation and terminology from Clote [4]. An *operator*, here also called *(definition) scheme*, is a mapping from functions to functions. Let $\mathcal{X}$ be a set of functions (possibly given in a slightly informal notation), and let OP be a collection of operators. The *function algebra* $[\mathcal{X}; \text{OP}]$ is the smallest set of functions containing $\mathcal{X}$ and closed under the operations of OP. COMP denotes the definition scheme called *composition*, i.e. the scheme $f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_m(\vec{x}))$ where $m \geq 0$. PR denotes the scheme for *primitive recursion*, i.e.

$$f(\vec{x}, 0) = g(\vec{x}) \qquad f(\vec{x}, y+1) = h(\vec{x}, y, f(\vec{x}, y))$$

BR denotes the scheme *bounded (primitive) recursion*, i.e. the scheme

$$f(\vec{x}, 0) = g(\vec{x}) \qquad f(\vec{x}, y+1) = h(\vec{x}, y, f(\vec{x}, y)) \qquad f(\vec{x}, y) \leq j(\vec{x}, y)$$

Let S denote the successor function. Let I^n_i denote the projection function $I^n_i(x_1, \ldots, x_i, \ldots, x_n) = x_i$ where $1 \leq i \leq n$ are fixed numbers. Let I denote the set of all such projection functions. Let C_k denote the 0-ary constant function $C_k = k$ where $k \in \mathbb{N}$. For any set $\mathcal{F}$ of number-theoretic functions, $\mathcal{F}_\star = \{f \mid f \in \mathcal{F} \text{ and } \text{ran}(f) = \{0, 1\}\}$; i.e. the 0-1-valued functions

of $\mathcal{F}$. We follow the standard conventions, and identify a 0-1-valued function with a relation over the natural numbers. The Grzegorczyk classes $\mathcal{E}^0, \mathcal{E}^1$ and $\mathcal{E}^2$ are defined by $\mathcal{E}^0 = [I, C_0, S; \text{COMP}, \text{BR}]$, $\mathcal{E}^1 = [I, C_0, S, +; \text{COMP}, \text{BR}]$ and $\mathcal{E}^2 = [I, C_0, S, +, x^2 + 2; \text{COMP}, \text{BR}]$. The next lemma, which we state without proof, gives some important and well known properties of the classes.

LEMMA 1. (i) For any $f \in \mathcal{E}^0$ there exist fixed numbers i, k where $1 \leq i \leq n$ such that $f((x_1, \ldots, x_n) \leq x_i + k$. (ii) For any $f \in \mathcal{E}^1$ there exists a fixed number k such that $f(\vec{x}) \leq k \max(\vec{x})$. (iii) For any $f \in \mathcal{E}^2$ there exists a polynomial p such that $f(\vec{x}) \leq p(\vec{x})$.

A relation $R(\vec{x})$ is *rudimentary* when there exist a Δ_0^0 statement $\phi(\vec{x})$ in Peano Arithmetic such that $R(\vec{x})$ holds iff $\mathbb{N} \models \phi(\vec{x})$. We will use $\Delta_0^{\mathbb{N}}$ to denote the class of rudimentary relations.

3 A Hierarchy Stratifying $\mathcal{E}^2$

The next definition yields hierarchy $\mathcal{G} = \bigcup_{n \in \mathbb{N}} \mathcal{G}^n$ stratifying the class $\mathcal{E}^2$. The hierarchy is Grzegorczyk-like in the sense that the growth of the functions in the class $\mathcal{G}^n$ is restricted by explicit bounds.

DEFINITION 2. Let BCOMP denote the definition scheme called *bounded composition*, i.e. the scheme

$$f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_m(\vec{x})) \qquad f(\vec{x}, y) \leq j(\vec{x}, y)$$

where $m \geq 0$.

Define the *bounded successor* function $\hat{\mathsf{S}}$ by $\hat{\mathsf{S}}(x, y) = \begin{cases} x + 1 & \text{if } x < y \\ y & \text{otherwise.} \end{cases}$

Define the function $G_{k,n}$ by $G_{k,n}(x_1, \ldots, x_k) = (\max(x_1, \ldots, x_k, 1) + 1)^{n+1} - 1$. We will normally suppress the k and use G_n to denote the function $G_{k,n}$ for any $k \geq 0$. Further, we will overload notation and also use G_n to denote the set of all $G_{i,j}$ such that $i \geq 1$ and $j \leq n$.

Let $\mathcal{G}^n \stackrel{\text{def}}{=} [I, C_0, \hat{\mathsf{S}}, G_n; \text{BCOMP}, \text{BR}]$, and define the hierarchy $\mathcal{G}$ by $\mathcal{G} = \bigcup_{n \in \mathbb{N}} \mathcal{G}^n$ □

The next lemma states some basic properties of the classes in the hierarchy.

LEMMA 3. (i) we have $f(\vec{x}) \leq G_n(\vec{x})$ for any $f \in \mathcal{G}^n$; (ii) the graph of G_n, i.e. the relation $G_n(\vec{x}) = y$, is in $\mathcal{G}_\star^0$; (iii) the relations of $\mathcal{G}^n$ are closed under Boolean connectives and quantification bounded by a function; (iv) $\Delta_0^{\mathbb{N}} \subseteq \mathcal{G}_\star^0$.

Proof. We prove (i) by induction on a build-up of f from the functions in the algebra $[I, C_0, \hat{\mathsf{S}}, G_n; \text{BCOMP}, \text{BR}]$. When f is one of the initial functions in the algebra, we have $f(\vec{x}) \leq G_n(\vec{x})$. (In particular we have $\hat{\mathsf{S}}(x, y) \leq G_n(x, y)$ in the case when $n = 0$.) Assume f is generated by bounded composition, i.e. defined by

the scheme BCOMP, then we have $f(\vec{x}) \leq G_n(\vec{x})$ straightaway from the induction hypothesis. Assume f is generated by bounded recursion, i.e. defined by the scheme BR, then we also have $f(\vec{x}) \leq G_n(\vec{x})$ straightaway from the induction hypothesis. We skip (ii), (iii) and (iv). □

The next lemma relates the hierarchy to the small Grzegorczyk classes.

LEMMA 4. (i) $\mathcal{G} = \mathcal{E}^2$; (ii) $\mathcal{E}^0$ and $\mathcal{G}^0$ are incomparable, viz. $\mathcal{E}^0 \not\subseteq \mathcal{G}^0$ and $\mathcal{G}^0 \not\subseteq \mathcal{E}^0$; (iii) $\mathcal{E}^1$ and $\mathcal{G}^1$ are incomparable; (iv) $\mathcal{G}^0_\star \subseteq \mathcal{E}^0_\star$; (v) $\mathcal{E}^1_\star \subseteq \mathcal{G}^1_\star$.

Proof. The inclusion $\mathcal{G} \subseteq \mathcal{E}^2$ is obvious. Assume $f \in \mathcal{E}^2$. By Lemma 1, we have $f(\vec{x}) \leq p(\vec{x})$ for some polynomial p. Then we also have $f(\vec{x}) \leq G_n(\vec{x})$ for a sufficiently large n. It follows that we have $f \in \mathcal{G}^n$ for some n, whence $\mathcal{E}^2 \subseteq \mathcal{G}$. This proves (i). To see that we have $\mathcal{E}^0 \not\subseteq \mathcal{G}^0$, let e.g. $f(x) = x + 17$ and note that $f \in \mathcal{E}^0$, but $f \notin \mathcal{G}^0$ by Lemma 3 (i). To see that $\mathcal{G}^0 \not\subseteq \mathcal{E}^0$, notice that $G_0(x, y) \notin \mathcal{E}^0$ by Lemma 1. To see that $\mathcal{E}^1 \not\subseteq \mathcal{G}^1$, let e.g. $f(x) = x \times 17$ and note that $f \in \mathcal{E}^1$, but $f \notin \mathcal{G}^1$ by Lemma 3 (i). To see that $\mathcal{G}^1 \not\subseteq \mathcal{E}^1$ note that the function $x^2 \in \mathcal{G}^1$, but, by Lemma 1, it is not in $\mathcal{E}^1$. This proves (ii) and (iii). (iv) follows from a result of Bel'tyukov, see Kutylowski [17]. (v) is a consequence of the fact that for any $f \in \mathcal{E}^1$, $f(\vec{x})$ is almost everywhere less than $G_1(\vec{x}) \in \mathcal{G}^1$. □

DEFINITION 5. We define the class $\mathcal{P}$ of number-theoretic functions by $\mathcal{P} = [I, C_1; \text{COMP}, \text{PR}]$. □

Note that the definition of the class $\mathcal{P}$ does not embody explicit bounds.

PROPOSITION 6 (BASIC FUNCTIONS). The following functions belong to $\mathcal{P}$. (i) C_0,C_1; (ii) for each fixed $k \in \mathbb{N}$ the function $C_k(x)$ where $C_k(x) = k$ if $x \geq k$, and $C_k(x) = x$ otherwise (so $C_k(x)$ is the almost everywhere constant function yielding k for all but finitely many values of x); (iii) P(x) (predecessor); (iv) $x \dot{-} y$ (modified subtraction); (v) the function $x \oplus_{y+1} 1$ where $x \oplus_{y+1} 1 = 0$ if $x \geq y$, and $x \oplus_{y+1} 1 = x + 1$ otherwise; (vi) $\hat{\mathsf{S}}(x, y)$; (vii) $c(x, y, z)$ where $c(x, y, z) = x$ if $z = 0$ and $c(x, y, z) = y$ if $z \neq 0$; $(viii)$ $\max(x, y)$.

Proof. We will argue that the functions of the lemma can be defined from projections and the constant 1 by composition and primitive recursion.

To define the constant function C_0 is slightly nontrivial. Define g by primitive recursion such that $g(x, 0) = x$ and $g(x, y + 1) = y$. Then we can define the predecessor P from g since $P(x) = g(x, x)$. Further, we can define the constant function C_0 by $C_0 = P(1)$. This proves that (i) and (iii) holds; (iv) holds since we have $x \dot{-} 0 = x$ and $x \dot{-} (y + 1) = P(x \dot{-} y)$. For (vii), $c(x, y, 0) = I^2_1(x, y)$ and $c(x, y, z + 1) = I^4_2(x, y, c(x, y, z), z))$, is an instance of PR. $(viii)$ follows by $\max(x, y) = c(x, y, C_1 \dot{-} (x \dot{-} y))$; (v) from $x \oplus_{m+1} 1 = c(0, m \dot{-} ((m \dot{-} x) \dot{-} 1), m \dot{-} x)$. (ii) remains; thus let $M^0(z) = 0$ and $M^{n+1}(z) = M^n(z) \oplus_{z+1} 1$. We can define M^n for any fixed $n \in \mathbb{N}$. Further, $M^n(z) = n \pmod{z + 1}$. Hence $C_k(x) = c(x, M^k(x), P^{(k)}(x))$ where $P^{(k)}$ is the predecessor function repeated k times.

□

COROLLARY 7. $\mathcal{P} = \mathcal{G}^0$.

Proof. $G_0 = \max(\vec{x}, C_1)$. Since all functions of $\mathcal{P}$ are non-increasing, it is easy to see that replacing PR and COMP by BR and BCOMP in the definition of $\mathcal{P}$ leaves the resulting class of functions unchanged. By Proposition 6 (*vi*) and (*viii*) and the fact that $C_1 = G_0(C_0) \in \mathcal{G}^0$ the corollary follows. □

LEMMA 8 (NORMAL FORM). For any $f \in \mathcal{G}^n$ there exists $f' \in \mathcal{P}$ such that $f(\vec{x}) = f'(G_n(\vec{x}), \vec{x})$.

Proof. Let $f \in \mathcal{G}^n$. By an easy induction on a (any) definition of f, using the fact that any intermediate function is bounded by $G_n(\vec{x})$ (Lemma 3). □

COROLLARY 9. If $\mathcal{G}^0_\star \subseteq \Delta_0^{\mathbb{N}}$, then $\mathcal{E}^2_\star \subseteq \Delta_0^{\mathbb{N}}$.

Proof. Assume $\mathcal{G}^0_\star \subseteq \Delta_0^{\mathbb{N}}$ and $R \in \mathcal{E}^2_\star$. We will prove $R \in \Delta_0^{\mathbb{N}}$. By Lemma 8 we have $R' \in \mathcal{G}^0_\star$ and a fixed $n \in \mathbb{N}$ such that $R(\vec{x})$ holds iff $R'(G_n(\vec{x}), \vec{x}) > 0$. Lemma 3 (*iii*) says that the relation $G_n(\vec{x}) = y$ is in $\mathcal{G}^0_\star$. Let p be a polynomial expressible in the theory $\Delta_0^{\mathbb{N}}$ such that $p(\vec{x}) \geq G_n(\vec{x})$. Now, $R(\vec{x})$ is true iff $\exists y \leq p(\vec{x})\,[\, G_n(\vec{x}) = y \ \wedge \ R'(y, \vec{x})\,]$, thus $R \in \Delta_0^{\mathbb{N}}$. □

4 The Level-0 Typed λ-Calculus and the Hierarchy $\mathcal{L}$

The following are standard definitions and notation from type-theory.

0 is a *type*; $\vartheta \to \vartheta'$ is a type if ϑ and ϑ' are types; $\vartheta \times \vartheta'$ is a type if ϑ and ϑ' are types. $\vartheta_1, \ldots, \vartheta_n \to \vartheta'$ denotes the type $\vartheta_1 \to (\vartheta_2 \to \cdots (\vartheta_n \to \vartheta') \cdots)$.

We say a type ϑ has *level* n when $\mathrm{lev}(\vartheta) = n$ where $\mathrm{lev}(\mathbf{0}) = 0$; $\mathrm{lev}(\vartheta \to \vartheta') = \max(\mathrm{lev}(\vartheta) + 1, \mathrm{lev}(\vartheta'))$; and $\mathrm{lev}(\vartheta \times \vartheta') = \max(\mathrm{lev}(\vartheta), \mathrm{lev}(\vartheta'))$.

Also each type ϑ may be uniquely written as $\vartheta_1, \ldots, \vartheta_n \to \vartheta_{n+1}$ where $\mathrm{lev}(\vartheta_{n+1}) = 0$, then the *arity of* ϑ, $\mathrm{ar}(\vartheta) = n$.

DEFINITION 10. Define the *terms* of *the standard level-***0** *λ-calculus* by ($M : \vartheta$ means M is a term of type ϑ)

- *(Variables)* We have a set X of variables $x_0^\vartheta, x_1^\vartheta, x_2^\vartheta, \ldots$ for each type ϑ with $\mathrm{lev}(\vartheta) = 0$. An $x_i^\vartheta \in X$ is a term of type ϑ.
- *(λ-abstraction.)* $(\lambda x.M) : \vartheta \to \vartheta'$ is a term if $x : \vartheta$ and $M : \vartheta'$ is a term.
- *(Application)* (MN) is a term of type ϑ' if $M : \vartheta \to \vartheta'$ and $N : \vartheta$ are terms.
- *(Product)* $\langle M, N\rangle : \vartheta \times \vartheta'$ is a term if $M : \vartheta$ and $N : \vartheta'$ are terms.
- *(Projections)* $(\mathsf{fst}M : \vartheta)$ is a term $((\mathsf{snd}M) : \vartheta')$ if $M : \vartheta \times \vartheta'$ is a term.

The reduction rules of the calculus are $(\lambda x M)N \to M[x := N]$ (β-conversion); $\mathsf{fst}\langle M, N\rangle \to M$; and $\mathsf{snd}\langle M, N\rangle \to N$. We will study extensions L^- and L of the standard typed λ-calculus. Both extensions will normalise, and we will say that the two terms M and N are *equal*, in symbols $M = N$, when M and N reduce to the same normal form. $M \equiv N$ means syntactical identity. $M \in N$ means M is a sub-term of N. The set of free variables in M is denoted $\mathrm{FV}(M)$ □

Note that by *level-***0** typed λ-calculus, we mean that only variables of level zero type is included; hence only level zero abstraction is allowed.

We assume the reader is familiar with the typed λ-calculus, and we will use the standard conventions from the literature: $MNPQ$ abbreviates $(((MN)P)Q)$ and $\lambda xyz.M$ means $\lambda x.(\lambda y.(\lambda z.M))$. Superscripts indicating type is usually omitted.

For more on the λ-calculus see e.g. [9] or [25]

Definition 11. The calculus L^- is the standard level-**0** λ-calculus extended with the constant $\mathbf{1} : \mathbf{0}$, and for each type ϑ with $\mathrm{lev}(\vartheta) = 0$ the *recursor* R_ϑ of type $\mathbf{0} \to \vartheta \to \vartheta, \vartheta, \mathbf{0} \to \vartheta$.

Extend the calculus L^- to the calculus L by introducing constants $0 : \mathbf{0}$ (zero), $\mathsf{S} : \mathbf{0} \to \mathbf{0}$. The extended reduction-relation is given by adding the rules

$$\mathbf{1} \to \mathsf{S}0 \quad ; \quad \mathsf{R}_\vartheta HG0 \to G \quad \text{and} \quad \mathsf{R}_\vartheta HG(\mathsf{S}N) \to HN(\mathsf{R}_\vartheta HGN)$$

We use $\underline{n}$ to denote the *numeral* $\mathsf{S}^{(n)}0$. Here $\mathsf{S}^{(n)}0$ is the standard shorthand for iterated application, i.e. $M^{(0)}N \equiv N$; $M^{(n+1)}N \equiv M(M^{(n)}N)$. Num denotes the set of numerals. □

The above definitions may be summarised as follows:

The standard level-**0** λ-calculus is the fragment of standard typed λ-calculus with products where all higher-type variables are removed.

L^- is a very restricted fragment of Gödels T (identify **1** with $\mathsf{S}0$); reduction rules for the recursors are omitted and higher-type variables are banned. Also no successor constant is included. e.g. the term $\mathsf{R}_\vartheta(M, N, \mathbf{1})$ is irreducible in the calculus L^- when M and N are. The calculus L^- should be thought of as a calculus for defining functions.

The calculus L is the fragment of Gödels T where only the type-restrictions are upheld (if one temporarily 'forgets' the constant **1** and the rule $\mathbf{1} \to \mathsf{S}0$). It is well known that any closed L-term of type **0** normalises to a unique numeral. Thus, a closed term M of type $\mathbf{0}^n \to \mathbf{0}$ defines a unique function $f : \mathbb{N}^n \to \mathbb{N}$, and the value $f(\vec{n})$ can be computed by normalising the term $M\underline{n}$. The calculus L should be thought of as a calculus for computing functions.

The calculus L captures the primitive recursive functions, and thus is far too powerful for our purposes. However, if we disallow occurrences of the successor S in the term M, the class of functions definable is of course severely restricted. At a first glance it is easy to think that most interesting functions are thrown out with the successor function. We will see that this is not the case, indeed, any $0-1$ valued function computable by a Turing machine working in linear space can be defined by an L^--term, and then computed in the L-calculus.

In the following we will also see that a further refinement of the L^--terms into a hierarchy of terms, in turn induces an interesting sub-recursive hierarchy.

Definition 12. A function $f : \mathbb{N}^k \to \mathbb{N}$ *is defined by* M if $M : \mathbf{0}, \ldots, \mathbf{0} \to \mathbf{0}$ and $\forall \vec{n} \in \mathbb{N}^n \big(M\underline{\vec{n}} =_L \underline{f(\vec{n})}\big)$. A term defining f is denoted M_f. If $M : \mathbf{0}, \ldots, \mathbf{0} \to \mathbf{0}$, then f_M is the unique function defined by $\underline{f_M(\vec{n})} =_L M\underline{\vec{n}}$.

Define the *product rank* $\pi(\vartheta)$ of a type ϑ by $\pi(\mathbf{0}) = 0$;
$\pi(\vartheta \to \vartheta') = \max(\pi(\vartheta), \pi(\vartheta'))$ and $\pi(\vartheta \times \vartheta') = \pi(\vartheta) + \pi(\vartheta') + 1$.

Define the *product rank* $\pi(M)$ of the L^--term M by

$$\pi(M) \stackrel{\text{def}}{=} \max\{\pi(\vartheta) \mid M \text{ has a sub-term of type } \vartheta\}$$

Let $\mathcal{L}^n \stackrel{\text{def}}{=} \{f_M \mid \pi(M) \leq n\}$ and define the hierarchy $\mathcal{L}$ by $\mathcal{L} = \bigcup_{n\in\mathbb{N}} \mathcal{L}^n$. □

The next lemma shows that a surprisingly large arsenal of functions is definable by terms of product rank 0.

PROPOSITION 13. We have $\mathcal{P} \subseteq \mathcal{L}^0$, and thus all the basic functions given in Proposition 6 belong to $\mathcal{L}^0$.

Proof. We recall that $\mathcal{P} = [I, C_1; \text{COMP}, \text{PR}]$. The constant function C_1 is defined by the initial L^--term $\mathbf{1}$. The projection function $I^n_i(x_1, \ldots, x_n) = x_i$ is defined by the L^--term $\lambda x_1 \cdots x_n.x_i$ (for any fixed $i, n \in \mathbb{N}$ such that $1 \leq i \leq n$). The set of functions defined by L^--terms of product rank 0 is obviously closed under composition and primitive recursion. Thus, it follows that $\mathcal{P} \subseteq \mathcal{L}^0$. □

Note that if the relation $f(\vec{x}) = y$ is in $\mathcal{L}^n_\star$ then $f \in \mathcal{L}^n$, since $f(\vec{x}) = \mu y \leq \max(\vec{x}, 1)[f(\vec{x}) = y]$ and we can show that $\mathcal{L}^n$ is closed under bounded μ-operator. Hence $\mathcal{L}^i_\star = \mathcal{L}^j_\star$ iff $\mathcal{L}^i = \mathcal{L}^j$.

The aim of this section is to prove the following theorem.

THEOREM 14. We have $\mathcal{G}^n_\star = \mathcal{L}^n_\star$ for every $n \in \mathbb{N}$.

Note that $G_n(b)$ equals the greatest number in base $\max(b, 1)+1$ of of $(n+1)$-digits, and that a term containing a sub-term of type $\mathbf{0}^n$, will have product rank at least n.

DEFINITION 15. Let $\mathbf{0}^n$ be defined by $\mathbf{0}^0 = \mathbf{0}$; $\mathbf{0}^{n+1} = \mathbf{0} \times \mathbf{0}^n$.

For $0 \leq n$ we define the set Num^n of *generalised numerals of product rank n* inductively by

$$(i)\ \text{Num}^0 = \text{Num} \ \text{ and } \ (ii)\ \text{Num}^{n+1} = \{\langle N, M\rangle \mid N \in \text{Num}, M \in \text{Num}^n\}$$

It is readily checked from previous definitions that $N \in \text{Num}^n$ satisfies $N : \mathbf{0}^n$ and $\pi(N) = n$. We write $\langle N_0, \ldots, N_n\rangle$ or $\langle \vec{N}\rangle$ for $\langle N_1, \ldots \langle N_{n-1}, N_n\rangle \ldots\rangle$.

Next, define a map $[\cdot]^n_b : \{m \mid m \leq G_n(b)\} \to \text{Num}^n$, for each $b \geq 1$ and $n \geq 0$, by $[m]^n_b = \langle \underline{a}_0, \ldots, \underline{a}_n\rangle$, where $a_0, \ldots, a_n \leq b$ are the unique natural numbers such that $m = \Sigma^n_{i=0} a_i(b+1)^i$. Clearly, each $[\cdot]^n_b$ is an injection onto a set $Num^n_b \subseteq \text{Num}^n$. Also, whenever $[m]^n_b$ is written it is tacitly assumed that $m \leq (b+1)^{n+1} - 1$. We may also write $M[\vec{\alpha} := m]$ to indicate that some list of (free) variables in M are substituted for the numerals $a_0, \ldots, a_n$ such that $m = \Sigma^n_{i=0} a_i(b+1)^i$ when b and n are clear from context. □

We first attack the $\mathcal{G}^n_\star \subseteq \mathcal{L}^n_\star$ direction of Theorem 14. We need a technical lemma.

LEMMA 16. Let β be a fixed variable of type $\mathbf{0}$. For any $f \in \mathcal{P}$ there exists an L^--*term* $F^n : \mathbf{0}^n, \ldots, \mathbf{0}^n \to \mathbf{0}^n$ of product rank n, such that

$$(F^n[\beta := \underline{b}])[m_1]^n_b \cdots [m_k]^n_b =_L [f(\vec{m})]^n_b$$

for all $b, \vec{m}$ where $\max(\vec{m}, 1) \leq (b+1)^n - 1$.

Proof. For $n = 0$ we have $\mathcal{G}^0 = \mathcal{P} \subseteq \mathcal{L}^0$ by Corollary 7 and Proposition 13. For $n \geq 1$ The proof is by induction, for all n simultaneously, on a (any) definition of f in $\mathcal{P}$. The induction is quite straightforward due to the non-increasing nature of $\mathcal{P}$. The only involved step is when f is defined from h and g by primitive recursion.

By the I.H, we then have terms H' and G' satisfying the hypothesis of the lemma. Let $H = H'\vec{X}$ and $G = G'\vec{X}$. Then $H : \mathbf{0}^n, \mathbf{0}^n \to \mathbf{0}^n$ and $G : \mathbf{0}^n$ We need to construct a term $R_{H,G}$ which satisfies $R_{H,G}[m]^n_b[m_1]^n_b \cdots [m_k]^n_b =_L H\{m\}^n_b G$ where $H\{0\}G \equiv G$ and $H\{m+1\}^n_b G \equiv H[m]^n_b(H\{m\}^n_b G)$. For the construction we need terms with the following properties: A term $<_\beta$: $\mathbf{0}^n, \mathbf{0}^n \to \mathbf{0}$ which satisfies $<_{\underline{b}} [m_0]^n_b[m_1]^n_b =_L \underline{1}$ if $m_0 < m_1$ and $<_{\underline{b}} [m_0]^n_b[m_1]^n_b =_L \underline{0}$ else (we skip the details). Furthermore, define a term $\tilde{H}$ by

$$\tilde{H} \equiv \lambda\gamma.\mathsf{R}\big(\lambda\delta_0\delta_1.H\langle\alpha_0, \ldots, \alpha_n\rangle\gamma\big)\gamma(< \langle\alpha_0, \ldots, \alpha_n\rangle\mu)$$

Here $\vec{\alpha}$, μ, γ are all fixed variables, while the δ_i are dummy variables. Notice that the $\vec{\alpha}$ and μ are free in $\tilde{H}$ It is easy to verify that

$$\tilde{H}[\vec{\alpha} := m_0; \mu := [m_1]^n_b] =_L \begin{cases} \lambda\gamma.\gamma & \text{if} \quad m_1 < m_0 \\ \lambda\gamma.H[m]^n_b\gamma & \text{else} \end{cases}$$

Now construct a term $\tilde{H}^k$ by induction on k by

$$\tilde{H}^0 \equiv R\big(\lambda\alpha_0.(\tilde{H}[\alpha_0 := \hat{\mathsf{S}}\alpha_0\beta])\big)\big((\tilde{H}[\alpha_0 := \underline{0}])\gamma\big)\beta \text{ and}$$

$$\tilde{H}^{k+1} \equiv R\big(\lambda\alpha_{k+1}\gamma.(\tilde{H}^k[\alpha_{k+1} := \hat{\mathsf{S}}\alpha_{k+1}\beta])\big)\big(\tilde{H}^k[\alpha_{k+1} := \underline{0}]\big)\beta$$

It is important to note that the α_i are the same fixed variables as in the construction of $\tilde{H}$. It is $k = n$ which is useful for us, even if the definition is well defined for all k. The term $\hat{\mathsf{S}}$ is here a (any) term defining the function $\hat{\mathsf{S}}$; $\underline{0}$ is formally shorthand for any L^--term defining the function C_0. It is readily checked from the definition that $\pi(\tilde{H}^k) = n$ for all k

It follows by a straightforward induction on k (accompanied by rather tedious book-keeping) that $(\lambda\mu\vec{X}.\tilde{H}^n G[\beta := \underline{b}])[m]^n_b\vec{N} =_L (H'\vec{N})^{(m)}\{m\}(G'\vec{N})$. The induction step now follows directly. □

Informally, the term $(\lambda\mu\vec{X}.\tilde{H}G[\beta := \underline{b}])[m]^n_b\vec{N}$ reduces to a term $\lambda\vec{X}(\tilde{H}^{((b+1)^{n+1})}G)\vec{N}$ where each $\tilde{H}$ 'knows it's position p' in the row by the information carried by the α_i. It then reduces to $\lambda\gamma.\gamma$ if it is not supposed to be there, and $\lambda\gamma.H[p]^{n+1}_b\gamma$ if it should.

We obtain $\mathcal{G}^n_\star \subseteq \mathcal{L}^n_\star$ as a corollary.

Proof (of $\mathcal{G}^n_\star \subseteq \mathcal{L}^n_\star$). Let $\mathcal{G}^n_\star \ni f : \mathbb{N}^k \to \mathbb{N}$, and choose $f' \in \mathcal{P}$ as guaranteed by Lemma 8, i.e. $f(\vec{m}) = f'(G_n(\vec{m}), \vec{m})$. Let F' be a term as in Lemma 16, $b = \max\{\vec{m}\}$, and define $F \in L^-$ by

$$F \equiv \lambda\vec{x}.(\mathsf{fst}(\lambda\beta.(F'\langle\beta,\ldots,\beta\rangle\langle x_1,[0]^n_b\rangle\cdots\langle x_k,[0]^n_b\rangle)\max(\vec{x})$$

Here $\max\vec{x}$ is shorthand for a (any) term defining the function max, $[0]^n_b \equiv \langle\underline{0},\ldots,\underline{0}\rangle$, and we obtain

$$F\underline{\vec{m}} =_L \mathsf{fst}(\lambda\beta.(F'\langle\beta,\ldots,\beta\rangle\langle m_1,[0]^n_b\rangle\cdots\langle m_k,[0]^n_b\rangle)\max(\underline{\vec{m}}) =_L$$
$$\mathsf{fst}\big((F'[\beta := \underline{b}])[G_n(b)]^n_b[m_1]^n_b\cdots[m_k]^n_b\big) =_L \mathsf{fst}[f(\vec{m})]^n_b =_L \underline{f(\vec{m})}$$

The second equality is $[G_n(b)]^n_b \equiv \langle\underline{\vec{b}}\rangle$, the third equality is Lemmata 8 and 16 and the last follows from $\mathsf{fst}[m]^n_b = \underline{m}$ when $m \leq b$. □

We now turn to the other direction, $\mathcal{L}^n_\star \subseteq \mathcal{G}^n_\star$, of Theorem 14, and prove the stronger result $\mathcal{L}^n \subseteq \mathcal{G}^n$. Thus given a term $M : \mathbf{0},\ldots,\mathbf{0} \to \mathbf{0}$, we must show that the function f_M defined by $f_M(\vec{n}) = k$ iff $M\underline{\vec{n}} =_L \underline{k}$. The converse does not hold, since $f \in \mathcal{L}^n$ implies f non-increasing, while e.g. $G_2 \in \mathcal{G}^2$ is not.

The natural way to proceed, is by an inductive argument of some sort. If $M : \mathbf{0},\ldots,\mathbf{0} \to \mathbf{0}$ is an L^--term, we may assume w.l.o.g. that M is L^--normal i.e. no redex $\mathsf{fst}\langle N_0, N_1\rangle$ ($\mathsf{snd}\langle N_0, N_1\rangle$) or $(\lambda x.N_0)N_1$ occurs in M. We will in the continuation also assume that all abstractions $\lambda x.M_0$ are unique with respect to the variable, something which may always be achieved with a suitable renaming of bound variables.

Under these assumptions it is possible to show that any sub-term $N \in M$ must be on one of the following forms: (B1) $N \equiv x$; (B2) $N \equiv \mathbf{1}$; (PRJ) $N \equiv \mathsf{fst}N_0$ ($\mathsf{snd}N_0$) where $\mathrm{lev}(N_0) = 0$ (i.e. $N_0 : \vartheta_N$ and $\mathrm{lev}(\vartheta_N) = 0$); (PRD) $\langle N_0, N_1\rangle$ where $\mathrm{lev}(N_0) = \mathrm{lev}(N_1) = 0$; (ABS) $N \equiv \lambda x.N_0$ where $\mathrm{lev}(N_0) \leq 1$; (REC) $N \equiv \mathsf{R}HGN$.

This all follows from type considerations, except for item *(REC)*. A closed L^--term may actually have the form $\mathsf{R}HG$ and still be closed type $\mathbf{0} \to \mathbf{0}$. In that case we may replace any such sub-term by $\lambda x.\mathsf{R}HGx$ without changing the defined function.

In order to make this work, we will need to attach some meaning to 'f_N' for 'most' sub-terms $N \in M$; since in general, they are neither closed nor typed $\mathbf{0},\ldots,\mathbf{0} \to \mathbf{0}$. In order to get around this obstacle we first address the typing issue.

Consider a member $\sigma = n_1,\ldots,n_k$ of $\mathbb{N}^{<\omega}$, i.e. a finite sequence of natural numbers, and let $\sigma\sigma'$ denote *concatenation* of sequences. Armed with pairing such a sequence may be represented a number of ways, inductively defined by $n = n$ and $\sigma\sigma' = \langle\sigma,\sigma'\rangle$. Obviously every sequence of length $|\sigma| = k$ is representable by $k-1$ applications of such pairings, and the representation is not unique. Indeed we get one representation for each type ϑ with $\pi(\vartheta) = k-1$ and $\mathrm{lev}(\vartheta) = 0$. Also if $z > \max\{n \mid n \in \sigma\}$ we may code such a representation as a number $\lceil\sigma\rceil^\vartheta_z$ with the following inductive schema

$$\lceil n\rceil = n \quad\text{and}\quad \lceil\langle\sigma,\sigma'\rangle\rceil^\vartheta_z = \lceil\sigma\rceil^\vartheta_z \cdot (G_k(z)+1) + \lceil\sigma'\rceil^\vartheta_z \text{ where } k = |\sigma'|$$

That $\lceil \sigma \rceil_z^\vartheta \leq G_{\pi(\vartheta)}(z)$ is easily verified.

Every closed L-term $N : \vartheta$ with $\mathrm{lev}(\vartheta) = 0$ is identified in a canonical way with a represented sequence σ, e.g. $\langle\langle n_1, n_2\rangle, n_3\rangle$ with $\langle\langle \underline{n_1}, \underline{n_2}\rangle, \underline{n_3}\rangle$.

We are now ready to sketch a proof of $\mathcal{L}^n \subseteq \mathcal{G}^n$

Proof (of $\mathcal{L}^n \subseteq \mathcal{G}^n$ (sketch)). $\mathcal{O}$ denotes a set-theoretic bijection between a finite subset of X (the set of variables of the calculus L^-) and an initial segment of $\mathbb{N}\setminus\{0\}$, hence $\mathcal{O}$ induces an ordering of $\mathrm{dom}(\mathcal{O})$. We next define inductively over the build-up of a term M an interpretation $[\![M]\!]^{\mathcal{O}} \in \mathcal{G}$, when $\mathrm{FV}(M) \subseteq \mathrm{dom}(\mathcal{O})$. It has the desired property that

$$\underline{f_M(\vec{n})} \stackrel{\mathrm{def}}{=} M\underline{\vec{n}} = \underline{[\![M]\!]^{\emptyset}(\max(\vec{n}, 1), \vec{n})} \quad \text{and} \quad [\![M]\!]^{\mathcal{O}} \in \mathcal{G}^{\pi(M)}$$

for any closed normal $M : \mathbf{0}, \ldots, \mathbf{0} \to \mathbf{0}$.

A fully detailed proof uses the fact that $G_n(k) \geq \lceil N \rceil_k^\vartheta$, where $N : \vartheta$ is closed, $\mathrm{lev}(\vartheta) = 0$ and $k \geq \max\{m \mid \underline{m} \in N\}$. Combined with the non-increasing nature of numerals in L-terms, we are able to interpret open and/or type $\vartheta_1, \ldots, \vartheta_n \to \vartheta_{n+1}$, in a meaningful way. This enables us to carry out the induction step for sub-terms N of M by interpreting e.g. $N : \mathbf{0} \times \mathbf{0} \to \mathbf{0} \times \mathbf{0}$ as a function $f : \mathbb{N}^{2+k} \to \mathbb{N}^2$, where k depends on $\mathcal{O}$. The key point is that $G_n(\vec{x})$ is exactly large enough to encode a sequence of length $n + 1$. Piecing things together, the result follows. □

Why do we think $\mathcal{L}_\star^n = \mathcal{G}_\star^n$ is worth mentioning? Simply because it provides a fresh (to our knowledge) point of attack for the original $\mathcal{E}_\star^0$ vs. $\mathcal{E}_\star^2$ problem. λ-calculi is very well known, thoroughly studied and of intuitive nature. It constitutes a relatively concrete mathematical theory, possibly suited for settling this long open problem.

An interesting possible generalisation of the hierarchies $\mathcal{G}^n$ and $\mathcal{L}^n$ arises from the more general function G_ϑ for all types. Define $G'_{\mathbf{0}}(b) = b + 1$; $G'_{\vartheta\times\vartheta'}(b) = G'_\vartheta(b)G'_{\vartheta'}(b)$ and $G'_{\vartheta\to\vartheta'}(b) = G'_{\vartheta'}(b)^{G'_\vartheta(b)}$, and let $G_\vartheta = G'_\vartheta - 1$. Then our G_n corresponds with G_ϑ when $\pi(\vartheta) = n$ and $\mathrm{lev}(\vartheta) = 0$. We suspect that this construction gives rise to (transfinite) hierarchies $\mathcal{G}^\vartheta$ and $\mathcal{L}^\vartheta$ based on a suitable well-ordering of the types.

References

1. A. Beckmann and A. Weiermann. *Characterizing the elementary recursive functions by a fragment of Gödel's T.* Arch. Math. Logic 39, No.7, 475-491 (2000).
2. S. J. Bellantoni and S. Cook. *A New Recursion-Theoretic Characterization of the Polytime Functions.* Computational Complexity, 2 (1992), 7–110.
3. A.P. Bel'tyukov. *A machine description and the hierarchy of initial Grzegorczyk classes* Zap. Naucn. Sem. Leninigrad. Otdel. May. Inst. Steklov. (LOMI) 88 (1979) 30-46; J. Soviet Math. 20 (1982)
4. P. Clote. *Computation Models and Function Algebra.* In: Handbook of Computability Theory. Ed Griffor, ed., Elsevier 1996.

5. M.-A. Esbelin, M. More. *Rudimentary relations and primitive recursion: A toolbox.* Theoretical Computer Science 193 (1998) 129-148.
6. R. Gandy. *Some relations between classes of low computational complexity.* Bulletin of London Mathematical Society (1984) 127-134.
7. A. Grzegorczyk. *Some classes of recursive functions.* Rozprawy Matematyczne, No. IV, Warszawa, 1953.
8. K. Harrow *Sub-elementary classes of functions and relations.* Doctoral Dissertation, New York University, Department of Mathematics 1973.
9. J.R. Hindley and J.P. Seldin. *Introduction to Combinators and λ-caclulus.* London Mathematical Society, Student Texts **1**, Cambrigde University Press 1984.
10. N.D. Jones. *The expressive power of higher-order types or, life without CONS.* J. Functional Programming 11 (2001) 55-94.
11. N.D. Jones. *LOGSPACE and PTIME characterized by programming languages.* Theoretical Computer Science 228 (1999) 151-174.
12. L. Kristiansen. *Neat function algebraic characterizations of LOGSPACE and LINSPACE.* Computational Complexity (accepted).
13. L. Kristiansen and K-H. Niggl. *On the computational complexity of imperative programming languages.* Theoretical Computer Science 318 (2004), 139-161.
14. L. Kristiansen and P.J. Voda. *Complexity classes and fragments of C.* Information Processing Letters 88 (2003), 213-218.
15. L. Kristiansen and P.J. Voda. *The surprising power of restricted programs and Gödel's functionals.* In M. Baaz and J.A. Makowsky (Eds.): Computer Science Logic, LNCS 2803, pp. 345-358, Springer, 2003.
16. L. Kristiansen and P.J. Voda. *Programming languages capturing complexity classes.* (Submitted.)
17. M. Kutylowski. *Small Grzegorczyk classes.* J. London Math. Soc. (2) 36 (1987) 193-210
18. D. Leivant. *Intrinsic theories and computational complexity.* In "Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94", Leivant (ed.), 177-194, Springer, 1995.
19. P. Odifreddi. *Classical Recursion Theory, Vol II.* North-Holland, 1999.
20. J.B. Paris and A. Wilkie. *Counting problems in bounded arithmetic.* In "Methods in mathematical logic. Proceedings, Caracas 1983". Lecture Notes in Mathematics 1130, pp. 317-340, Springer, 1985.
21. R.W. Ritchie. *Classes of predictably computable functions.* Trans. Am. Math. Soc. 106, 139-173 (1963).
22. H.E. Rose *Subrecursion. Functions and hierarchies.* Clarendon Press, Oxford, 1984.
23. H. Simmons. *The realm of primitive recursion.* Arch. Math. Logic 27 (1988), 177-188.
24. H. Simmons. *Derivation and computation. Taking the Curry-Howard correspondence seriously.* Cambridge Tracts in Theoretical Computer Science. 51. Cambridge: Cambridge University (2000).
25. Terese. *Term Rewriting Systems*, (M. Bezem, J. W. Klop, R. de Vrijer, editors), Cambridge University Press (2003)

The Flow of Data and the Complexity of Algorithms

Lars Kristiansen[1,2] and Neil D. Jones[3]

[1] Faculty of Engineering, Oslo University College,
Cort Adelers gate 30,
N-0254 Oslo, Norway
larskri@iu.hio.no
http://www.iu.hio.no/~larskri
[2] Department of Mathematics, University of Oslo,
Postboks 1053, Blindern,
N-0316 Olso, Norway
[3] Department of Computer Science,
University of Copenhagen,
Copenhagen, Denmark
neil@diku.dk
http://www.diku.dk/~neil

1 Introduction

Let C be a program written in a formal language in order to be executed by some kind of machinery. A *statement about C* might be true or false and has the form $\mathtt{C}:M$. For the time being, just consider the statement $\mathtt{C}:M$ as a collection of data yielding information about the resources required to execute C; and if we know that $\mathtt{C}:M$ is true (or false), we know something useful when it comes to determine the computational complexity of C. Let Γ be a set of statements, and let $\Gamma \models \mathtt{C}:M$ denote that $\mathtt{C}:M$ will be true if all the statements in Γ are true. (The statements in Γ might say something about the computational complexity of the subprograms of C.) If $\Gamma = \emptyset$, we will simply write $\models \mathtt{C}:M$.

Overview: We will define the semantic relation $\models$ mathematically, and then introduce a corresponding provability relation $\vdash$ by a syntactic proof calculus for deriving statements of the form $\Gamma \vdash \mathtt{C}:M$. We have a soundness theorem for the calculus, i.e., we can prove $\Gamma \vdash \mathtt{C}:M \Rightarrow \Gamma \models \mathtt{C}:M$, and we are also able to prove an important completeness property of the calculus.

Our proof calculus is based on a careful and detailed analysis of the relationship between the resource requirements of a computation and the way data might flow during the computation. This analysis extends and refines the insights acquired by researches as done by Bellantoni & Cook ([2] normal and safe variables), Simmons ([16], active and dormant variables), Leivant ([14], ramification), and in particular, Kritiansen & Niggl ([9, 10] measures). The insight that there is a relationship between the absence and presence of successor-like functions and the computational complexity of a program is a part of the foun-

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 263–274, 2005.

dation of our calculus, see e.g., Jones [5, 6], Kristiansen & Voda [11, 12], and Kristiansen [8].

Even if our research builds on, and is comparable to, the research discussed above, it has a different emphasis, e.g., we are not aiming at implicit characterisations of complexity classes (even if such characterisations will be easy corollaries of our results). The overall goal of our research is to achieve a better understanding of the relationship between syntactical constructions in natural programming languages and the computational resources required to execute the programs.

Not much research has been conducted along these lines previously. Some exceptions are a thesis by Caseiro [3]; papers by Lee, Jones & Ben-Amram, and Jones & Bohr [13, 7] which analyse the relationship between program syntax and program termination; and a thesis by Frederiksen [4] that contains syntactical flow analyses sufficient to recognise that a functional program to runs in polynomial time. The work of Kristiansen & Niggl [9, 10] and some work of Niggl (see [15] for an overview) are also to a certain extent exceptions.

Our programming language is not well equipped syntactically, but is natural in the sense that it is an essential fragment of many popular real-life programming languages, e.g., C. By doing derivations in our calculus, we are able to establish useful information about the computational complexity of programs, and as our theorems show, the calculus is powerful. Thus, it might be tempting to discuss real-life applications. Be that as it may, such applications are hardly required to justify this paper. We believe this paper conveys a theoretical insight that has a value in its own right.

2 Programs, Commands and Expressions

We consider nondeterministic imperative programs that manipulate natural numbers held in a fixed number of program variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$. Programs may be iterative but not recursive. We will call such a program a *command* in the variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$.

2.1 Syntax

The *core expression* and the *core commands* have forms given by the grammar

$$
\begin{array}{rcl}
\mathtt{X} \in \text{Variable} & ::= & \mathtt{X}_1 \mid \mathtt{X}_2 \mid \mathtt{X}_3 \mid \ldots \\
\mathtt{e} \in \text{Expression} & ::= & \mathtt{X} \mid \mathtt{(e\ +\ e)} \mid \mathtt{(e\ *\ e)} \\
\mathtt{C} \in \text{Command} & ::= & \mathtt{skip} \mid \mathtt{X{:=}e} \mid \mathtt{C}_1\mathtt{;C}_2 \mid \mathtt{loop\ X\ \{C\}} \\
& \mid & \mathtt{if\ ?\ then\ C\ else\ C} \mid \mathtt{while\ ?\ do\ \{C\}}
\end{array}
$$

The variable $\mathtt{X}_\ell$ is not allowed to occur in the body $\mathtt{C}$ of the loop $\mathtt{loop\ X}_\ell\ \mathtt{\{C\}}$, and we will usually omit parentheses in the expressions.

2.2 Semantics

A core command is executed as expected from its syntax, so we omit a detailed formalisation. Each variable holds a natural number. The loop command `loop X {C}` executes the command `C` in its body n times in a row, where n is the value stored in `X` when the loop starts. The content of `X` will not be modified during the execution (`X` is not allowed to occur in `C`). The command `if ? then` $\mathtt{C}_1$ `else` $\mathtt{C}_2$ makes a nondeterministic choice and executes one, and only one, of the commands $\mathtt{C}_1$ and $\mathtt{C}_2$. The command $\mathtt{C}_1\mathtt{;}\mathtt{C}_2$ executes first the command $\mathtt{C}_1$ and then the command $\mathtt{C}_2$. Commands of the form `X:=e` are ordinary assignment statements, and the command `skip` does nothing. Finally, the command `while ? do {C}` makes a nondeterministic choice and executes either the command `skip` or the command `C; while ? do {C}`.

Although generally familiar, the language above fails Turing completeness owing to the lack of constants, and uninterpreted tests in `if` and `while`. On the other hand, the forthcoming inference rules will be seen both sound and complete. A straightforward extension yields *sound* but *incomplete* inference rules for a Turing complete language.

Let `C` be a command in the variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$. The relation

$$[\![\mathtt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n)$$

holds iff there *exists an execution of* `C` such that variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$ respectively hold the numbers $x_1, \ldots, x_n$ when the execution starts, and variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$ respectively hold the numbers $x'_1, \ldots, x'_n$ when the execution terminates. It is defined in the expected way.

2.3 *mwp*-Bounds on Variable Growth

Given command `C` and variables $\mathtt{X}_i, \mathtt{X}_j$, our goal is to discover data-flow relations between the *initial value* x_i of $\mathtt{X}_i$ and the *final value* x'_j of $\mathtt{X}_j$ that hold whenever $[\![\mathtt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n)$. An *mwp*-bound is a number-theoretic expression of the form

$$\max(\vec{x}, \vec{y}, \vec{z}, q(\vec{y}, \vec{z})) + p(\vec{z})$$

where $\vec{x}$, $\vec{y}$, and $\vec{z}$ are disjoint lists of variables, and q and p are honest polynomials[1], and any of the lists $\vec{x}$, $\vec{y}$ and $\vec{z}$ might be empty.

In *mwp*, *m* stands for "maximum", *p* stands for "polynomial", and *w* stands for "weak polynomial". We call $\vec{x}$ the *m-variables* of the *mwp*-bound; $\vec{y}$ the *w-variables* of the *mwp*-bound; $\vec{z}$ the *p-variables* of the *mwp*-bound.

We will use $W, V, U, \ldots$ to denote *mwp*-bounds. When convenient we will display the variables in an *mwp*-bound W by the notation $W(\vec{x}; \vec{y}; \vec{z})$ where $\vec{x}$, $\vec{y}$ and $\vec{z}$ are respectively the *m*-variables, *w*-variables and *p*-variables of W.

[1] A polynomial p is *honest* if it is monotone in all its variables, e.g., if y in p implies $p(y, \vec{x}) \leq p(y + 1, \vec{x})$.

Given a command $\mathtt{C}$ in $\mathtt{X}_1, \ldots, \mathtt{X}_n$, a *set of bounds* has the form

$$[\![\mathtt{C}]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \;\Rightarrow\; x'_1 \leq W_1 \wedge \ldots \wedge x'_n \leq W_n \qquad (*)$$

where $W_1, \ldots, W_n$ are *mwp*-bounds.

Truth of (*) implies that any values computed by the command $\mathtt{C}$ will be polynomially bounded in the inputs. The value computed into $\mathtt{X}_1$ by command $\mathtt{loop}\,\mathtt{X}_2\,\{\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_1\}$ grows exponentially, and thus the command has no *mwp*-bounds.

Example. An example of a set of bounds:

$$[\![\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2\}]\!](x_1, x_2, x_3 \rightsquigarrow x'_1, x'_2, x'_3) \Rightarrow x'_1 \leq W_1 \wedge x'_2 \leq W_2 \wedge x'_3 \leq W_3$$

where $W_1(x_1;;x_2,x_3) = x_1 + x_2 \cdot x_3$ and $W_2(x_2;;) = x_2$ and $W_3(x_3;;) = x_3$. □

3 A Matrix Algebra of *mwp*-Bounds

We introduce the *mwp*-algebra and some basic matrix-theoretic notation.

3.1 The Form of *mwp*-Matrices

For mathematical convenience, we will group all the relations between pairs of variables into a single data structure, an $n \times n$ matrix M that records the kinds of dependencies, but not the exact polynomials. We write $i \xrightarrow{\alpha}_M j$ in case $M_{ij} = \alpha$.

If the relation $i \xrightarrow{\alpha}_M j$ holds, then $\alpha \in \{\bot, m, w, p\}$ yields information about the data flow from the *source variable* $\mathtt{X}_i$ to the *target variable* $\mathtt{X}_j$ that occurs while executing a command $\mathtt{C}$.

- If $\alpha = \bot$, the source variable's value is not used to compute the target variable's new value
- If $\alpha = m$, the source variable's value *will not be increased*, and *will not be added* to the target variable's original value. This happens, e.g., in the program $\mathtt{X}_j\mathtt{:=}\mathtt{X}_i$.
- If $\alpha = w$, the source variable's value *might be increased*, but *will not be added* to the target variable's original value. This happens, e.g., in the program $\mathtt{X}_j\mathtt{:=}\mathtt{X}_i\mathtt{*}\mathtt{X}_i$.
- If $\alpha = p$, the source variable's value *might be increased* and *might also be added* to the target variable's original value. This happens, e.g, in the program $\mathtt{X}_j\mathtt{:=}\mathtt{X}_j\mathtt{+(}\mathtt{X}_i\mathtt{*}\mathtt{X}_i\mathtt{)}$.

Our calculus will deduce statements $\mathtt{C}\mathtt{:}M$ of a more concise (and less specific) form that imply that *mwp*-bounds exist. Let $\mathtt{C}$ be a command in $\mathtt{X}_1, \ldots, \mathtt{X}_n$ and let M be a $n \times n$ matrix over the set $\{\bot, m, w, p\}$.

By definition *statement* $\mathtt{C}\mathtt{:}M$ *is true*, written $\models \mathtt{C}\mathtt{:}M$, iff there exist *mwp*-bounds $W_1, \ldots, W_n$ such that (*) holds, where x_i is an m-variable of W_j iff $M_{ij} = m$; x_i is an w-variable of W_j iff $M_{ij} = w$; and x_i is a p-variable of W_j iff $M_{ij} = p$.

Example. The example above can now be expressed much more concisely in matrix form, but the exact polynomials involved are lost:

$$\texttt{loop}\,\texttt{X}_3\,\{\texttt{X}_1\texttt{:=}\texttt{X}_1\texttt{+}\texttt{X}_2\} : \begin{pmatrix} m & \bot & \bot \\ p & m & \bot \\ p & \bot & m \end{pmatrix}$$

□

3.2 The Matrix Algebra

Let $\mathcal{V} = \{\bot, m, w, p\}$. The elements in $\mathcal{V}$ are ordered as follows: $\bot < m < w < p$. We use Greek letter $\alpha, \beta, \gamma \ldots$ to denote the elements in $\mathcal{V}$. The *least upper bound* of $\alpha, \beta \in \mathcal{V}$ is denoted by $\alpha + \beta$, i.e., $\alpha + \beta = \alpha$ if $\alpha \geq \beta$; otherwise $\alpha + \beta = \beta$. Let $\alpha_1, \ldots, \alpha_n$ be a sequence of values of form $\mathcal{V}$, then $\sum_{i=1\ldots n} \alpha_i \stackrel{\text{def}}{=} \alpha_1 + \cdots + \alpha_n$. The *product* of $\alpha, \beta \in \mathcal{V}$ is denoted by $\alpha \times \beta$ and defined by $\alpha \times \beta = \alpha + \beta$ if $\alpha, \beta \in \{m, w, p\}$; otherwise $\alpha \times \beta = \bot$.

Fix some $n \in \mathbb{N}$. We use $M, A, B, C, \ldots$ to denote $(n \times n)$ matrices over $\mathcal{V}$, and M_{ij} denotes the element in the i'th row and j'th column in the matrix M. We define the *least upper bound* $A \oplus B$ of the matrices A and B by $M = A \oplus B$ iff $M_{ij} = A_{ij} + B_{ij}$. We say that the matrix M is an *upper bound* the matrix A, in notation $M \geq A$, if there exists a matrix B such that $M = A \oplus B$. Thus we have a partial ordering of the universe of matrices. The ordering symbols $\geq, \leq, >, <$ have their standard meaning with respect to this ordering, and we will use standard terminology, that is, we may say that A lies above B when $A \geq B$, that A is a matrix strictly below B when $A < B$, etcetera. We define the *product* $A \otimes B$ of the matrices A and B by $M = A \otimes B$ iff $M_{ij} = \sum_{k=1,\ldots,n} A_{ik} \times B_{kj}$ (standard matrix multiplication). The *zero matrix* is denoted by $\mathbf{0}$. We define $\mathbf{0}$ by $M = \mathbf{0}$ iff $M_{ij} = \bot$ for all indices i, j. The *identity matrix* is denoted by $\mathbf{1}$. We define $\mathbf{1}$ by $M = \mathbf{1}$ iff $M_{ij} = m$ if $i = j$, and $M_{ij} = \bot$ if $i \neq j$. A unary operation on matrices, denoted $*$ and called *closure*, is defined by the infinite sum

$$M^* \;=\; 1 \oplus M \oplus (M \otimes M) \oplus (M \otimes M \otimes M) \oplus (M \otimes M \otimes M \otimes M) \oplus \ldots$$

Let $\mathcal{M}$ denote the set of $(n \times n)$ matrices. The algebraic structure $(\mathcal{M}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ is a finite closed semiring. The closure operator is defined in any closed semiring, and we have $M^* = 1 \oplus (M \otimes M^*)$. For the definition of a closed semiring and more on related algebra, see, e.g., [1].

We use $V, U, T \ldots$ to denote vectors over $\mathcal{V}$, and V_i denotes the i'th element in the vector V. We define the *product* $V \times \alpha$ of the vector V and the value $\alpha \in \mathcal{V}$ by $V' = V \times \alpha$ iff $V'_i = V_i \times \alpha$. We define the *least upper bound* by $T \oplus U$ of the vectors T and U by $V = T \oplus U$ iff $V_i = T_i \oplus U_i$. We define the *zero vector* $\mathbf{0}$ by $V = \mathbf{0}$ iff $V_i = \bot$ for any index i.

Let $V \stackrel{k}{\leftarrow} \alpha$ denote the *modification* of the vector V defined by $V' = V \stackrel{k}{\leftarrow} \alpha$ iff $V'_i = \alpha$ if $i = k$, and $V'_i = V_i$ if $i \neq k$. Let M be a matrix and let V be a vector. Then $M \stackrel{k}{\leftarrow} V$ denotes the matrix obtained by replacing the k'th column in M by the vector V, that is, $M' = M \stackrel{k}{\leftarrow} V$ iff $M'_{ij} = V_i$ if $j = k$, and $M'_{ij} = M_{ij}$ if $j \neq k$.

Example. Assume $n = 4$ and let $V = \begin{pmatrix} m \\ p \\ \bot \\ p \end{pmatrix}$. Then

$$\mathbf{1} \overset{2}{\leftarrow} V \;=\; \begin{pmatrix} m & \bot & \bot & \bot \\ \bot & m & \bot & \bot \\ \bot & \bot & m & \bot \\ \bot & \bot & \bot & m \end{pmatrix} \overset{2}{\leftarrow} \begin{pmatrix} m \\ p \\ \bot \\ p \end{pmatrix} \;=\; \begin{pmatrix} m & m & \bot & \bot \\ \bot & p & \bot & \bot \\ \bot & \bot & m & \bot \\ \bot & p & \bot & m \end{pmatrix}$$

□

Before we proceed to define the proof calculus, let us indicate how the matrices will be used in a book-keeping process recording information on the data flow in program executions. Recall that the matrix M induces the relations $\overset{m}{\rightarrow}_M$, $\overset{w}{\rightarrow}_M$, $\overset{p}{\rightarrow}_M$ in a natural way, that is, $i \overset{\alpha}{\rightarrow}_M j$ iff $M_{ij} = \alpha$. If the statement $\mathtt{C} : M$ is derivable in the calculus, the relations $\overset{m}{\rightarrow}_M$, $\overset{w}{\rightarrow}_M$, $\overset{p}{\rightarrow}_M$ will give information on how data might flow during an execution of the program $\mathtt{C}$. The closure operator plays an important role in the derivation rules for the loop statements. It is easy to see the closure A^* is the least matrix B above $\mathbf{1} \oplus A$ such that $i \overset{\alpha}{\rightarrow}_B k \wedge k \overset{\beta}{\rightarrow}_B j \;\Rightarrow\; i \overset{\alpha+\beta}{\rightarrow}_B j$.

4 The Calculus

We now introduce a proof calculus that allows formal derivations of true statements.

We derive $\vdash \mathtt{e} : V$, for core expression $\mathtt{e}$ and vector V, by the rules

$$(E1)\ \vdash \mathtt{X}_i : \mathbf{0} \overset{i}{\leftarrow} m$$

$$(E2)\ \frac{\vdash \mathtt{e}_1 : V \qquad \vdash \mathtt{e}_2 : U}{\vdash \mathtt{e}_1 \mathtt{*} \mathtt{e}_2 : (V \oplus U) \times w} \qquad (E3)\ \frac{\vdash \mathtt{e}_1 : V \qquad \vdash \mathtt{e}_2 : U}{\vdash \mathtt{e}_1 \mathtt{+} \mathtt{e}_2 : (V \oplus U) \times w}$$

$$(E4)\ \frac{\vdash \mathtt{e}_1 : V \qquad \vdash \mathtt{e}_2 : U}{\vdash \mathtt{e}_1 \mathtt{+} \mathtt{e}_2 : (V \times p) \oplus U} \qquad (E5)\ \frac{\vdash \mathtt{e}_1 : V \qquad \vdash \mathtt{e}_2 : U}{\vdash \mathtt{e}_1 \mathtt{+} \mathtt{e}_2 : V \oplus (U \times p)}$$

Explanation of rules $(E3), (E4), (E5)$: An expression such as $\mathtt{X}_1\mathtt{+}\mathtt{X}_2$ can be assigned more than one correct vector:

$$\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} w \\ w \end{pmatrix} \quad \text{or} \quad \vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} p \\ m \end{pmatrix} \quad \text{or} \quad \vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} m \\ p \end{pmatrix}.$$

Let Γ be a set of statements, and let $\mathtt{C} : M$ be a statement. We can derive $\Gamma \vdash \mathtt{C} : M$ by applying the rules

$$(S)\ \Gamma \vdash \mathtt{skip} : \mathbf{1} \qquad (I)\ \frac{\Gamma_1 \vdash \mathtt{C}_1 : A \qquad \Gamma_2 \vdash \mathtt{C}_2 : B}{\Gamma_1 \cup \Gamma_2 \vdash \mathtt{if\ ?\ then\ C_1\ else\ C_2} : A \oplus B}$$

$$(A)\ \frac{\vdash \mathtt{e} : V}{\Gamma \vdash \mathtt{X}_\ell \mathtt{:=} \mathtt{e} : \mathbf{1} \overset{\ell}{\leftarrow} V} \qquad (C)\ \frac{\Gamma_1 \vdash \mathtt{C}_1 : A \qquad \Gamma_2 \vdash \mathtt{C}_2 : B}{\Gamma_1 \cup \Gamma_2 \vdash \mathtt{C}_1 \mathtt{;} \mathtt{C}_2 : A \otimes B}$$

further, we can apply the rule

$$(L)\ \frac{\Gamma \vdash \mathtt{C}\colon A}{\Gamma \vdash \mathtt{loop}\ \mathtt{X}_\ell\ \{\mathtt{C}\}\colon B}$$

if the following conditions are satisfied: (1) $\xrightarrow{w}_{A^*}$ and $\xrightarrow{p}_{A^*}$ are irreflexive, and (2) B is the least matrix above A^* such that $\exists j[j \xrightarrow{p}_B i] \Rightarrow \ell \xrightarrow{p}_B i$. We can apply the rule

$$(W)\ \frac{\Gamma \vdash \mathtt{C}\colon M}{\Gamma \vdash \mathtt{while}\ ?\ \mathtt{do}\ \{\mathtt{C}\}\colon M^*}$$

if the following conditions are satisfied: (1) $\xrightarrow{w}_{M^*}$ is irreflexive, and (2) $\xrightarrow{p}_{M^*}$ is empty. Finally, we have the *weakening* and *assumption* rules

$$\frac{\Gamma \vdash \mathtt{C}\colon M}{\Gamma \cup \Gamma' \vdash \mathtt{C}\colon M \oplus M'} \qquad \text{and} \qquad \{\mathtt{C}\colon M\} \vdash \mathtt{C}\colon M$$

for any matrix M', set of statements Γ', and statement $\mathtt{C}\colon M$.

Theorem 1 (Soundness). $\Gamma \vdash \mathcal{C}\colon M \Rightarrow \Gamma \models \mathcal{C}\colon M$.

We say that a core command is *feasible* if there exists a matrix M such that $\models \mathtt{C}\colon M$. We say that a core command is *derivable* if there exists a matrix M such that $\vdash \mathtt{C}\colon M$.

Theorem 2 (Completeness). *Any feasible core command is derivable.*

Theorem 2 *does not* assert that we have $\models \mathtt{C}\colon M \Rightarrow \vdash \mathtt{C}\colon M$ for any matrix M. That is not true.

Note the relationship between the forms of the core commands and the calculus. For each syntactic construction in the core language there is a corresponding derivation rule in the calculus. The core language is carefully constructed to obtain Theorem 2. This completeness property does not contradict the "halting problem" as in certain respects the core language is a weak language. A core command can compute very rapidly increasing functions and even enter an infinite loop, but it cannot set the value of a variable to , e.g., 0. Thus the core language does not yield full Turing computability.

Suppose we e.g. extend the language by adding expressions `0`, `1`, and interpreting tests in `if` and `while` commands. With these assumptions, we will obtain full Turing computability. The assumption rule $\{\mathtt{C}\colon M\} \vdash \mathtt{C}\colon M$ permits us to integrate these, and other commands working on natural numbers, in the calculus. If the language yields full Turing computability, the calculus will of course be incomplete, i.e., there exists a set of statements Γ and a statement $\mathtt{C}\colon M$ such that $\Gamma \models \mathtt{C}\colon M$ and $\Gamma \nvdash \mathtt{C}\colon M$.

Theorem 1 is proved by induction on the height of the derivation of $\vdash \mathtt{C}\colon M$. Here is a proof sketch for Theorem 2: Let $\mathtt{C}'$ be a command in $\mathtt{X}_1, \ldots, \mathtt{X}_n$. Assume that the command $\mathtt{C}'$ is not derivable. (We will prove that $\mathtt{C}'$ is not feasible.) Then there exists a subcommand $\mathtt{C}$ of $\mathtt{C}'$ such that $\mathtt{C}$ is derivable, but either $\mathtt{loop}\ \mathtt{X}_\ell\ \{\mathtt{C}\}$ or $\mathtt{while}\ ?\ \mathtt{do}\ \{\mathtt{C}\}$ is not. Prove that there exist fixed $m, k \in \mathbb{N}$ such that for any $x_1, \ldots, x_n \geq 2$ we have

$$[\![\mathtt{C}^m]\!](x_1, \ldots, x_n \rightsquigarrow x'_1, \ldots, x'_n) \ \wedge \ x'_k \geq 2x_k \qquad (\dagger)$$

where $\mathtt{C}^0 \equiv \mathtt{skip}$ and $\mathtt{C}^{\ell+1} \equiv \mathtt{C}; \mathtt{C}^{\ell}$. (Perhaps it will be helpful to phrase (†) more lyrically: There exist $m \in \mathbb{N}$ and a variable $\mathtt{X}_k$ such that when $\mathtt{C}$ is executed m times in a row on inputs ≥ 2, the content of $\mathtt{X}_k$ will be at least doubled.) It follows that there exists an execution of the command $\mathtt{C}'$ such that the value the command computes into the variable $\mathtt{X}_k$ is not bounded by a polynomial in the inputs. Thus, there will be no matrix M such that $\models \mathtt{C}' : M$. Thus, $\mathtt{C}'$ is not a feasible command. In this proof it is essential that a core command cannot decrease the value of a variable when its inputs are ≥ 1 (otherwise it might multiply by zero).

5 Examples

Example. Let $\mathtt{X}_1 := \mathtt{X}_2;\ \mathtt{X}_2 := \mathtt{X}_3;\ \mathtt{X}_3 := \mathtt{X}_1$ be a command in $\mathtt{X}_1, \mathtt{X}_2, \mathtt{X}_3, \mathtt{X}_4$. We have the derivation

$$\dfrac{\dfrac{\dfrac{\vdash \mathtt{X}_2 : \begin{pmatrix} \bot \\ m \\ \bot \\ \bot \end{pmatrix}}{\vdash \mathtt{X}_1 := \mathtt{X}_2 : \begin{pmatrix} \bot & \bot & \bot & \bot \\ m & m & \bot & \bot \\ \bot & \bot & m & \bot \\ \bot & \bot & \bot & m \end{pmatrix}} \quad \dfrac{\vdash \mathtt{X}_3 : \begin{pmatrix} \bot \\ \bot \\ m \\ \bot \end{pmatrix}}{\vdash \mathtt{X}_2 := \mathtt{X}_3 : \begin{pmatrix} m & \bot & \bot & \bot \\ \bot & \bot & \bot & \bot \\ \bot & m & m & \bot \\ \bot & \bot & \bot & m \end{pmatrix}}}{\vdash \mathtt{X}_1 := \mathtt{X}_2;\ \mathtt{X}_2 := \mathtt{X}_3 : \begin{pmatrix} \bot & \bot & \bot & \bot \\ m & \bot & \bot & \bot \\ \bot & m & m & \bot \\ \bot & \bot & \bot & m \end{pmatrix}} \quad \dfrac{\vdash \mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \\ \bot \end{pmatrix}}{\vdash \mathtt{X}_3 := \mathtt{X}_1 : \begin{pmatrix} m & \bot & m & \bot \\ \bot & m & \bot & \bot \\ \bot & \bot & \bot & \bot \\ \bot & \bot & \bot & m \end{pmatrix}}}{\vdash \mathtt{X}_1 := \mathtt{X}_2;\ \mathtt{X}_2 := \mathtt{X}_3;\ \mathtt{X}_3 := \mathtt{X}_1 : \begin{pmatrix} \bot & \bot & \bot & \bot \\ m & \bot & m & \bot \\ \bot & m & \bot & \bot \\ \bot & \bot & \bot & m \end{pmatrix}}$$

Assume $[\![\mathtt{X}_1 := \mathtt{X}_2;\ \mathtt{X}_2 := \mathtt{X}_3;\ \mathtt{X}_3 := \mathtt{X}_1]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4')$. Since the calculus is sound, there exist *mwp*-bounds W_1, W_2, W_3, W_4 such that

$$x_1' \leq W_1(x_2;;) \ \wedge\ x_2' \leq W_2(x_3;;) \ \wedge\ x_3' \leq W_3(x_2;;) \ \wedge\ x_4' \leq W_4(x_4;;)\ .$$

By inspecting the command we can check that this indeed is the case since we have $x_1' \leq x_2$, $x_2' \leq x_3$, $x_3' \leq x_2$, and $x_4' \leq x_4$. □

Example. We continue the derivation from the previous example and apply the L-rule

$$\dfrac{\begin{array}{c}\vdots \\ \vdash \mathtt{X}_1 := \mathtt{X}_2;\ \mathtt{X}_2 := \mathtt{X}_3;\ \mathtt{X}_3 := \mathtt{X}_1 : M\end{array}}{\vdash \mathtt{loop}\ \mathtt{X}_4 \{\mathtt{X}_1 := \mathtt{X}_2;\ \mathtt{X}_2 := \mathtt{X}_3;\ \mathtt{X}_3 := \mathtt{X}_1\} : A}\ L$$

where $M = \begin{pmatrix} \bot & \bot & \bot & \bot \\ m & \bot & m & \bot \\ \bot & m & \bot & \bot \\ \bot & \bot & \bot & m \end{pmatrix}$. What should the matrix A be like? The conditions for applying the L-rule are (1) $\xrightarrow{w}_{M^*}$ and $\xrightarrow{p}_{M^*}$ are irreflexive, and (2) A is the least matrix above M^* such that $\exists j[j \xrightarrow{p}_A i] \ \Rightarrow\ \ell \xrightarrow{p}_A i$. To check if (1) is satisfied we have to compute M^*. The outright boring computation results in $M^* = \begin{pmatrix} m & \bot & \bot & \bot \\ m & m & m & \bot \\ m & m & m & \bot \\ \bot & \bot & \bot & m \end{pmatrix}$, and we see that (1) is satisfied satisfied since the diagonal of M^* does not contain any w or p. In order to find the matrix A satisfying

condition (2) note that $M^* \geq M^*$ and that $\exists j[j \xrightarrow{p}_{M^*} i] \Rightarrow 4 \xrightarrow{p}_{M^*} i$ holds trivially since the value p does not occur in M^*. Thus, (2) is satisfied when $A = M^*$ and

$$\frac{\vdash \mathtt{X_1 := X_2;\ X_2 := X_3;\ X_3 := X_1} : M}{\vdash \mathtt{loop\ X_4\{X_1 := X_2;\ X_2 := X_3;\ X_3 := X_1\}} : M^*}\ L$$

is an admissible inference. Now, assume

$$[\![\mathtt{loop\ X_4\{X_1 := X_2;\ X_2 := X_3;\ X_3 := X_1\}}]\!](x_1, x_2, x_3, x_4 \rightsquigarrow x_1', x_2', x_3', x_4')$$

Since the calculus is sound, there exist *mwp*-bounds W_1, W_2, W_3, W_4 such that

$$x_1' \leq W_1(x_1, x_2, x_3;;) \ \wedge\ x_2' \leq W_2(x_2, x_3;;) \ \wedge\ x_3' \leq W_3(x_2, x_3;;) \ \wedge\ x_4' \leq W_4(x_4;;)$$

The reader is encouraged to analyse the command and verify that we have $x_1' \leq \max(x_1, x_2, x_3)$, $x_2' \leq \max(x_2, x_3)$, $x_3' \leq \max(x_2, x_3)$, and $x_4' \leq x_4$. Note that x_1 has to occur in the bound on x_1' since the loop's body might not be executed at all. □

Example. Let $\mathtt{X_1{+}X_2}$ be an expression in the variables $\mathtt{X_1}, \mathtt{X_2}, \mathtt{X_3}$. We can find several (three) vectors V such that $\vdash \mathtt{X_1{+}X_2} : V$. If we apply (E3), we can derive

$$\frac{\vdash \mathtt{X_1} : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \qquad \vdash \mathtt{X_2} : \begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix}}{\vdash \mathtt{X_1{+}X_2} : (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus \begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix}) \times w} \quad \text{and} \quad (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus \begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix}) \times w = \begin{pmatrix} w \\ w \\ \bot \end{pmatrix} \qquad \text{(I)}$$

if we apply (E4), we can derive

$$\frac{\vdash \mathtt{X_1} : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \qquad \vdash \mathtt{X_2} : \begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix}}{\vdash \mathtt{X_1{+}X_2} : (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \times p) \oplus \begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix}} \quad \text{and} \quad (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \times p) \oplus \begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix} = \begin{pmatrix} p \\ m \\ \bot \end{pmatrix} \qquad \text{(II)}$$

and if we apply (E5), we can derive

$$\frac{\vdash \mathtt{X_1} : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \qquad \vdash \mathtt{X_2} : \begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix}}{\vdash \mathtt{X_1{+}X_2} : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus (\begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix} \times p)} \quad \text{and} \quad \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus (\begin{pmatrix} \bot \\ m \\ \bot \end{pmatrix} \times p) = \begin{pmatrix} m \\ p \\ \bot \end{pmatrix}. \qquad \text{(III)}$$

□

Example. Let $\mathtt{C} \equiv \mathtt{loop\,X_3\,\{X_1 := X_1{+}X_2\}}$. Assume $[\![\mathtt{C}]\!](x_1, x_2, x_3 \rightsquigarrow x_1', x_2', x_3')$. One can easily prove by induction on $x_3 \in \mathbb{N}$ that $x_1' \leq x_1 + x_2 \times x_3$, $x_2' \leq x_2$, and $x_3' \leq x_3$. Hence, we have $\models \mathtt{C} : M$ for some matrix M. Let us try to find a matrix M' such that $\vdash \mathtt{C} : M'$. (Theorem 2 assures that at least one such matrix exists, but note that theorem does not state that $\models \mathtt{C} : M$ implies $\vdash \mathtt{C} : M$.) We

will try to extend each of the three derivations from the previous example. We start with (I).

$$\dfrac{\dfrac{\vdots}{\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} w \\ w \\ \bot \end{pmatrix}}}{\dfrac{\vdash \mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} w & \bot & \bot \\ w & m & \bot \\ \bot & \bot & m \end{pmatrix}}{\vdash \mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2\} : ?}\ L}$$

When trying to apply the L-rule, we find that we cannot. Let $A = \begin{pmatrix} w & \bot & \bot \\ w & m & \bot \\ \bot & \bot & m \end{pmatrix}$. Then $A^* = A$. One of the conditions for applying the rule, states that the relation $\xrightarrow{w}_{A^*}$ should be irreflexive. This is not the case as $1 \xrightarrow{w}_{A^*} 1$. The same thing occurs when we extend derivation (II). We do not succeed as the conditions for applying the L-rule are not fulfilled. Let us see what happens when we extend (III).

$$\dfrac{\dfrac{\vdots}{\vdash \mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} m \\ p \\ \bot \end{pmatrix}}}{\dfrac{\vdash \mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2 : \begin{pmatrix} m & \bot & \bot \\ p & m & \bot \\ \bot & \bot & m \end{pmatrix}}{\vdash \mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2\} : ?}\ L}$$

Now the L-rule is applicable. Let $B = \begin{pmatrix} m & \bot & \bot \\ p & m & \bot \\ \bot & \bot & m \end{pmatrix}$. We have $B^* = B$. There are no w's or p's on the diagonal of B^*, i.e., the relations $\xrightarrow{w}_{B^*}$ and $\xrightarrow{p}_{B^*}$ are irreflexive, and hence condition (1) for applying the L-rule is satisfied. Next we have to find a matrix C such that condition (2) will be satisfied. Let $C = \begin{pmatrix} m & \bot & \bot \\ p & m & \bot \\ p & \bot & m \end{pmatrix}$. Then C is the least matrix above B^* such that $\exists j[j \xrightarrow{p}_C i] \ \Rightarrow\ 3 \xrightarrow{p}_C i$. Thus, we can complete the derivation

$$\dfrac{\dfrac{\vdots}{\vdash \mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2 : B}}{\vdash \mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_1\mathtt{:=}\mathtt{X}_1\mathtt{+}\mathtt{X}_2\} : C}$$

Note that we got the matrix C "by adding a p" to B^* such that $3 \xrightarrow{p}_C 1$. This reflects that the variable $\mathtt{X}_3$ governing the loop influences the value computed into $\mathtt{X}_1$. In general, when the inference

$$(L)\ \dfrac{\Gamma \vdash \mathtt{C} : A}{\Gamma \vdash \mathtt{loop}\ \mathtt{X}_\ell\ \{\mathtt{C}\} : B}$$

is admissible, we get the matrix B by computing the closure A^* and then "add some p's". Why we have to compute the closure is obvious, the matrix B has to yield an mwp-bound for the command $\underbrace{\mathtt{C};\mathtt{C};\ldots\mathtt{C}}_{k}$ for any $k \in \mathbb{N}$. The reason we (possibly) have to "add some p's" is that the loop variable $\mathtt{X}_\ell$ will have an impact on the bounds of (some of) the values computed in $\mathtt{C}$. □

Example. Let $\mathtt{X}_1{+}\mathtt{X}_1$ be an expression in the variables $\mathtt{X}_1, \mathtt{X}_2, \mathtt{X}_3$. We can find three possible derivations of a statement of the form $\mathtt{X}_1{+}\mathtt{X}_2 : V$. Two of them have the same bottom line.

$$\frac{\vdash \mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \qquad \vdash \mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix}}{\vdash \mathtt{X}_1{+}\mathtt{X}_1 : (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix}) \times w} \quad \text{and} \quad (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix}) \times w = \begin{pmatrix} w \\ \bot \\ \bot \end{pmatrix} \qquad \text{(I)}$$

$$\frac{\vdash \mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \qquad \vdash \mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix}}{\vdash \mathtt{X}_1{+}\mathtt{X}_1 : (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \times p) \oplus \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix}} \quad \text{and} \quad (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \times p) \oplus \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} = \begin{pmatrix} p \\ \bot \\ \bot \end{pmatrix} \qquad \text{(II)}$$

$$\frac{\vdash \mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \qquad \vdash \mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix}}{\vdash \mathtt{X}_1{+}\mathtt{X}_1 : \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \times p)} \quad \text{and} \quad \begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \oplus (\begin{pmatrix} m \\ \bot \\ \bot \end{pmatrix} \times p) = \begin{pmatrix} p \\ \bot \\ \bot \end{pmatrix} \qquad \text{(III)}$$

In the derivations we apply respectively the rules (E3), (E4), and (E5). □

Example. The command $\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2 := \mathtt{X}_1{+}\mathtt{X}_1\}$ is derivable. We give two derivations. They extend respectively derivation (I) and (II) of the previous example.

$$\frac{\begin{array}{c}\vdots \\ \vdash \mathtt{X}_1{+}\mathtt{X}_1 : \begin{pmatrix} w \\ \bot \\ \bot \end{pmatrix}\end{array}}{\vdash \mathtt{X}_2 := \mathtt{X}_1{+}\mathtt{X}_1 : \begin{pmatrix} m & w & \bot \\ \bot & \bot & \bot \\ \bot & \bot & m \end{pmatrix}}\,L}{\vdash \mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2 := \mathtt{X}_1{+}\mathtt{X}_1\} : \begin{pmatrix} m & w & \bot \\ \bot & m & \bot \\ \bot & \bot & m \end{pmatrix}}\,L \qquad \frac{\frac{\begin{array}{c}\vdots \\ \vdash \mathtt{X}_1{+}\mathtt{X}_1 : \begin{pmatrix} p \\ \bot \\ \bot \end{pmatrix}\end{array}}{\vdash \mathtt{X}_2 := \mathtt{X}_1{+}\mathtt{X}_1 : \begin{pmatrix} m & p & \bot \\ \bot & \bot & \bot \\ \bot & \bot & m \end{pmatrix}}\,L}{\vdash \mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2 := \mathtt{X}_1{+}\mathtt{X}_1\} : \begin{pmatrix} m & p & \bot \\ \bot & m & \bot \\ \bot & p & m \end{pmatrix}}\,L$$

In the derivation to the right we are forced to add a p in the matrix when the L-rule is applied, whereas in the left one we are not. (See condition (2) for applying the L-rule.) Assume

$$[\![\mathtt{loop}\,\mathtt{X}_3\,\{\mathtt{X}_2 := \mathtt{X}_1{+}\mathtt{X}_1\}]\!](x_1, x_2, x_3 \rightsquigarrow x_1', x_2', x_3') \ .$$

Let us study the form of the mwp-bounds the derivations yield for x_2'. The left one yields a bound $x_2' \leq W(x_2; x_1;)$, whereas the right one yields a bound $x_2' \leq W'(x_2;; x_1, x_3)$. Thus, the left derivation is the preferred one in the sense that the derivation actually record that we can find a polynomial bound on the value computed into the variable $\mathtt{X}_2$ which does not depend on the content of $\mathtt{X}_3$. (If $\mathtt{X}_3$ stores 0, the assignment $\mathtt{X}_2 := \mathtt{X}_1{+}\mathtt{X}_1$ will not be executed; otherwise it will be executed. So the value computed into $\mathtt{X}_2$ does depend on $\mathtt{X}_3$, but there exists a polynomial bound on the value which does not.) □

Example. The command $\mathtt{loop}\,\mathtt{X}_2\,\{\mathtt{X}_1 := \mathtt{X}_1{+}\mathtt{X}_1\}$ is not derivable. The value computed into $\mathtt{X}_1$ is doubled each time the loop's body is executed. Thus, the value cannot be bounded by a polynomial in the inputs, and then by Theorem 1, the command is not derivable. Let us see what happens when we search for a derivation of the command. There are two vectors V such that $\vdash \mathtt{X}_1{+}\mathtt{X}_1 : V$. We have

$\vdash \mathtt{X_1{+}X_1} : \binom{w}{\perp}$ and $\vdash \mathtt{X_1{+}X_1} : \binom{p}{\perp}$. We try to search for a derivation from each of them.

$$\dfrac{\dfrac{\vdash \mathtt{X_1{+}X_1} : \binom{w}{\perp}}{\vdash \mathtt{X_1 := X_1{+}X_1} : \binom{w\ \perp}{\perp\ m}}}{\vdash \mathtt{loop\,X_2\,\{X_1 := X_1{+}X_1\}} : ?}\ L \qquad \dfrac{\dfrac{\vdash \mathtt{X_1{+}X_1} : \binom{p}{\perp}}{\vdash \mathtt{X_1 := X_1{+}X_1} : \binom{p\ \perp}{\perp\ m}}}{\vdash \mathtt{loop\,X_2\,\{X_1 := X_1{+}X_1\}} : ?}\ L$$

In both cases we find that condition (1) for applying the L-rule is violated, and if we study the other rules of the calculus, we easily see that there will be no way to derive the command. □

References

1. A.V. Aho, J.E. Hopcroft, and D. Jeffrey. *The design and analysis of computer algorithms.* Addison-Wesley Publishing Co., Reading, Mass.-London-Amsterdam, 1975.
2. S.J. Bellantoni and S. Cook. *A New Recursion-Theoretic Characterization of the Polytime Functions.* Computational Complexity, 2:97–110 (1992).
3. V.H. Caseiro *Equations for Defining Poly-time Functions.* Ph.D. thesis, Dept. of informatics, Faculty of Mathematics and Natural Sciences University of Oslo, February 1997
4. C.C. Frederiksen. *Automatic runtime analysis for first order functional programs.* Master Thesis, Dep. of Computer Science, University of Copenhagen, 2002.
5. N.D. Jones. *The expressive power of higher-order types or, life without CONS.* J. Functional Programming 11 (2001), 55-94.
6. N.D. Jones. *LOGSPACE and PTIME characterized by programming languages.* Theoretical Computer Science 228 (1999), 151-174.
7. N.D. Jones and N. Bohr. *Termination analysis of the untyped lambda-calculus.* Rewriting Techniques and Applications, Springer LNCS Volume 3091 (2004), 1-23.
8. L. Kristiansen. *Neat function algebraic characterizations of LOGSPACE and LINSPACE.* Computational Complexity 14 (2005), 72–88.
9. L. Kristiansen and K.-H. Niggl. *On the computational complexity of imperative programming languages.* Theoretical Computer Science 318 (2004), 139-161.
10. L. Kristiansen and K.-H. Niggl. *The Garland Measure and Computational Complexity of Stack Programs.* Electronic Notes in Theoretical Computer Science, Volume 90, Elsevier 2003.
11. L. Kristiansen and P.J. Voda. *Complexity classes and fragments of C.* Information Processing Letters 88 (2003), 213-218.
12. L. Kristiansen and P.J. Voda. *Programming languages capturing complexity classes.* Submitted.
13. C.S. Lee, N. Jones, and A.M. Ben-Amram. *The size-change principle for program termination.* ACM Principles of Programming Languages ACM Press 2001, 81-92.
14. D. Leivant. *Intrinsic theories and computational complexity.* In "Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94", Leivant (ed.), 177-194, Springer, 1995.
15. K.-H. Niggl. *Control Structures in Programs and Computational Complexity.* Annals of Pure and Applied Logic 133 (2005), 247–273 MAY.
16. H. Simmons. *The realm of primitive recursion.* Arch. Math. Logic 27, 177-188 (1988).

On a Question of Sacks — A Partial Solution on the Positive Side

Andrew E.M. Lewis

University of Leeds, Leeds LS2 9JT, England

Abstract. Let us say that a c.e. operator E is degree invariant on any given Turing degree a if $X, Y \in a \to E(X) \equiv_T E(Y)$. In [4] we construct a c.e. operator E such that $\forall X[X <_T E(X) <_T X']$. While we are unable to produce degree invariance everywhere, we are able to ensure that for every degree a there exists b such that $a \vee 0' = b \vee 0'$ and E is degree invariant on b. What appears here is an abbreviated version of the material from that paper, stopping short of most technical details.

1 Introduction

The following question, first asked by Sacks [7] in 1963, has been a matter of great interest to computability theorists over the last forty years.

Definition 1. *A c.e. operator is a c.e. subset of* $2^{<\omega} \times \omega$. *Let* E *be a c.e. operator and* $X \subseteq \omega$. *Identifying sets with their characteristic functions we denote by* $E(X)$ *the set* $\{x : (\exists \sigma \subset X)[(\sigma, x) \in E]\}$. *We call* E *degree invariant if it is the case* $\forall X, Y(X \equiv_T Y \to E(X) \equiv_T E(Y))$.

Question 2. *(Sacks) Does there exist a c.e. operator* E *which is degree invariant and such that for all* $X \subseteq \omega$: $X <_T E(X) <_T X'$?

The work that has been done to date has been largely concentrated on an attempt to show that such a degree invariant c.e. operator cannot exist, or at least to limit the kinds of solutions that might exist. The reason for this is likely two-fold. Most believed that Sacks' question would eventually be answered in the negative and the unapproachability of the positive side also provided a formidable obstacle. Lachlan [3] has shown that if we are to construct a positive solution then we cannot require the degree invariance to be uniform i.e. we cannot require that there is a function h which takes (pairs of) indices of reductions between any sets A and B to (pairs of) indices of reductions between $E(A)$ and $E(B)$. Downey and Shore have shown that if E is a degree invariant c.e. operator which is at least the identity on a cone, then either $E(A)'' \equiv_T A''$ on a cone or $E(A)'' \equiv_T A'''$ on a cone. Perhaps the most useful result on the negative side is that of Slaman and Steel [8], that AD implies $\equiv_T$ admits no finite resolution. This result implies serious restrictions as regards those approaches that could possibly lead to a positive solution. Despite extensive study over a number of years no proof that a positive solution to Sacks' question cannot exist has been found.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 275–286, 2005.

Some results have been obtained on the positive side without being written up. Slaman and Steel have constructed a c.e. operator which acts like a positive solution to Sacks' question for the degrees c.e. in $0'$. Downey and Shore have constructed a positive solution for the tt-degrees. In [4] we present the following partial solution on the positive side: we construct a c.e. operator E which, given any $X \subseteq \omega$, produces a set $E(X)$ such that $X <_T E(X) <_T X'$ and such that for every degree a there exists b such that $a \vee 0' = b \vee 0'$ and E is degree invariant on b. What appears here is an abbreviated version of the material from that paper, stopping short of most technical details.

2 The Atomic Strategies

In his paper [9] 'Post's problem and degree invariant functions', written in 1996, Xiaoding Yi attempted to construct a degree invariant solution to Post's problem. Let $\{(\Psi_i, \Phi_i, \Upsilon_i, \Lambda_i)\}_{i\in\omega}$ be an effective listing of all the (ordered) quadruples of Turing functionals. Yi suggested that we should construct two c.e. operators E and D such that for every $X \subseteq \omega$, $X \leq_T E(X)$ and such that for every $i \in \omega$ the following requirements are satisfied:

$R_i : (\forall X, Y \subseteq \omega)[(X = \Phi_i^Y) \wedge (Y = \Psi_i^X) \rightarrow E(Y) \leq_T E(X)]$,
$P_i : (\forall X \subseteq \omega)[E(X) \neq \Upsilon_i^X]$,
$N_i : (\forall X \subseteq \omega)[D(X) \neq \Lambda_i^{E(X)}]$.

In order to ensure that $X \leq_T E(X)$ for all $X \subseteq \omega$ we can simply insist that $(\forall X \subseteq \omega)(\forall n \in \omega)[E(X)(2n) = X(n)]$. The atomic strategies that we might use in order to satisfy individual requirements are also quite simple. In order to satisfy P_i requirements we might suppose that to each $\sigma \in 2^{<\omega}$ there may be associated a witness, p_i^σ say. If we find that $\Upsilon_i^\sigma(p_i^\sigma) \downarrow = 0$ then we enumerate the axiom $(\sigma, p_i^\sigma) \in E$. The requirement P_i will then be satisfied for all sets X extending σ. For any $\sigma \in 2^{<\omega}$, $n \in \omega$ and at any point in the construction let us say, hopefully without causing undue confusion, that $E(\sigma)(n) = 1$ iff we have already enumerated an axiom $(\sigma', n) \in E$ for some $\sigma' \subseteq \sigma$ and that $E(\sigma)(n) = 0$ otherwise.

In order to satisfy an N_i requirement we might similarly suppose that to each $\sigma \in 2^{<\omega}$ there may be associated a witness, n_i^σ say. If we find at any stage that $\Lambda_i^\tau(n_i^\sigma) \downarrow = 0$ for some $\tau \subset E(\sigma)$ and that $\tau \subset E(\sigma')$ for all strings σ' extending σ then we can enumerate the axiom $(\sigma, n_i^\sigma) \in D$ and restrain the enumeration of axioms of the form $(\sigma', n) \in E$ such that σ' is compatible with σ and $\tau(n) \downarrow = 0$.

(†) In what follows it will be convenient to adopt the convention that for any $\sigma \in 2^{<\omega}$ and $i, n \in \omega$, $\Psi_i^\sigma(n) \downarrow$ only if the computation converges in less than $|\sigma|$ steps (so that $n <| \sigma | - 1$) and $\Psi_i^\sigma(n') \downarrow$ for all $n' < n$ (and similarly for Φ_i, Υ_i and Λ_i).

Definition 3. *Given $\sigma, \phi \in 2^\omega$ we shall denote $\sigma|\phi$ if σ and ϕ are incompatible.*

For the sake of requirement R_i certain numbers are reserved for enumeration into $E(X)$, those of the form $2\langle 1, i, j\rangle + 1$ let's say (so that those of the form $2\langle 0, i, j\rangle + 1$ may be used as witnesses for the requirement P_i). Suppose that at some stage of the construction we find that there exist strings ϕ, ϕ' and σ such that $\phi \subseteq \phi'$, $\sigma|\phi$, $\sigma \subseteq \Phi_i^{\phi'}$ and $\Psi_i^{\sigma} = \phi$. Let $|\phi| = n_0$ and let n_1 be the maximum such that $\langle 1, i, n_1\rangle <| \sigma|$. Define $n = min\{n_0, n_1 + 1\}$ and let τ be the initial segment of $E(\phi)$ of length $2n$. For every $n' < n$, if $\tau(2n'+1) = 1$ and it is the case that $E(\sigma)(2\langle 1, i, n'\rangle+1) = 0$ then enumerate the axiom $(\sigma, 2\langle 1, i, n'\rangle+1) \in E$ (we insist that if any axiom $(\sigma', n'') \in E$ is to be enumerated then $n'' < 2|\sigma'|$ – this allows us to maintain a certain 'tidiness'). Restraints to which the requirement R_i is subject mean that, even where the requirement is satisfied, the relevant Turing reductions will not be uniform. Of course, if we use only these techniques in order to satisfy the requirement R_i then the possibility remains that there exist distinct $X, Y \subseteq \omega$ such that $Y = \Psi_i^X$, Φ_i^Y is partial but is not incompatible with X and for infinitely many $\phi \subset Y$ there exists ϕ' extending ϕ and $\sigma \subset X$ such that $\sigma \subseteq \Phi_i^{\phi'}$ and $\phi = \Psi_i^{\sigma}$, so that we act in order to try and ensure $E(Y) \leq_T E(X)$. We shall address this matter in the next section.

In order to satisfy all of the N_i and all of the P_i requirements it is clear that a simple finite injury construction would suffice. When we try to satisfy these together with the R_i requirements, however, several problems are immediately apparent. Let us suppose, momentarily only, that we were to prioritize the requirements thus: $R_0, P_0, N_0, R_1, P_1, N_1, ...$ Then define:

Definition 4. *Given $X, Y \subseteq \omega$ and $n \in \omega$ we denote $L_n(X, Y)$ if there exists a finite sequence of sets $X = X_0, X_1, .., X_m = Y$ such that for each $m' < m$ there exists $i \leq n$, $\Psi_i^{X_{m'}} = X_{m'+1}$ and $\Phi_i^{X_{m'+1}} = X_{m'}$.*

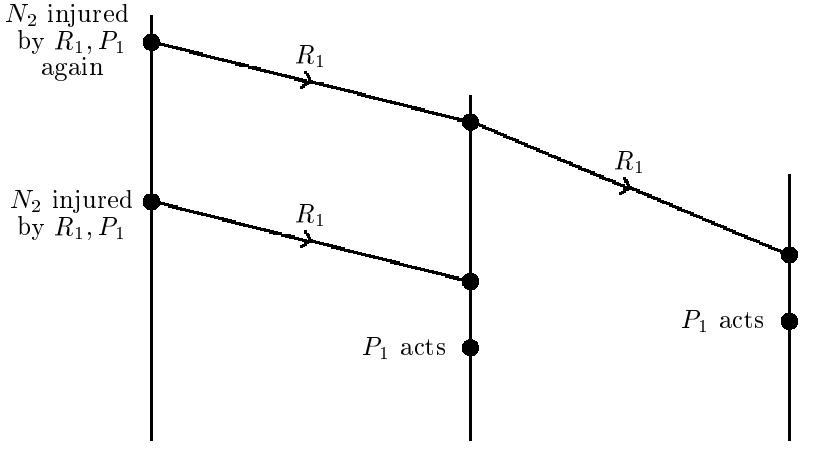


Fig. 1

Since for each $X \subseteq \omega$ and $n \in \omega$ there may exist an infinite number of $Y \subseteq \omega$ such that $L_n(X, Y)$ any N requirement for that set, for example, may be injured

by the actions taken on behalf of P requirements of higher priority for infinitely many other sets, and the action of higher priority R requirements on infinitely many other sets in order to accomodate those enumerations (see fig 1). Similarly P and R requirements may be injured an infinite number of times by higher priority requirements.

In order to get the partial result, then, we do not act to satisfy all R requirements. We begin, infact, by not acting to satisfy any R requirements at all and then proceed to define 'priority cones' (where a cone in this context is all those sets which extend a specified initial segment) on which we act to satisfy a specific R requirement with a certain priority. So for each set X and $i \in \omega$ we may not act to satisfy R_i with respect to X or we may eventually act to satisfy R_i with a certain priority n. In the latter case denote $Pr_i(X) = n$ and otherwise leave $Pr_i(X)$ undefined.

Definition 5. *Given $X, Y \subseteq \omega$ and $n \in \omega$ we denote $L'_n(X, Y)$ if there exists a finite sequence of sets $X = X_0, X_1, .., X_m = Y$ such that for each $m' < m$ there exists i, $\Psi_i^{X_{m'}} = X_{m'+1}$, $\Phi_i^{X_{m'+1}} = X_{m'}$ and $Pr_i(X_{m'}) \downarrow \leq n$.*

We allocate priority cones in such a way that for any $X \subseteq \omega$ and $n \in \omega$ there are a finite number of distinct sets Y such that $L'_n(X, Y)$. For every set Y we are able to show that there is a set X such that $X \oplus \emptyset' \equiv_T Y \oplus \emptyset'$ and for which we act to satisfy all requirements

$$R'_i(X) : (\forall Y' \subseteq \omega)[(X = \Phi_i^{Y'}) \wedge (Y' = \Psi_i^X) \rightarrow E(Y') \equiv_T E(X)].$$

In defining priority cones we are, in effect, constructing a total tree of sets X for which we are able to satisfy all requirements of the form $R'_i(X)$. It is the fact that the priority cones, once initially defined, may need to be adjusted (redefined) a finite number of times which means that, in order to compute this total tree, an oracle for $0'$ is required – so that we are only able to achieve the partial result.

There is another difficulty which immediately presents itself. The following is just one example of this difficulty. Let us suppose that a requirement R_i is allocated priority n on all sets extending a certain string σ. Later we may wish to act in order to satisfy a requirement $N_{i'}$ with $i' \geq n$ on some string σ' extending σ. Suppose that at that stage of the construction we have previously found strings ϕ, ϕ' such that $\phi \subseteq \phi'$, $\sigma' \upharpoonright \phi$, $\sigma' \subseteq \Phi_i^{\phi'}$ and $\Psi_i^{\sigma'} = \phi$. If $|\phi| = m$ then let τ be the initial segment of $E(\phi)$ of length $2m$. For the sake of simplicity, let us suppose that σ' is of length at least $\langle 1, i, m-1 \rangle + 1$. It may be the case that, because of the action taken on behalf of some $P_{i''}$ with $i'' > i'$ we have already enumerated an axiom so that for some ψ extending ϕ, τ is not an initial segment of $E(\psi)$. We may later find σ'', ψ' such that $\sigma' \subset \sigma''$, $\psi \subset \psi'$, $\sigma'' \subseteq \Phi_i^{\psi'}$ and $\Psi_i^{\sigma''} = \psi$. Thus the action taken on behalf of the lower priority requirement $P_{i''}$ on all sets extending ψ, combined with the action that we take on behalf of R_i, injures the requirement $N_{i'}$ on all sets extending σ''. Even if such an enumeration (for $P_{i''}$) has not already taken place before the stage at which we wish to act in order to satisfy the requirement $N_{i'}$, we cannot subsequently restrain (on behalf

of this requirement $N_{i'}$) such an enumeration from taking place unless ψ is of sufficient length such that $\sigma' \subseteq \Phi_i^{\psi}$. We shall discuss how to solve this problem using infinite injury techniques shortly, but let us first describe the priority cone construction.

3 The Priority Cone Construction

In what follows we shall introduce terminologies and notations which are used in [4], but which are more complicated than is necessary in order to present the material which appears in this paper, since most technical details are omitted. We shall make their use since it seems preferable that all notations and terminologies should be in line with [4].

We shall proceed to define priority cones by enumerating 'priority axioms' of the form (σ, i, c, n) for $\sigma \in 2^{<\omega}$, $i, n \in \omega$ and $c \in \{0, 1\}$. We shall enumerate the priority cones in pairs. Each such pair will consist of a 'primary cone' and a 'secondary cone'. If we enumerate a priority axiom of the form $(\sigma, i, 0, n)$ then this defines a primary cone and means that we will act to satisfy requirement R_i on all sets X such that $\sigma \subset X$ with priority n so long as if Ψ_i^X is total then it is in the corresponding secondary cone. Let $r : \omega \to \omega$ be a computable function such that for all $i \in \omega$, $\Phi_{r(i)} = \Psi_i$, $\Psi_{r(i)} = \Phi_i$, $\Upsilon_{r(i)} = \Upsilon_i$ and $\Lambda_{r(i)} = \Lambda_i$ and such that for all $i, j \in \omega$, if $r(i) = j$ then $r(j) = i$. If we enumerate a priority axiom of the form $(\sigma, i, 1, n)$ then this defines a secondary cone and means that we shall act to satisfy requirement $R_{r(i)}$ on all sets X such that $\sigma \subset X$ with priority n so long as if $\Psi_{r(i)}^X$ is total then it is in the corresponding primary cone. We shall also define what will be called 'satisfaction cones'. If we define the satisfaction cone (σ, i, n) for $\sigma \in 2^{<\omega}$ and $i, n \in \omega$ then this will mean that for all sets X such that $\sigma \subset X$, if there exists a set Y such that $Y = \Psi_i^X$ and $X = \Phi_i^Y$ then we shall act in order to guarantee that $E(X) \equiv_T E(Y)$.

We have initially that J is defined to be the emptyset. Into J we shall enumerate triples of the form (σ, m, t) such that $\sigma \in 2^{<\omega}$ and $m, t \in \omega$. If we enumerate the triple (σ, m, t) into J then this will mean that we are looking to define a primary cone 'above' σ i.e. we are looking to enumerate a priority axiom of the form $(\sigma', i, 0, n)$ for some $\sigma' \supseteq \sigma$ and for some $i, n \in \omega$. If $(\sigma, m, t), (\sigma'', m', t') \in J$ and $m < m'$ then we shall look to define a primary cone above σ before we look to do so above σ''. We shall also have initially that S is defined to be the emptyset. It is into the set S that we shall enumerate priority axioms when we wish to define a priority cone. Priority axioms may be removed from this set. It is into the set S' that we shall enumerate axioms defining satisfaction cones. We may also remove axioms from S'.

It will be convenient, infact, to 'name' the priority cones that we enumerate into S. The priority axiom defining any such cone might, of course, be regarded as a name for that cone but would certainly prove cumbersome in what is to follow. We shall name, then, the n^{th} primary cone to be enumerated into S, $cone(n, 0)$. Now an axiom may be enumerated into S, then removed and subsequently enumerated into S again. When such an axiom defining a primary cone

is enumerated back into S we shall regard this axiom as defining a new cone, and it will receive a new name accordingly. The secondary cone associated with the primary cone $cone(n,0)$, we shall name $cone(n,1)$ (so that, as will become apparent form the construction, the axiom defining this cone may change a finite number of times). If $cone(n,c)$ is defined by an axiom of the form (σ,i,c,n') we shall say that the priority of this cone is n'.

Let us return to one of the problems that we discussed earlier. We must deal appropriately with the situation in which for some $i \in \omega$ there exist distinct $X,Y \subseteq \omega$ such that $Y = \Psi_i^X$, Φ_i^Y is partial but is not incompatible with X and for infinitely many $\phi \subset Y$ there exists ϕ' extending ϕ and $\sigma \subset X$ such that $\sigma \subseteq \Phi_i^{\phi'}$ and $\phi = \Psi_i^{\sigma}$. In order to solve this problem, then, we shall computably enumerate sets $\upsilon(a,\sigma)$. All such sets are initially empty. Given $\sigma \in 2^{<\omega}, a \in \omega$ suppose that $a = \langle n,c \rangle$, that at stage s of the construction $cone(n,c)$ is presently in S and is defined by the priority axiom (σ',i,c,n') say, and that we find strings ϕ, and ϕ' extending ϕ, such that all of the following apply.

(i) The strings σ and σ' are compatible.

(ii) Let (σ'',i,c',n') be the priority axiom defining the secondary/primary cone corresponding to the primary/secondary $cone(n,c)$. Then $\sigma'' \subseteq \phi'$.

(iii) If $c = 0$ then define $r'(i) = i$ and if $c = 1$ then define $r'(i) = r(i)$. Then $\sigma \subseteq \Phi_{r'(i)}^{\phi'}$ and $\phi = \Psi_{r'(i)}^{\sigma}$.

(iv) Let ϕ'' be the longest string such that $\phi'' \in \upsilon(a,\sigma)$ if there exists such and otherwise define $\phi'' = \emptyset$ (we let $\emptyset$ denote the empty string). Then $\Phi_{r'(i)}^{\phi''} \subset \Phi_{r'(i)}^{\phi}$.

In this case we should enumerate at stage s all strings $\phi''' \subseteq \phi$ into all sets $\upsilon(a,\sigma''')$ such that $\sigma \subseteq \sigma'''$. In what follows it will often be notationally convenient to freely identify partial functions defined on initial segments, finite sequences and k-tuples for varying $k \in \omega$.

Definition 6. *Given $\sigma,\sigma' \in 2^{<\omega}$ and $\alpha = (a_1,..,a_k) \in \omega^{<\omega}$ for $k \geq 0$ we shall denote at stage s of the construction, $l_{s,\alpha}(\sigma,\sigma')$ if $\sigma' = \emptyset$ or if there exists a finite sequence of strings $\sigma = \sigma_1,...,\sigma_{k+1} \supseteq \sigma'$ such that for all k' if $1 \leq k' \leq k$ then $\sigma_{k'+1} \in \upsilon(a_{k'},\sigma_{k'})$. For any $n \in \omega$ we shall denote at stage s of the construction, $l_{s,n}(\sigma,\sigma')$ if there exists $\alpha = (a_1,..,a_k)$ such that for all k' with $1 \leq k' \leq k$, if $a_{k'} = \langle n',c' \rangle$ then $cone(n',c')$ is a priority cone presently in S of priority $\leq n$, and such that $l_{s,\alpha}(\sigma,\sigma')$. We shall denote $l_{s,n,a}(\sigma,\sigma')$ if it is also the case that $a_1 = a$.*

Definition 7. *At any stage in the construction and for any $\sigma' \in 2^{<\omega}$ we define $m(\sigma')$ to be the number of axioms in S' of any form (σ,i,n) such that $\sigma \subset \sigma'$. Note that for any $\sigma' \in 2^{<\omega}$ the value $m(\sigma')$ may change as the construction progresses.*

Definition 8. *Given pairwise incompatible strings $\sigma_1,..,\sigma_k$ for some $k \in \omega$ we say that $\sigma_{k'}$ for some k' with $1 \leq k' \leq k$ is the 'leftmost' ('rightmost') if*

for each k'' with $1 \leq k'' \leq k$ such that $k'' \neq k'$, there exists a least m such that $\sigma_{k'}(m) \downarrow \neq \sigma_{k''}(m) \downarrow$ and in each case for this least m, $\sigma_{k'}(m) = 0$ ($\sigma_{k'}(m) = 1$).

Definition 9. *Given any $\sigma, \sigma' \in 2^{<\omega}$ we let $\sigma \star \sigma'$ be the finite binary string which is σ concatenated with σ'.*

At stage $s = 0$: enumerate the triples $(0,0,0)$ and $(1,0,0)$ into J.

At stage $s > 0$:
Step 1) For each string σ of length $\leq s$ (taken in lexicographical order) and each $a \in \omega$ such that if $a = \langle n, c \rangle$ then $cone(n,c)$ is a priority cone presently in S, defined by the priority axiom (σ', i, c, n') say, check to see whether there exist ϕ and ϕ' extending ϕ of length $\leq s$ which satisfy (i)–(iv) above. If so then enumerate every string $\phi''' \subseteq \phi$ into every set $v(a, \sigma''')$ such that $\sigma \subseteq \sigma'''$.

Step 2) First we check to see whether there exists an axiom $(\sigma_0, i, c, n) \in S$ and two incompatible strings σ_1 and σ_2 extending σ_0 such that if (σ_0, i, c, n) defines $cone(n', c)$ presently in S and $a = \langle n', c \rangle$ then $l_{s,n,a}(\sigma_1, \sigma_2)$. If not then proceed to step 3). Otherwise, of all such axioms consider those such that n has the least value and of these choose that such that c has the least value (there will only be one such). Choose σ_1, σ_2 as above such that there is the shortest possible $\alpha = (a_1, .., a_k)$ (that with the least value of k) that witnesses that it is the case $l_{s,n,a}(\sigma_1, \sigma_2)$ and such that, this latter condition satisfied, σ_1 takes the shortest possible value and perform the following:

(i) Remove that axiom (σ_0, i, c, n) from S. If $c = 0$ then remove all axioms from S of any form (σ_3, i', c', n'') such that $n'' \geq n$. If $c = 1$ then remove all axioms from S of any form (σ_3, i', c', n'') such that $n'' > n$. For all n'', c', σ_3, make $v(\langle n'', c' \rangle, \sigma_3)$ empty unless $cone(n'', c')$ is presently in S and of priority $< n$. Remove all axioms from S' of any form (σ_3, i', n'') such that $n'' \geq n$. Remove all triples from J of any form (σ_3, m, t) such that $t \geq n$.
(ii) If $c = 0$; then let $\sigma_3 = \Psi_i^{\sigma_1}$. Enumerate the axioms $(\sigma_1, i, 0, n)$ and $(\sigma_3, i, 1, n)$ into S. If this is the k^{th} time that we have enumerated an axiom defining a primary cone into S then the priority cones that these axioms define shall be called $cone(k, 0)$ and $cone(k, 1)$ respectively. Enumerate the axiom (σ_1, i, n) into S' and enumerate the triples $(\sigma_1 \star 0, m(\sigma_1 \star 0), n)$ and $(\sigma_1 \star 1, m(\sigma_1 \star 1), n)$ into J.
(iii) If $c = 1$; then there exists σ_3 of length $\leq s$ lying in the primary cone corresponding to this secondary cone (i.e. the only cone in S defined by a priority axiom of the form $(\sigma_4, i, 0, n)$ for some σ_4) such that $\sigma_1 \subseteq \Psi_i^{\sigma_3}$. Enumerate the axiom $(\sigma_1, i, 1, n)$ into S. Suppose that the name of the primary cone corresponding to this secondary cone is $cone(k, 0)$. Then the name of this cone is $cone(k, 1)$. Enumerate the axiom (σ_3, i, n) into S'. Enumerate the triples $(\sigma_3 \star 0, m(\sigma_3 \star 0), n)$ and $(\sigma_3 \star 1, m(\sigma_3 \star 1), n)$ into J.
d) Proceed to stage $s + 1$.

Definition 10. *Given any triple $(\sigma, m, t) \in J$ we shall say that this triple has an 'extension' in J if there exists a triple $(\sigma', m', t') \in J$ such that $\sigma \subset \sigma'$. At*

any stage of the construction we define the set J' to be all of those triples in J which do not have an extension in J.

Step 3) Let m be the least such that there exists a triple (σ, m, t) in J' (for some $\sigma \in 2^{<\omega}$ and $t \in \omega$). For that value of m let (σ, m, t) be that triple in J' such that σ is leftmost. Let i be the least such that Ψ_i^σ is incompatible with σ and there is no axiom in S of any form $(\sigma', i, 0, n')$ for any $\sigma' \subset \sigma$ and $n' \in \omega$. If no such i exists then remove the triple (σ, m, t) from J, enumerate into J the triple $(\sigma \star 0, m, t)$ and proceed to stage $s+1$. Otherwise enumerate the axioms $(\sigma, i, 0, n)$ and $(\Psi_i^\sigma, i, 1, n)$ into S where n is the least ≥ 1 such that there does not exist any axiom in S of any form (σ', i', c', n). If this is the k^{th} time that we have enumerated an axiom defining a primary cone into S then the priority cones that these axioms define shall be called $cone(k, 0)$ and $cone(k, 1)$ respectively. Enumerate the axiom (σ, i, n) into S'. Enumerate the triples $(\sigma \star 0, m+1, n)$ and $(\sigma \star 1, m+1, n)$ into J. Proceed to stage $s+1$.

Lemma 11. *Given $\sigma \in 2^{<\omega}$ let us denote $m^\star(\sigma) = n$ if there exists a stage s such that after stage s it is always the case $m(\sigma) = n$. Then for each $n \in \omega$ there exist precisely 2^{n+1} axioms of any form (σ, i, n') such that $m^\star(\sigma) \downarrow = n$ (and for some $i, n' \in \omega$) which are enumerated into S' and which are never subsequently removed. For each such axiom there exist precisely two axioms of the form (σ', i', n'') such that $m^\star(\sigma') \downarrow = n+1$ and $\sigma \subset \sigma'$, which are enumerated into S' and which are never subsequently removed (and the two such strings σ' are incompatible).*

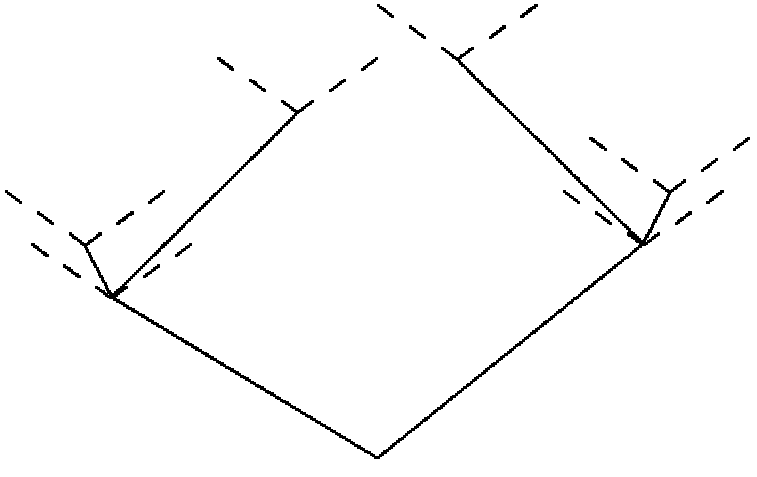


Fig. 2. The satisfaction cones

Lemma 12. *Fix $X \subseteq \omega$ and $n \in \omega$. Suppose that $A \subseteq 2^{<\omega}$ satisfies the following properties; 1) the strings in A are pairwise incompatible and 2) for each $\sigma' \in A$ there exists some $s \in \omega$ and $\sigma \subset X$ such that $l_{s,n}(\sigma, \sigma')$. Then A is finite.*

So for each $Y \subseteq \omega$ consider the set X which is defined as follows. There are two axioms that are enumerated into S' of any form (σ, i, n) such that $m^\star(\sigma) \downarrow = 0$ and which are never subsequently removed. If $Y(0) = 0$ then choose that such that σ is the leftmost of the two and if $Y(0) = 1$ then choose that such that σ is the rightmost. Then $\sigma \subset X$. There are two axioms that are enumerated into S' of any form (σ', i', n') such that $m^\star(\sigma') \downarrow = 1$, $\sigma \subseteq \sigma'$, and which are never subsequently removed. If $Y(1) = 0$ then choose that such that σ' is the leftmost of the two and if $Y(1) = 1$ then choose that such that σ' is the rightmost. Then $\sigma' \subset X$, and so on. Then $X \oplus 0' \equiv_T Y \oplus 0'$ and we shall act to satisfy all requirements $R_i'(X)$.

4 Further Considerations

In satisfying the R requirements it will be simplest, in fact, to have a different set of numbers reserved for enumeration into $E(X)$ associated with each priority cone that we enumerate into S (just because then there is a fixed priority associated with each such). For $cone(n, c)$ it will be the numbers $2\langle 1, \langle n, c\rangle, j\rangle + 1$ that are available for enumeration into $E(X)$.

Let us return to the problem that we addressed briefly before describing the priority cone construction. We wish to avoid a situation in which an N requirement with respect to a set X may be 'injured' an infinite number of times by the actions of lower priority P requirements. We propose to solve this problem using infinite injury techniques. So suppose that at some stage of the construction we wish to act in order to satisfy a requirement N_i on some string σ. Then there is a witness that we wish to enumerate into D, $n_i^\sigma = n$ say. For the reasons described previously we should not immediately enumerate the axiom $(\sigma, n) \in D$ – enumerate instead an 'attention notification' δ, of the form (σ, i, n) maybe. We might then proceed as follows. Let τ be the shortest initial segment of $E(\sigma)$ such that $\Lambda_i^\tau(n) \downarrow = 0$. At any stage s of the construction we shall act to satisfy P and N requirements on strings of length s and for any axioms that we enumerate $(\sigma', n') \in E$ it will be the case $|\sigma'| = s$ (generally speaking we shall 'act on' strings of length s at stage s). Thus we shall have that $\tau \subset E(\sigma')$ for all strings σ' extending σ. It is then a finite task to enumerate all of the 'N-demands' and 'N-requests' in the set, lets call it $T(\delta)$, which we shall define for now (although this definition is subsequently revised in [4]) to be the smallest set satisfying the following conditions. Given any $k \geq 0$ and any k-tuple, $\alpha = (a_1, .., a_k)$:

(i) Define τ_0 as follows. For all $m \in \omega$, if $\tau(2m + 1)$ is defined then $\tau_0(m) \downarrow = \tau(2m + 1)$, and otherwise $\tau_0(m)$ is undefined.

(ii) For $\ell = 1$ to k, given $\tau_{\ell-1}$ (a finite binary string) find all of those $m <| \tau_{\ell-1}|$ which are of the form $\langle 1, a_\ell, j\rangle$ for some $j \in \omega$. If a_ℓ is of the form $\langle n', c\rangle$ and $cone(n', c)$ is presently in S and of priority $\leq i$ then for each such m define $\tau_\ell(j) = \tau_{\ell-1}(m)$. Otherwise define $\tau_\ell = \emptyset$.

(iii) Given τ_k. Find all of those $m <| \tau_k|$ which are of the form $\langle 1, a, j\rangle$ for some $a, j \in \omega$. If a is not of the form $\langle n', c\rangle$ such that $cone(n', c)$ is presently in S and of priority $\leq i$ then the 'N-demand' $(R, \alpha, 2m+1, \tau_k(m))$ is in $T(\delta)$. Then find all of those $m <| \tau_k|$ which are of the form $\langle 0, i', j\rangle$ for some $i', j \in \omega$. For each such m, if $i' > i$ then the 'N-demand' $(P, \alpha, 2m+1, \tau_k(m))$ is in $T(\delta)$ and if $i' \leq i$ then the 'N-request' $(P, \alpha, 2m+1, \tau_k(m))$ is in $T(\delta)$.

Definition 13. *Given $\alpha = (a_1, .., a_k) \in \omega^{<\omega}$ let $\alpha' = (a'_1, .., a'_k)$ be defined as follows. For all ℓ such that $1 \leq \ell \leq k$, if $a_\ell = \langle n, c\rangle$ then $a'_{k+1-\ell} = \langle n, |1-c|\rangle$. We shall call α' the 'reverse cone map' of α.*

These N-demands may be regarded as a form of 'restraint' (to appeal to the standard intuition), as we shall see. Now for any string σ' we may subsequently find that at some stage s' and for some $\alpha' \in \omega^{<\omega}$ we have $l_{s',\alpha'}(\sigma', \sigma)$. Suppose that σ' is of length s' so that $E(\sigma') = E(\sigma'')$ for all $\sigma'' \supseteq \sigma'$. Let α be the reverse cone map of α'. Let's say that an N-demand or an N-request (H, α, m, t) for $H \in \{P, R\}$, $m, t \in \omega$ in $T(\delta)$ is 'consistent' with σ' if $E(\sigma')(m) = t$. If all of the N-demands and N-requests in $T(\delta)$ of the form (H, α, m, t) for $H \in \{P, R\}$, $m, t \in \omega$ are consistent with σ' we might then proceed as follows. First consider those N-demands of this form for which $H = P$. Of these consider those for which $t = 0$ and find that which has the largest value of m. For this value of m enumerate the 'P-restraint' (P, δ, σ', m). Next consider those N-demands for which $H = R$. Of these consider those for which $t = 0$ and find that which has the largest value of m. For this value of m enumerate the 'R-restraint' (R, δ, σ', m). In order to record this action we should then enumerate a 'satisfaction notification' $(\delta, \sigma', \alpha)$.

If we subsequently find some extension of σ, σ'' say, such that for $s'' =| \sigma''|$ we have $l_{s'',\alpha}(\sigma'', \sigma')$ then we may now be in a position to be able to enumerate the axiom $(\sigma'', n) \in D$. There may, however, be other values of α which we must consider. Suppose, in fact, that for every α for which we have enumerated some N-demand or N-request (H, α, m, t) into $T(\delta)$, there is a string σ' for which we have enumerated a satisfaction notification $(\delta, \sigma', \alpha)$, and such that at stage s'' we have $l_{s'',\alpha}(\sigma'', \sigma')$. Then we may enumerate the axiom $(\sigma'', n) \in D$.

Suppose that at some stage of the construction we enumerate the P-restraint or the R-restraint (H, δ, σ, m). At any subsequent stage of the construction we may enumerate a restraint cancellation (H, δ, σ', m) for $\sigma' \supseteq \sigma$. The restraint (H, δ, σ, m) is then considered to be void with respect to all strings extending σ'. Whenever a priority cone of priority $\leq i$ is enumerated into or removed from S all witnesses for the requirement N_i will become undefined and all associated attention notifications, satisfaction notifications and restraints made void.

Definition 14. *Given any $\sigma \in 2^{<\omega}$ and $i \in \omega$ consider (at any stage of the construction) all of the P-restraints that we have enumerated of any form (P, δ, σ', m), for any δ which is an attention notification enumerated in order to satisfy a requirement $N_{i'}$ such that $i' < i$ and for $\sigma' \subseteq \sigma$, and which are not void with respect to σ. Let m be the greatest such that there exists such a P-restraint (P, δ, σ', m). Then $P(\sigma, i) = m$. We define $R(\sigma, i)$ similarly.*

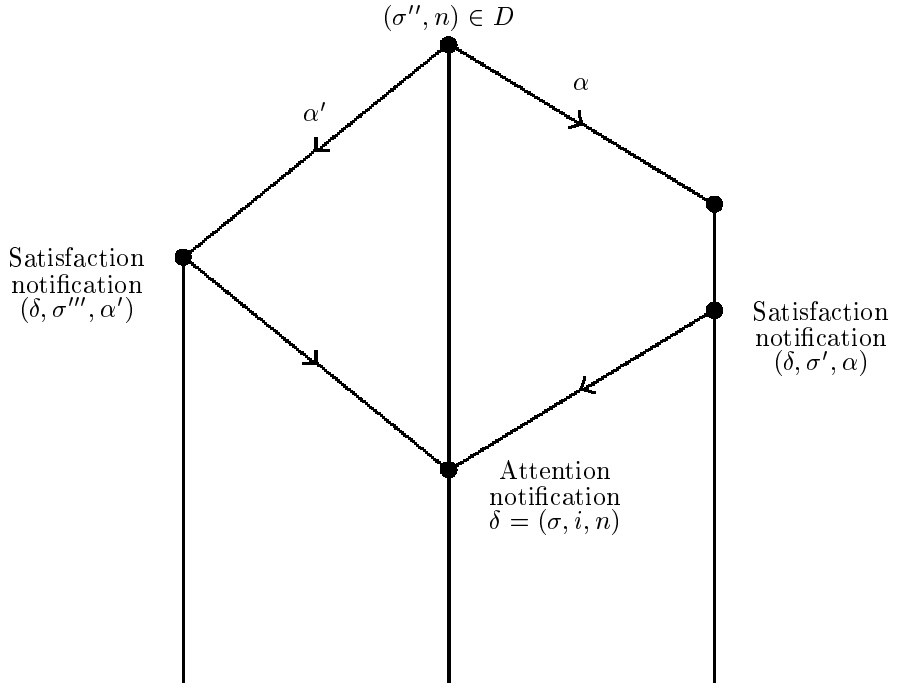


Fig. 3

There are three significant problems which remain in applying the techniques we have just described. It is actually in dealing with these problems that the construction becomes complicated and we shall finish here just by giving some impression of the approach that is required. The first of these problems is the fact that Turing functionals may be partial. Thus we must consider witnesses for N requirements which have associated with them a 'guess' as to which of the relevant Turing functionals will be partial. In [4] we define the notion of a partiality guess γ for i. Since the details are a little involved we shall omit them here. The second problem of some significance is caused by the restraints to which an R requirement may be subject. Consider the following sequence of events. At some stage of the construction we wish to act in order to satisfy the requirement N_i with respect to a string σ. Thus we wish to enumerate a witness n, which has associated with it a partiality guess γ for i, into $D(X)$ for all $X \supset \sigma$. Rather than doing so we enumerate an attention notification δ for the reasons that have been discussed. Let τ be the shortest initial segment of $E(\sigma)$ for which $\Lambda_i^\tau(n) \downarrow = 0$. Upon enumerating the attention notification we enumerate the N-demand (H, α, m, t) into $T(\delta)$ (since γ does not assert the partiality of α). At some subsequent stage of the construction s we find $\sigma' \supset \sigma$ and σ'' such that $l_{s,\alpha}(\sigma', \sigma'')$ and such that the N-demand (H, α, m, t) is not consistent with σ''. Now there may exist $X \supset \sigma'$, α' an initial segment of α and Y such that $l_{\alpha'}(X, Y)$ and such that the R restraints on Y are always large enough so that,

despite the fact that (H, α, m, t) is not consistent with σ'', it remains the case, $\tau \subset E(X)$. Thus, even in the case that γ is correct as a partiality guess for i with respect to X, the requirement N_i is not satisfied. Therefore we must actually consider witnesses for N requirements which have associated with them, not only a partiality guess, but also a restraint function for that partiality guess. The restraint function should be thought of as a guess as regards each value $\liminf_s R(\sigma_s^Y, i)$, such that $l_i(X, Y)$ and as regards the finite number of times that restraints will drop below each such value. Having established the precise nature of the witnesses that we shall use in order to satisfy N requirements we must then find some way of ensuring that the restraints imposed by N requirements drop infinitely often along the length of any given $X \subseteq \omega$ – for all remaining details we refer the reader to [4].

References

1. Friedberg, R.M. *Two recursively enumerable sets of incomparable degrees of unsolvability,* Proc.Nat.Ac.Sci. 43 (1957), 236-238.
2. Kechris, A.S. and Moschovakis, Y. *Cabal Seminar 76-77,* LNMS 689 Springer-Verlag, Berlin 1978.
3. Lachlan, A.H. *Uniform enumeration operators,* Journal of Symbolic Logic 40 (1975), 401-409.
4. Lewis, A.E.M. *A partial solution to a question of Sacks,* unpublished.
5. Muchnik, A.A. *On the unsolvability of the problem of reducability in the theory of algorithms,* Dokl.Akad.Nauk. SSSR N.S. 108 (1956), 29-32.
6. Post, E.L. *Recursively enumerable sets of positive integers and their decision problems,* Bull.Am.Math.Soc 50 (1944) 284-316.
7. Sacks, G.E. *Degrees of unsolvability,* Annals of Math. Studies 55, Princeton University Press.
8. Slaman, T and Steel, J.R. *Definable functions on degrees* Cabal Seminar 81-85.
9. Steel, J.R. *A classification of jump operators,* Journal of Symbolic Logic 47 (1982) 347-358.
10. Xiaoding, Y. *Post's problem and degree invariant functions,* (1996) Unpublished.

The Low Splitting Theorem in the Difference Hierarchy

Angsheng Li*

Institute of Software,
Chinese Academy of Sciences,
Beijing 100080, P. R. China
angsheng@gcl.iscas.ac.cn

Abstract. It is shown that for any 2-*computably enumerable* Turing degrees $\boldsymbol{a}$, $\boldsymbol{l}$, if $\boldsymbol{l}' = \boldsymbol{0}'$, and $\boldsymbol{l} < \boldsymbol{a}$, then there are 2-computably enumerable Turing degrees $\boldsymbol{x}_0$, $\boldsymbol{x}_1$ such that both $\boldsymbol{l} \leq \boldsymbol{x}_0$, $\boldsymbol{x}_1 < \boldsymbol{a}$ and $\boldsymbol{x}_0 \vee \boldsymbol{x}_1 = \boldsymbol{a}$ hold, extending the Robinson low splitting theorem for the computably enumerable degrees to the difference hierarchy.

1 Introduction

A set $A \subseteq \omega$ is called *computably enumerable* (c.e., for short) if there is an algorithm to enumerate the elements of it. Ershov defined the difference hierarchy of sets by generalising the notion of a computably enumerable set.

Definition 1 (Ershov, [11],[12]). *A set $A \subseteq \omega$ is called n-computably enumerable (n-c.e., for short), if there is a computable function g such that for all $x \in \omega$,*
(a) $g(x, 0) = 0$,
(b) $\lim_s g(x, s) = A(x)$, and
(c) $|\{s \mid g(x, s) \neq g(x, s + 1)\}| \leq n$.

For $n = 0$, the only 0-c.e. set is just the empty set $\emptyset$, the 1-c.e. sets are the c.e. sets, and the 2-c.e. sets are the differences of two c.e. sets. Because of the latter coincidence, we also call the 2-c.e. sets d.c.e. sets. In [11], [12], Ershov has extended the definition to all computable ordinals, providing a hierarchy that resolves the Δ^0_2 subsets of ω, the sets which can be approximated by a computable function. For $n \geq 1$, we say that a Turing degree is *n-computably enumerable* (n-c.e., for short), if it contains an n-c.e. set. Let $\mathcal{E}_n$ be the set of all n-c.e. Turing degrees. Then $\mathcal{E}_1 = \mathcal{C}$, the set of all c.e. Turing degrees, and it is also known that for all n, $\mathcal{E}_n \subset \mathcal{E}_{n+1}$. Lachlan observed that for $n \geq 1$, the n-c.e. Turing degrees are downward dense.

* The author is partially supported by National Distinguished Young Investigator Award No. 60325206 and by NSFC Grand International Joint Project, New Directions in Theory and Applications of Models of Computation, No. 60310213 (P. R. China).

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 287–296, 2005.

Lemma 2 (Lachlan). *Given any n-c.e. degree* $\boldsymbol{a} > \mathbf{0}$*, there is a c.e. degree* $\boldsymbol{b}$ *such that* $\boldsymbol{a} > \boldsymbol{b} > \mathbf{0}$*.*

The Lachlan's lemma gives an elementary difference between the global Turing degrees and the n-c.e. Turing degrees below any nonzero c.e. degree, because in the Δ_2^0 Turing degrees, every nonzero c.e. degree bounds a minimal degree (Yates, 1970, [4], [16]). Then an interesting question is whether or not the c.e. degrees and the n-c.e. degrees are elementarily equivalent for $n \geq 2$. This was answered negatively by Arslanov [1] by proving that every nonzero n-c.e. degree joins to $\mathbf{0'}$ with a low d.c.e. degree, while in the c.e. case, Cooper [5] and Yates (unpublished) proved that there is a nonzero c.e. degree $\boldsymbol{a}$ such that for any c.e. degree $\boldsymbol{x}$, $\boldsymbol{a} \vee \boldsymbol{x} = \mathbf{0'}$ if and only if $\boldsymbol{x} = \mathbf{0'}$ ($\boldsymbol{a}$ is called *noncuppable*). This theorem gives an elementary difference between the c.e. degrees and the n-c.e. degrees, for every $n \geq 2$, and also indicates that, for every $n \geq 2$, the ideal of n-c.e. degrees noncuppable in $\mathcal{E}_n$ is trivial, i.e., $\{\mathbf{0}\}$. Downey [10] proved that the diamond lattice can be embedded into the 2-c.e. degrees preserving both 0 and 1, giving another difference between the c.e. degrees and the n-c.e. degrees for all $n \geq 2$, because Lachlan [13] had previously shown that the diamond lattice cannot be embedded into the c.e. degrees preserving both 0 and 1. Furthermore, Li and Yi [14] have shown that there are two incomplete d.c.e. degrees $\boldsymbol{a}_0$, and $\boldsymbol{a}_1$ such that every nonzero n-c.e. degree joins to $\mathbf{0'}$ with one of them, extending both the Arslanov's cupping theorem and Downey's diamond theorem. In fact, the n-c.e. degrees for $n \geq 2$ are not even dense, provided by Cooper et al. [7], in contrast with the Sacks' density theorem of the c.e. degrees [17].

On the other hand, the n-c.e. degrees for $n \geq 2$ do share some "nice" properties with the computably enumerable degrees. Cooper [6] proved that the Sacks' splitting theorem [16] can be generalised to the n-c.e. degrees for $n \geq 2$: For any nonzero n-c.e. degree $\boldsymbol{a}$, there are n-c.e. degrees $\boldsymbol{b}, \boldsymbol{c}$ such that $\boldsymbol{b} < \boldsymbol{a}$, $\boldsymbol{c} < \boldsymbol{a}$ and $\boldsymbol{b} \vee \boldsymbol{c} = \boldsymbol{a}$. This theorem was further improved by Cooper and Li [9]:

Theorem 3. *For any 2-c.e. degree* $\boldsymbol{a}$*, and c.e. degree* $\boldsymbol{b}$*, if* $\boldsymbol{b} < \boldsymbol{a}$*, then there are 2-c.e. degrees* $\boldsymbol{x}_0$*,* $\boldsymbol{x}_1$ *such that both* $\boldsymbol{b} < \boldsymbol{x_0}$, $\boldsymbol{x_1} < \boldsymbol{a}$ *and* $\boldsymbol{x}_0 \vee \boldsymbol{x}_1 = \boldsymbol{a}$ *hold.*

The theorem suggests an interesting question on whether or not the 2-c.e. degrees $\boldsymbol{a}$ such that every 2-c.e. degree above it is splittable over it in the 2-c.e. degrees are the same as the c.e. degrees.

Noting that in the c.e. degrees, Sacks' splitting theorem can be extended with jump restrictions, Robinson [15] showed the following:

Theorem 4 (Robinson Low Splitting Theorem). *For any c.e. degrees* $\boldsymbol{l} < \boldsymbol{a}$*, if* $\boldsymbol{l'} = \mathbf{0'}$*, then there are c.e. degrees* $\boldsymbol{x}_0$*,* $\boldsymbol{x}_1$ *such that both* $\boldsymbol{l} < \boldsymbol{x}_0, \boldsymbol{x}_1 < \boldsymbol{a}$*, and* $\boldsymbol{x}_0 \vee \boldsymbol{x}_1 = \boldsymbol{a}$ *hold.*

It is natural to ask whether the Robinson low splitting theorem can be extended to the finite levels of the difference hierarchy. In the present paper, we

try to answer this question. Our main theorem is an extension of the Robinson low splitting theorem to the difference hierarchy at level 2.

Theorem 5. *Let m be a natural number such that $0 < m \leq 2$. Then for any m-c.e. set A, any 2-c.e. set L, if $L' \leq_{\mathrm{T}} \emptyset'$, then there are 2-c.e. sets X_0, and X_1 with the following properties,*

(1) $X_0 \leq_{\mathrm{T}} A \oplus L$, $X_1 \leq_{\mathrm{T}} A \oplus L$, and $A \leq_{\mathrm{T}} X_0 \oplus X_1$,

(2) If $A \leq_{\mathrm{T}} X_i \oplus L$ for some $i \leq 1$, then there is an $(m-1)$-c.e. set I such that both $I \leq_{\mathrm{T}} A \oplus L$ and $A \leq_{\mathrm{T}} I \oplus L$ hold.

In fact, the result in the case of $m = 1$ has already been proved previously by Arslanov, Cooper and Li in [3]. We consider only the case of $m = 2$. The proof for Theorem 5 is a non-uniform inductive argument. By applying the result, we have a generalisation of the Robinson's low splitting theorem in the difference hierarchy.

Theorem 6. *For any 2-c.e. degrees $\boldsymbol{a}, \boldsymbol{l}$, if $\boldsymbol{l} < \boldsymbol{a}$ and $\boldsymbol{l'} = \boldsymbol{0'}$, then there are 2-c.e. degrees $\boldsymbol{x}_0$, and $\boldsymbol{x}_1$ such that both $\boldsymbol{l} < \boldsymbol{x}_0$, $\boldsymbol{x}_1 < \boldsymbol{a}$, and $\boldsymbol{a} = \boldsymbol{x}_0 \vee \boldsymbol{x}_1$ hold.*

Proof. By applying Theorem 5 twice. □

To analyse more applications of Theorem 6, we look at some previous results. Using the techniques in the proof of the Sacks' density theorem, Cooper, Lempp and Watson [8] showed that for any $n > 1$, any c.e. degrees $\boldsymbol{b} < \boldsymbol{a}$, there is a properly n-c.e. degree $\boldsymbol{c}$ such that $\boldsymbol{b} < \boldsymbol{c} < \boldsymbol{a}$. Therefore, for any $n > 1$, there is a low proper n-c.e. degree $\boldsymbol{l}$. Let P_n, and Q_n be the sets of all n-c.e. degrees $\boldsymbol{x}$ above which every n-c.e. degree $\boldsymbol{y}$ is splittable over $\boldsymbol{x}$ in the n-c.e. degrees, and of all n-c.e. degrees $\boldsymbol{x}$ such that for every n-c.e. degree $\boldsymbol{y}$, if $\boldsymbol{y} > \boldsymbol{x}$, then there is an n-c.e. degree $\boldsymbol{z}$ satisfying $\boldsymbol{y} > \boldsymbol{z} > \boldsymbol{x}$, respectively.

By Theorem 6, we have that both (i) and (ii) below hold,

(i) $\mathcal{C} \subset P_2$,

(ii) $\mathcal{C} \subset Q_2$,

where $\mathcal{C}$ is the set of all computably enumerable degrees.

This refutes an appealing candidate of possible definitions of the computably enumerable degrees in the d.c.e. Turing degrees. The result stated in (ii) above answers a question by Arslanov in [2] (Question 2.6). However it is still open whether or not for any natural number n, there is a low splitting theorem for n-c.e. degrees.

In this article, we sketch a proof of our main result, Theorem 5. After formulating the conditions of the theorem according to requirements, we describe the strategies to satisfy the requirements, and analyse the consistency of the strategies so that a priority argument can work for a full construction and the verification of the construction. Our notation and terminology are standard, and generally follow that of Soare [18]. A few special notations are taken from the paper Cooper and Li [9].

2 The Requirements and the Strategies

We first formulate the requirements.

2.1 The Requirements

Suppose that A, L are 2-c.e. sets such that $L' \leq_{\mathrm{T}} \emptyset'$. Let Θ be a Turing functional such that $\Theta(K)$ is total and $L' = \Theta(K)$, where $L' = \{x \mid \Psi_x(L;x)\downarrow\}$, $\{\Psi_x \mid x \in \omega\}$ is an effective enumeration of all Turing functionals Ψ.

To prove Theorem 5, we construct 2-c.e. sets X_0, X_1 to satisfy the following requirements,

$\mathcal{T}$: $X_0,\ X_1 \leq_{\mathrm{T}} A \oplus L$, $A \leq_{\mathrm{T}} X_0 \oplus X_1$

$\mathcal{R}_{2i}$: $A = \Phi_i(X_0, L) \rightarrow (\exists I_{2i}$ computably enumerable $)[I_{2i} \leq_{\mathrm{T}} A \oplus L \ \&\ A \leq_{\mathrm{T}} I_{2i} \oplus L]$

$\mathcal{R}_{2i+1}$: $A = \Phi_i(X_1, L) \rightarrow (\exists I_{2i+1}$ computably enumerable$)[I_{2i+1} \leq_{\mathrm{T}} A \oplus L \ \&\ A \leq_{\mathrm{T}} I_{2i+1} \oplus L]$

where $i \in \omega$, $\{\Phi_i \mid i \in \omega\}$ is an effective enumeration of all Turing functionals Φ. Furthermore, if A is a computably enumerable set, then the I_j in requirement $\mathcal{R}_j$ is the empty set $\emptyset$.

Clearly meeting the requirements is sufficient to prove the theorem.

2.2 The Lowness Approximation

Since $L' \leq_{\mathrm{T}} \emptyset'$, we suppose that Θ is a Turing functional such that $\Theta(K)$ is total and $L' = \Theta(K)$. The lowness condition of L is necessary, with which we will construct an auxiliary Turing functional Δ. By the s-m-n Theorem, there is a computable function f such that for all x, y,

$$\Delta(L; x, y) = \Psi_{f(x)}(L; y).$$

By the Recursion Theorem, we assume that the computable function f is given in advance of the construction.

Δ will be built as a computably enumerable set of axioms of the form $(\sigma, w, f(w))$ for $\sigma \in \omega^{<2}$, and for $w \in \omega$. We say that $(\sigma, w, f(w)) \in \Delta$ is *invalid at stage* s, if there is an l such that $\sigma(l)\downarrow \neq L_s(l)$, and $|\{v \mid v < s, L_v(l) \neq L_{v+1}(l)\}| = 2$, and *valid at stage* s, otherwise; and that $(\sigma, w, f(w)) \in \Delta$ is *correct at stage* s, if $\sigma \subset L_s$ holds, in the sense that σ is an initial segment of the characteristic function of L_s, and *incorrect at stage* s, otherwise.

We define $\Delta(L; w, f(w))\downarrow = 0$ if and only if there is an axiom $(\sigma, w, f(w)) \in \Delta$ such that $\sigma \subset L$ holds.

We say that s is a *progressive stage*, if for every witness w defined by some strategy, we have that

(i) $\Theta(K; f(w))\downarrow$,
(ii) If $\Theta(K; f(w))\downarrow = 0$, then $\Delta(L; w, f(w))\uparrow$, and
(iii) If $\Theta(K; f(w))\downarrow = 1$, then $\Delta(L; w, f(w))\downarrow$.

We build Δ at progressive stages, and we define $\Delta(L; w, f(w))$, only if $\Theta(K; f(w))\downarrow = 0$, and whenever we define Δ, we define $\Delta(L; w, f(w))\downarrow = 0$.

Suppose that we have defined $\Delta(L; w, f(w)) \downarrow = 0$ at stage v. Then at the next progressive stage $s > v$, we have that either (1) or (2) below occurs,

(1) $\Delta(L; w, f(w))$ is undefined at the beginning of stage s,
(2) $\Theta(K; f(w)) \downarrow = 1$ holds at the beginning of stage s.

The whole construction will be delayed at non-progressive stages, in the sense that no axiom for any functional can be built at these stages.

2.3 The $\mathcal{T}$-Strategy

To satisfy $X_0 \leq_{\mathrm{T}} A \oplus L$ and $X_1 \leq_{\mathrm{T}} A \oplus L$, we build Turing functionals Ξ_0 and Ξ_1 such that both $X_0 = \Xi_0(A, L)$ and $X_1 = \Xi_1(A, L)$ hold. Ξ_0 and Ξ_1 are built similarly, so we only describe the definition of Ξ_0. Intuitively speaking, Ξ_0 will be built as a computably enumerable set of axioms of the form (σ, τ, y, z). Then we define $\Xi_0(A, L; y) \downarrow = z$ if and only if there is an axiom $(\sigma, \tau, y, z) \in \Xi_0$ such that both $\sigma \subset A$ and $\tau \subset L$ hold. We say that an axiom $(\sigma, \tau, y, z) \in \Xi_0$ is *invalid at stage* s, if either (i) or (ii) below holds,

(i) There is an a such that $\sigma(a) \downarrow \neq A_s(a)$, and $|\{v < s \mid A_v(a) \neq A_{v+1}(a)\}| = 2$.
(ii) There is an l such that $\tau(l) \downarrow \neq L_s(l)$ and $|\{v < s \mid L_v(l) \neq L_{v+1}(l)\}| = 2$.

If an axiom $(\sigma, \tau, y, z) \in \Xi_0$ is not invalid at stage s, then we say that it is *valid at stage* s. We say that an axiom $(\sigma, \tau, y, z) \in \Xi_0$ is *correct at stage* s, if both $\sigma \subset A_s$ and $\tau \subset L_s$ hold, and *incorrect at stage* s otherwise.

To satisfy $X_0 = \Xi_0(A, L)$, we ensure that the definition of Ξ_0 and X_0 will satisfy the following properties:

(i) If an axiom (σ, τ, y, z) is enumerated into Ξ_0 at stage s, then $\sigma \subset A_s$, $\tau \subset L_s$, and $z = X_{0,s}(y)$ hold.
(ii) We enumerate an element y into X_0 at stage s, only if every axiom $(\sigma, \tau, y, 0) \in \Xi_0$ is incorrect at stage s.
(iii) We extract a number y from X_0 at stage s, only if every axiom $(\sigma, \tau, y, 1) \in \Xi_0$ is invalid at stage s.
(iv) For a fixed y, there are only finitely many axioms (σ, τ, y, z) which are enumerated into Ξ_0 during the course of the construction.

We say that (i)–(iv) are Ξ_0-*rules*. By the Ξ_0-rules, if $\Xi_0(A, L)$ is total, then $\Xi_0(A, L) = X_0$. In addition, we also ensure that for a fixed y, $\Xi_0(A, L; y)$ will be defined eventually. Therefore $\Xi_0(A, L) = X_0$.

To satisfy $A \leq_{\mathrm{T}} X_0 \oplus X_1$, we define, for each x, a *use block* U_x *of* x as follows:

(i) Define $U_0 = \{0, 1\}$.
Let y be the greatest element $z \in U_x$.
(ii) Define $U_{x+1} = [y + 1, y + (x + 2)2^{x+2}]$.

Notice that for each $y \in U_x$, we have that $y \geq x$, and that the union of all U_x is exactly ω.

We say that an element $y \in U_x$ is an x-*trace*, or *a trace of* x.

To realise the reduction of A to $X_0 \oplus X_1$, we ensure that for every x, we have the following

$$x \in A \text{ if and only if } (\exists y \in U_x)[X_0(y) \neq X_1(y)].$$

We call this the *splitting rule of the construction*. It is easy to see that the splitting rule ensures that $A \leq_{\mathrm{T}} X_0 \oplus X_1$. $\mathcal{T}$ is satisfied.

Before moving to the $\mathcal{R}$-strategy, we introduce the instructions to split an $x \in A$ into $X_0 \oplus X_1$.

During the course of the construction, for every j, we define r_j to be the *restraint* of the $\mathcal{R}_j$-strategy. We define for each x, a *height function* $h(x)$ *of* x according to the restraints r_j, and an *upward permitting marker* $m(x)$ of x.

We define $h(x)$ to be the least k such that $k < x \leq r_k$, and $m(x)$ to be the $A \oplus L$*-permitting marker* corresponding to r_k, where $k = h(x)$.

If $x \in A$ and $h(x) = k$, then we enumerate the least unused $y \in U_x$ into X_i, where $i = (h(x) + 1)(\mathrm{mod}\ 2)$.

However if $A \oplus L$ changes such that r_k for $k = h(x)$ drops, then $A \oplus L$ changes below $m(x)$, in which case, we redefine the height function $h(x)$ to be some number $k' > k$. In this sense, we call $m(x)$ the upward permitting marker of x.

For every x, if $x \in A$, the x*-splitting* will maximize its height function $h(x)$, which is of course bounded by x.

In contrast to the upward permitting marker, we may define a *downward permitting marker of* x, denoted by $p(x)$ as follows.

Suppose that $x \in A$, and $h(x) = h$. For every $j < h$, suppose that $(\sigma_1, w_1, f(w_1)), \cdots, (\sigma_l, w_l, f(w_l))$ are all Δ-axioms which have been defined by the $\mathcal{R}_j$-strategy, which are incorrect, and which may be injured by the x-splitting, then we define the j-th *downward permitting marker of* x by

$$p_j(x) = \max\{|\sigma_k| \mid k = 1, 2, \cdots, l\}.$$

And define $p(x)$ by

$$p(x) = \max\{p_j(x) \mid j < h(x)\}.$$

The intuition is as follows. Suppose that $(\sigma, w, f(w)) \in \Delta$ is an axiom defined by some $\mathcal{R}_j$-strategy for some $j < h(x)$. Since now the axiom for $\Delta(L; w, f(w))$ is not L-correct, so that the x-splitting may injure the Φ-computation, at the same time, the ξ_i-use for every $y \in U_x$ is lifted to be greater than $|\sigma|$. So if at a later stage, σ becomes correct, in the sense that $\sigma \subset L$, some l such that $\sigma(l) = 1$ leaves L, a permanent $A \oplus L$-permission occurs for every $y \in U_x$, then the x-splitting will require $\mathcal{R}_j$*-attention* by redefining the height function $h(x)$ of x to be $\leq j$. In so doing, the x-splitting will not injure the $\mathcal{R}_j$-strategy anymore.

One of the key points is to make sure that for a fixed $x > j$, the x-splitting requires $\mathcal{R}_j$-attention at most 2^j many times. This allows us to determine the size of the use block U_x in advance of the construction.

With this intuition in mind, we look at the $\mathcal{R}$-strategies in the next subsection.

2.4 An $\mathcal{R}$-Strategy

Suppose that $j = 2i$ for some i, and that we want to satisfy the following requirement:

$\mathcal{R}_j$: $A = \Phi_i(X_0, L) \to (\exists I_j, \text{c.e. })[I_j \leq_{\mathrm{T}} A \oplus L \ \&\ A \leq_{\mathrm{T}} I_j \oplus L]$.

Before describing the $\mathcal{R}$-strategy, we introduce the idea how to preserve a computation $\Phi_i(X_0, L; k)[v] \downarrow = 0$ say. We open a cycle k of the $\mathcal{R}$-strategy. Cycle k will define a *witness* w, say, as fresh. At a stage v, say, at which we have that both $\Theta(K; f(w)) \downarrow = 0$ and $\Phi_i(X_0, L; k) \downarrow = A(k)$ hold, define $\Delta(L; w, f(w)) \downarrow = 0$ with $\delta(w, f(w)) = a$ for some $a \geq \phi_i(k)[v]$, and define $\Gamma_j(I_j, L; k) \downarrow = A(k)$ with $\gamma_j(k)$ fresh and with L-use a. If at a later stage $s > v$ at which $\Phi_i(X_0, L; k)[v]$ becomes non-recoverable due to an X_0-extraction, then either there is an $x \leq a$ such that $x \in A_v - A_s$, in which case, we enumerate $\gamma_j(k)[v]$ into I_j and cancel the witness w, or there is an $x \leq a$ such that $x \in L_v - L_s$, in which case, both $\Delta(L; w, f(w))[v]$ and $\Gamma_j(I_j, L; k)[v]$ become invalid at stage s.

Therefore if both $\Delta(L; w, f(w))$ and $\Gamma_j(I_j, L; k)$ are defined, then $\Phi_i(X_0, L; k)[v]$ has not been injured by redefining the height function of $x \leq \phi_i(k)[v]$ for those x which had been enumerated into A before the $\Gamma_j(I_j, L; k)[v]$ was defined at stage v. On the other hand for any x, if x will enter A after stage v, and $x \leq \phi_i(k)[v]$, then the height function $h(x)$ will be less than or equal to j unless $\Phi_i(X_0, L; k)[v]$ is L-incorrect, in which case, the height function $h(x)$ maybe defined to be some $n > j$, under the condition that the x-splitting may require $\mathcal{R}_j$-attention at the stage at which $\Phi_i(X_0, L; k)[v]$ becomes L-correct. This will be realised by requiring that the $\xi_i(y)$-use for all $y \in U_x$ will be greater than $\phi_i(k)[v]$, before an x-splitting occurs.

In summary, we ensure that if both $\Delta(L; w, f(w))$ and $\Gamma_j(I_j, L; k)$ are defined, then $\Phi_i(X_0, L; k)[v]$ has not been injured by x-splitting for any x which was in A and which had been split before stage v. And for any x, if x enters A (or x-splitting occurs for the first time) after stage v, then the injury of $\Phi_i(X_0, L; k)[v]$ by the x-splitting can be restored at the stage at which $\Phi_i(X_0, L; k)[v]$ becomes L-correct. With this intuition, we now describe the instructions of the $\mathcal{R}_j$-strategy.

To satisfy the $\mathcal{R}$-requirement $\mathcal{R}_j$, we construct a computably enumerable set I_j and Turing functionals Γ_j and Λ_j such that the following properties hold:

(a) $A = \Gamma_j(I_j, L)$.
(b) $I_j = \Lambda_j(A, L)$.
(c) If A is a computably enumerable set, then $I_j = \emptyset$.

Λ_j is built by a usual permitting method, but we enumerate an element x into I_j, only if there is some $y < x$ which leaves A. Therefore if A is a computably enumerable set, then I_j is $\emptyset$.

Γ_j will be built by an infinite sequence of cycles $k \geq 0$. Cycle k of the $\mathcal{R}_j$-strategy will be responsible for defining $\Gamma_j(I_j, L; k)$, and will proceed as follows.

1. Define a witness $w(j, k)$ as fresh.
2. Wait for a stage v at which
   (2a) for any $k' < k$, if $w(j, k') \downarrow = w'$, then $\Theta(K; f(w')) \downarrow = 1$,

(2b) for $w = w(j,k)$, $\Theta(K; f(w)) \downarrow = 0$ holds,
(2c) $l(\Phi_i(X_0, L), A) > k$,
then:
– define $a(j,k,w) = \max\{\phi_i(k), m(x), p(x) \mid x \leq \phi_i(k), x \in A, h(x) \downarrow\}$,
– define $\Delta(L; w, f(w)) \downarrow = 0$ with $\delta(w, f(w)) = a(j,k,w)$,
– define $\Gamma_j(I_j, L; k) \downarrow = A(k)$ with $\gamma_j(k)$ fresh, and with L-use $a(j,k,w)$,
– let $\sigma = L \upharpoonright (a(j,k,w) + 1)$,
– create a *conditional restraint* $\boldsymbol{c}_j(k) = (j,k,w,\sigma,\phi_i(k),v)$, and
– let C_j be the set of all conditional restraints $\boldsymbol{c}_j(k) = (j,k,w,\sigma,r,v)$ which are still valid.

[**Remark.** (1). The conditional restraint $\boldsymbol{c}_j(k) = (j,k,w,\sigma,r,v)$ allows an x-splitting for $x \leq r$ to enumerate a $y \in U_x$ into X_0 if σ is both valid and incorrect, in which case, the Ξ_i*-permitting markers* for all $y \in U_x$ have been lifted to $|\sigma|$. In so doing if at a later stage, σ proves a correct initial segment of L, we get a permanent $A \oplus L$-permission for all $y \in U_x$ via Ξ_i for both $i = 0$ and 1, and then we can restore the computation $\Phi_i(X_0, L; k)[v]$ by extracting the x-traces which have been enumerated into X_0 since stage v from X_0. In this case, we say that the x-splitting receives $\mathcal{R}_j$*-attention*.
(2). For a given x, the x-splitting receives $\mathcal{R}_j$-attention at most once unless it has received $\mathcal{R}_{j'}$-attention for some $j' < j$.
(3). The definition of $a(j,k,w)$ will ensure that if $\Delta(L; w, f(w))$ is valid (of course w has not been cancelled), then $\Phi_i(X_0, L; k)[v]$ has not been injured by any x-splitting for those x with which $h(x)$ had been defined at the stage at which we defined $\Delta(L; w, f(w))$.
(4). If we build Γ_j for some j at a stage v, say, there is no x-splitting occurs during stage v.
(5) After we defined $\Delta(L; w, f(w))$ at stage v, we will wait for the next progressive stage s say. Now we know that either $\Theta(K; f(w)) \downarrow = 1$, or $L_v \upharpoonright (\delta(w, f(w))[v] + 1)$ has changed, or w can be cancelled during stage s.]

3. If at a stage $s > v$, there is $a \leq a(j,k,w)[v]$ such that $a \in A_v - A_s$, then
   – enumerate $\gamma_j(k)[v]$ into I_j,
   – set w to be undefined,
   – cancel cycle k' for all $k' > k$, and go back to step 1.
4. Otherwise, and there is an l such that $l \leq a(j,k,w)[v]$ and $l \in L_v - L_s$, then both $\Delta(L; w, f(w))[v]$ and $\Gamma_j(I_j, L; k)[v]$ become invalid automatically.
5. (*Special Attention*) If at stage $s > v$:
   (5a) $\Theta(K; f(w)) \downarrow = 1$,
   (5b) $\Delta(L; w, f(w)) \downarrow$,
   then:

   – let $\boldsymbol{c}_j(k) = (j,k,w,\sigma,r,v)$ be the conditional restraint defined by the $\mathcal{R}_j$-strategy at the stage we defined the current $\Delta(L; w, f(w))$,
   – for every x, if $x \leq r$, $h(x)[v] \uparrow$, and $h(x) \downarrow > j$, then extract every $y \in U_x$ from $X_0 \cup X_1$, and we say that x-splitting receives $\mathcal{R}_j$-attention at stage s,
   – let R_j be the set of all conditional restraints $\boldsymbol{c}_j(k') = (j,k',w',\sigma',r',v')$ which are still valid at stage s,

– define the X_0-*restraint* of the $\mathcal{R}_j$-strategy by $r_j = \max\{r \mid (\exists k', w', \sigma', r, v')[(j, k', w', \sigma', r, v') \in R_j]\}$, and
– we say that the $\mathcal{R}_j$-strategy *receives special attention*.

The Possible Outcomes The *possible outcomes of the $\mathcal{R}$-strategy* are as follows.

Case 1. There is a k such that step 3 of cycle k of the $\mathcal{R}_j$-strategy acts infinitely many times. In this case, both $\Gamma_j(I_j, L)$ and $\Phi_i(X_0, L)$ are partial, $\mathcal{R}_j$ is satisfied.

Note that there are only finitely many conditional restraints $(j, k', w', \sigma, r, v)$ which are created by the $\mathcal{R}_j$-strategy, and which are valid permanently. So we have that $\liminf_s r_j[s] \downarrow = r_j < \omega$ exists.

Case 2. Otherwise, and there is a k (the least) such that step 2 of cycle k of the $\mathcal{R}_j$-strategy acts infinitely often. In this case, $\lim_s w(j, k)[s] \downarrow = w < \omega$ exists, and $\Theta(K; f(w)) \downarrow = 0$, but $a(j, k, w)[s]$ is unbounded during the course of the construction, so is $\phi_i(k)[s]$. $\mathcal{R}_j$ is satisfied.

Notice that there are only finitely many stages at which cycle k' of the $\mathcal{R}_j$-strategy acts for $k' \neq k$. And almost every conditional restraint (j, k, w, σ, r, v) is either invalid or incorrect at every progressive stage.

By the action in step 5 of the $\mathcal{R}_j$-strategy, R_j is a finite set, so that $\lim_s r_j[s] \downarrow = r_j < \omega$ exists.

Case 3. Otherwise, and $\Gamma_j(I_j, L)$ is built infinitely many times. In this case, $\Gamma_j(I_j, L)$ is total. If there is a k such that $\Gamma_j(I_j, L; k) \downarrow \neq A(k)$ holds permanently, then by step 5 of the $\mathcal{R}_j$-strategy, $\Phi_i(X_0, L; k) \downarrow \neq A(k)$ holds permanently. By condition (2c) in step 2 of the $\mathcal{R}_j$-strategy, the $\mathcal{R}_j$-strategy acts only finitely many times. It is impossible that $\Gamma_j(I_j, L)$ is built infinitely often.

Therefore, in this case, for every k, we have that $\Gamma_j(I_j, L; k) \downarrow = A(k)$, giving $A = \Gamma_j(I_j, L)$. In addition, I_j is reducible to $A \oplus L$. So for every j, $\mathcal{R}_j$ is satisfied.

Case 4. Otherwise. Then let k be the least x such that $\Gamma_j(I_j, L; x)$ diverges. Then $\lim_s w(j, k)[s] \downarrow = w < \omega$ exists. By the choice of k, $\Delta(L; w, f(w))$ diverges. Therefore $\Theta(K; f(w)) \downarrow = 0$ holds. And for almost every progressive stage s, $l(\Phi_i(X_0, L), A)[s] \not> k$ occurs at stage s. $\mathcal{R}_j$ is satisfied, and the $\mathcal{R}_j$-strategy acts only finitely many times.

In summary, if there is no c.e. I such that $I \oplus L \equiv_{\mathrm{T}} A \oplus L$, then we have the following:

(a) $\mathcal{R}_j$ is satisfied.
(b) $\Gamma_j(I_j, L)$ is partial, and
(c) $\liminf_s r_j[s] \downarrow = r_j < \omega$ exists.

Property (c) ensures that for a fixed j, for almost every x, the height function $h(x)$ of x will be eventually greater than j. This again ensures that an $\mathcal{R}_j$-strategy is injured permanently only finitely many times by x-splitting for those x with height function $h(x) < j$. So every $\mathcal{R}_j$ is satisfied eventually.

Based on the strategies, we can build a priority construction, and show that all the requirements are satisfied, establishing the main result, Theorem 5.

References

1. M. M. Arslanov, Structural properties of the degrees below $\mathbf{0'}$, Dokl. Akad. Nauk SSSR (N.S.) 283 (1985), 270–273.
2. M. M. Arslanov, Open questions about the n-c.e. degrees, in Computability Theory and Its Applications: Current Trends and Open Problems (M. Merman, P. Cholak, S. Lempp and R. A. Shore, eds.), Contemporary Mathematics 257 (2000), 15–22.
3. M. M. Arslanov, S. B. Cooper and A. Li, There is no low maximal d.c.e. degree, Math. Logic Quart. 46 (2000), 409–416; 50, No. 6 628–636 (2004).
4. S. B. Cooper, Minimal degrees and the jump operator, J. Symbolic Logic 38 (1973) 249–271.
5. S. B. Cooper, On a theorem of C. E. M. Yates, 1974, handwritten notes.
6. S. B. Cooper, A splitting theorem for the n-r.e. degrees, Proc. Amer. Math. Soc. 115 (1992) 461–471.
7. S. B. Cooper, L. Harrington, A. H. Lachlan, S. Lempp, and R. I. Soare, The d-r.e. degrees are not dense, Ann. Pure Appl. Logic 55 (1991) 125–151.
8. S. B. Cooper, and S. Lempp, and P. Watson, Weak density and cupping in the d-r.e. degrees, Israel J. Math. 67 (1989) 137–152.
9. S. B. Cooper and A. Li, Turing definability in the Ershov hierarchy, J. London Math. Soc. (2) 66 (2002) 513–528.
10. R. Downey, D.r.e. degrees and the nondiamond theorem, Bull. London Math. Soc. 21 (1989) 43–50.
11. Y. L. Ershov, A hierarchy of sets, Part I, Algebra i Logika 7 (1968) 47–73 (Russian) (Algebra and Logic 7 (1968) 24–43 (English translation)).
12. Y. L. Ershov, A hierarchy of sets, Part II, Algebra i Logika 7 (1968) 15–47 (Russian) (Algebra and Logic 7 (1968) 212–232 (English translation)).
13. A. H. Lachlan, Lower bounds for pairs of recursively enumerable degrees, Proc. London Math. Soc. (3) 16 (1966) 537–569.
14. A. Li and X. Yi, Cupping the recursively enumerable degrees by d.r.e. degrees, Proc. London Math. Soc. (3) 78 (1999) 1–21.
15. R. W. Robinson, Interpolation and embedding in the recursively enumerable degrees, Ann. of Math. (2) 93 (1971), 285–314.
16. G. E. Sacks, On the degrees less than $\mathbf{0'}$, Ann. of Math. 77 (1963) 211–231.
17. G. E. Sacks, The recursively enumerable degrees are dense, Ann. of Math. (2) 80 (1964), 300–312.
18. R. I. Soare, Recursively enumerable sets and degrees (Springer, 1987).
19. C. E. M. Yates, Initial segments of the degrees of unsolvability, Part II: minimal degrees, J. Symbolic Logic, 35 (1970), 243–266.

Geometric Software: Robustness Issues and Model of Computation

André Lieutier

Dassault Systémes Provence,
53, avenue de l'Europe,
13082 Aix-en-Provence, Cedex 2, France
and
Modélisation Géométrique et Approximation,
Institut d'Informatique et Mathématiques Appliqueées de Grenoble,
B.P. 53, F-38041 Grenoble, Cedex 9, France
andre_lieutier@ds-fr.com

A significant proportion of algorithms for solid modelling and, more generally, algorithms based on geometric representations suffer of robustness problems. In fact, the impacted algorithms reveal to have numerical computations deeply nested with combinatorial computations. Traditionally, the numerical part of the computation is implemented in double precision floating-point numbers and generally leads to small rounding errors.

It is generally possible to control the condition numbers in the geometric computation and, therefore, these rounding errors have no serious consequences on the accuracy of the output. But an arbitrary small numerical error may break the consistency of the combinatorial part of the algorithm, provoking a fatal failure or an erroneous output. Typically, the numerical information produces combinatorial decisions through comparison tests: `if(`$F(x) > 0$`){}`.

In computational geometry, most studied algorithms deal with simple geometric primitives such as lines, circles or spheres and the computations involved in comparison tests, also called predicates, are polynomials of low algebraic degree. The depth of computation is usually low.

For these situations, the exact computation paradigm has been introduced. It consists in computing the predicate exactly, by evaluating the sign of low degree polynomials using for instance extended precision or modular arithmetic. The exact computation paradigm has successfully solved problems of consistency in algorithms that produce only a combinatorial output, such as Delaunay triangulation for meshing algorithms. However, as soon as one has to construct some geometry in the output, one has to produce consistent outputs, which are outputs for which the numerical part of the data is consistent with the combinatorial part.

The point is that these numerical outputs are necessarily rounded, in practice in floating point values. Indeed, the usage of alternate representations, such as extended arithmetic, rational and algebraic numbers is feasible only for small depth of computation. Sooner or later, it has to be rounded, and this numerical

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 297–298, 2005.

rounding has to be consistent with some sort of topological or combinatorial rounding.

The BRep data structure is widely used in computer aided design software for the representation of solids and shapes. These data structures aggregate very general classes of curves and surfaces together with the combinatorial structure of their incidence/boundary relations. Geometric algorithms operating on these data structures have to deal with the complexity of geometric operations on complex curves and surfaces while preserving the integrity of the resulting structure. Therefore, the problem of the inconsistencies in the combinatorial structure arising from numerical inaccuracies, well known for many usual computational geometry algorithms, can only be worse in the context of BRep data structures. Within the software architecture of modern computer aided geometric software, a curve or a surface is potentially any computable function. Therefore, in theory, an exact predicate would be in principle semi-decidable only.

In practice, such exact predicates are absolutely unrealistic. Looking at the workflow of geometric data in the industry, one has to consider a model of computation in which the depth of computation is not a priori bounded and, in this context, some kind of rounding, with a controlled loss of information, is unavoidable. This is why in CAD software, an uncertainty is, explicitly or not, associated to any numerical value, a technique that could be called the "up to epsilon" programming paradigm. However, today, the necessary "up to epsilon" model is more the art of skilled engineers than a well founded model of computation. In practice, algorithms particularly exposed to robustness problems are very expensive. In fact, studying the rounding process leads to look at the continuity of algorithms. One has to define a topology on both input and output spaces for which the function computed by the algorithm is continuous. An efficient way to do this makes use of continuous domain theory and computability in uncountable sets. Today, only a few algorithms have been devised in this theoretical framework. But it has the merit to formalize what is often understood as street wisdom or engineer skill.

Apparently, discontinuous operations in geometric modelling have a realistic Scott continuous formulation in this context and the rounding, that can be numerical rounding, geometrical rounding or topological/combinatorial rounding has a uniform theoretical formulation: one has to lose information, that is the rounded object is lower, in the information order, than the original object. Of course, this loss of information has still to be controlled to be acceptable. One related issue is the notion of persistent homology and homotopy, allowing to capture some topological information from partial geometric information.

The Dimension of a Point: Computability Meets Fractal Geometry*

Jack H. Lutz

Department of Computer Science,
Iowa State University,
Ames, IA 50011, USA

Recent developments in the theory of computing give a canonical way of assigning a dimension to each point of n-dimensional Euclidean space. Computable points have dimension 0, random points have dimension n, and every real number in $[0, n]$ is the dimension of uncountably many points. If X is a reasonably simple subset of n-dimensional Euclidean space (a union of computably closed sets), then the classical Hausdorff dimension of X is just the supremum of the dimensions of the points in X. In this talk I will discuss the meaning of these developments, their implications for both the theory of computing and fractal geometry, and directions for future research.

References

1. K. Falconer. *The Geometry of Fractal Sets*. Cambridge University Press, 1985.
2. K. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. John Wiley & Sons, 1990.
3. J. M. Hitchcock. Correspondence principles for effective dimensions. *Theory of Computing Systems*. To appear. Preliminary version appeared in *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming*, pages 561–571, 2002.
4. J. H. Lutz. Dimension in complexity classes. *SIAM Journal on Computing*, 32:1236–1259, 2003. Preliminary version appeared in *Proceedings of the Fifteenth Annual IEEE Conference on Computational Complexity*, pages 158–169, 2000.
5. J. H. Lutz. The dimensions of individual strings and sequences. *Information and Computation*, 187:49–79, 2003.

* This research was supported in part by National Science Foundation Grant 0344187.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, p. 299, 2005.

Accepting Networks of Splicing Processors

Florin Manea[1], Carlos Martín-Vide[2], and Victor Mitrana[1,2]

[1] Faculty of Mathematics and Computer Science, University of Bucharest, Str. Academiei 14, 70109, Bucharest, Romania
flmanea@funinf.cs.unibuc.ro

[2] Research Group in Mathematical Linguistics, Rovira i Virgili University, Pça. Imperial Tarraco 1, 43005, Tarragona, Spain
{carlos.martin, vmi}@urv.net

Abstract. We present *linear* time solutions to two NP-complete problems, namely SAT and the directed Hamiltonian Path Problem (HPP), based on accepting networks of splicing processors (ANSP) having all resources (size, number of rules and symbols) linearly bounded by the size of the given instance. The underlying structure of these ANSPs does not depend on the number of clauses, in the case of SAT, and the number of edges, in the case of HPP. Furthermore, the running time of the ANSP solving HPP does not depend on the number of edges of the given graph and this network provides *all solutions*, if any, of the given instance of HPP.

1 Introduction

In this paper, we modify the evolutionary processors placed in the nodes of the networks of evolutionary processors (NEP for short) by splicing processors. The origin of networks of evolutionary processors is twofold. In [6] we consider a computing model inspired by the evolution of cell populations, which might model some properties of evolving cell communities at the syntactical level. Cells are represented by words which describe their DNA sequences. Informally, at any moment of time, the evolutionary system is described by a collection of words, where each word represents one cell. Cells belong to species and their community evolves according to mutations and division which are defined by operations on words. Only those cells are accepted as surviving (correct) ones which are represented by a word in a given set of words, called the genotype space of the species. This feature parallels with the natural process of evolution.

On the other hand, a basic architecture for parallel and distributed symbolic processing, related to the Connection Machine [12] as well as the Logic Flow paradigm [7], consists of several processors, each of them being placed in a node of a virtual complete graph, which are able to handle data associated with the respective node. Each node processor acts on the local data in accordance with some predefined rules, and then local data becomes a mobile agent which can navigate in the network following a given protocol. Only such data can be communicated which can pass a filtering process. This filtering process

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 300–309, 2005.

may require to satisfy some conditions imposed by the sending processor, by the receiving processor or by both of them. All the nodes send simultaneously their data and the receiving nodes handle also simultaneously all the arriving messages, according to some strategies, see, e.g., [8, 12].

In [1](further developed in [2, 14, 3]), we modify this concept (considered in [5] from a formal language theory point of view) in the following way inspired from cell biology. Each processor placed in a node is a very simple processor, an evolutionary processor. By an evolutionary processor we mean a processor which is able to perform very simple operations, namely point mutations in a DNA sequence (insertion, deletion or substitution of a pair of nucleotides). More generally, each node may be viewed as a cell having genetic information encoded in DNA sequences which may evolve by local evolutionary events, that is point mutations. Each node is specialized just for one of these evolutionary operations. Furthermore, the data in each node is organized in the form of multisets of words (each word appears in an arbitrarily large number of copies), and all copies are processed in parallel such that all the possible events that can take place do actually take place. Obviously, the computational process described here is not exactly an evolutionary process in the Darwinian sense. But the rewriting operations we have considered might be interpreted as mutations and the filtering process might be viewed as a selection process. Recombination is missing but it was asserted that evolutionary and functional relationships between genes can be captured by taking only local mutations into consideration [16]. Consequently, networks of evolutionary processors might be viewed as bio-inspired computing models.

Here we replace the point mutations associated with each node by the missing operation mentioned above, that of splicing. This computing model is similar to some extent to test tube distributed systems based on splicing introduced in [4] and further explored in [15]. However, there are several differences: first, the model proposed in [4] is a language generating mechanism while ours is an accepting one, we use a single splicing step, that seems to be more realistic, while every splicing step in [4] is actually an infinite process consisting of iterated splicing steps, each splicing step in our model is reflexive and takes part between a so-called auxiliary word (proper to the node) and a word present in the node at some moment, the filters of our model are very simple and based on random context conditions which appear to us more suitable to be implemented. We want to stress from the very beginning that we are not concerned here with a possible biological implementation, though a matter of great importance.

In a series of papers, we present *linear* time solutions to some NP-complete problems using NEPs. Such solutions are presented for the Bounded Post Correspondence Problem in [1], for the "3-colorability problem" in [2] (with simplified networks), and for the Common Algorithmic Problem in [14]. In [13] we present two linear time solutions to two NP-complete problems, namely the 3CNF-SAT and the HPP (Hamiltonian Path Problem), based on accepting networks of evolutionary processors (ANEP) having all resources (size, number of rules and symbols) linearly bounded by the size of the given instance. However, [13] presents

for the first time such solutions based on ANEPs, and more important, by the definition of ANEPs, one can evaluate the descriptional (number of nodes, rules, symbols) and computational (time) complexity of these ANEPs with respect to their input word which is actually the given instance of the problem.

This paper fits the same line of research. First, we define two complexity classes on ANSP similarly to the classical time and space complexity classes defined on the standard computing model of Turing machine. By definition, ANSPs are always deterministic. Second, we present linear time solutions to two NP-complete problems, namely SAT and the directed Hamiltonian Path Problem (HPP), based on accepting networks of splicing processors (ANSP) having all resources (size, number of rules and symbols) linearly bounded by the size of the given instance. The underlying structure of these ANSPs does not depend on the number of clauses, in the case of SAT, and the number of edges, in the case of HPP. Furthermore, the running time of the ANSP solving HPP does not depend on the number of edges of the given graph and this network provides *all solutions*, if any, of the given instance of HPP.

2 Basic Definitions

We start by summarizing the notions used throughout the paper. An *alphabet* is a finite and nonempty set of symbols. The cardinality of a finite set A is written $card(A)$. Any finite sequence of symbols from an alphabet V is called *word* over V. The set of all words over V is denoted by V^* and the empty word is denoted by ε. The length of a word x is denoted by $|x|$ while $alph(x)$ denotes the minimal alphabet W such that $x \in W^*$.

A splicing rule over the alphabet V is a quadruple written in the form $\sigma = [(x, y); (u, v)]$, where x, y, u, v are words over V. Given a splicing rule σ over V as above and a pair of words (w, z) over the same alphabet V we define the action of σ on (w, z) by:

$$\sigma(w,z) = \{t \mid w = \alpha xy\beta, z = \gamma uv\delta \text{ for some words over } V\ \alpha,\beta,\gamma,\delta \text{ and } t = \alpha xv\delta \text{ or } t = \gamma uy\beta\}.$$

This action on pair of words can be naturally extended to pair of languages A, B by

$$\sigma(A,B) = \bigcup_{w\in A, z\in B} \sigma(w,z).$$

Furthermore, if M is a finite set of splicing rules over V, then we set

$$M(A,B) = \bigcup_{\sigma\in M} \sigma(A,B).$$

For two disjoint and nonempty subsets P and F of an alphabet V and a word w over V, we define the predicates

$$\varphi^{(1)}(w; P, F) \equiv P \subseteq alph(w) \quad \wedge\ F \cap alph(w) = \emptyset$$
$$\varphi^{(2)}(w; P, F) \equiv alph(w) \subseteq P$$
$$\varphi^{(3)}(w; P, F) \equiv P \subseteq alph(w) \quad \wedge\ F \not\subseteq alph(w)$$
$$\varphi^{(4)}(w; P, F) \equiv alph(w) \cap P \neq \emptyset\ \wedge\ F \cap alph(w) = \emptyset.$$

The construction of these predicates is based on *random-context conditions* defined by the two sets P (*permitting contexts/symbols*) and F (*forbidding contexts/symbols*). Informally, the first condition requires that all permitting symbols are and no forbidding symbol is present in w, the second one requires that all symbols of w are permitting ones, while the last two conditions are weaker variants of the first one such that some forbidding symbols may appear in w but not all of them, and at least one permitting symbol appears in w, respectively.

For every language $L \subseteq V^*$ and $\beta \in \{(1), (2), (3), (4)\}$, we define:

$$\varphi^{\beta}(L, P, F) = \{w \in L \mid \varphi^{\beta}(w; P, F)\}.$$

A *splicing processor* over V is a 6-tuple (S, A, PI, FI, PO, FO), where:

– S is a finite set of splicing rules over V.

– A is a finite set of *auxiliary words* over V. These auxiliary words are to be used by this node for splicing.

– $PI, FI \subseteq V$ are the *input* permitting/forbidding contexts of the processor, while $PO, FO \subseteq V$ are the *output* permitting/forbidding contexts of the processor (with $PI \cap FI = \emptyset$ and $PO \cap FO = \emptyset$).

We denote the set of splicing processors over V by SP_V.

An *accepting network of splicing processors* (ANSP for short) is a 6-tuple $\Gamma = (V, U, G, \mathcal{N}, \alpha, x_I, x_O)$, where:

- V and U are the input and network alphabet, respectively, $V \subseteq U$.
- $G = (X_G, E_G)$ is an undirected graph with the set of vertices X_G and the set of edges E_G. G is called the *underlying graph* of the network.
- $\mathcal{N} : X_G \longrightarrow SP_U$ is a mapping which associates with each node $x \in X_G$ the splicing processor $\mathcal{N}(x) = (S_x, A_x, PI_x, FI_x, PO_x, FO_x)$.
- $\alpha : X_G \longrightarrow \{(1), (2), (3), (4)\}$ defines the type of the *input/output filters* of a node. More precisely, for every node, $x \in X_G$, the following filters are defined:

$$\text{input filter: } \rho_x(\cdot) = \varphi^{\beta(x)}(\cdot; PI_x, FI_x),$$
$$\text{output filter: } \tau_x(\cdot) = \varphi^{\beta(x)}(\cdot; PO_x, FO_x).$$

  That is, $\rho_x(w)$ (resp. τ_x) indicates whether or not the word w can pass the input (resp. output) filter of x. More generally, $\rho_x(L)$ (resp. $\tau_x(L)$) is the set of words of L that can pass the input (resp. output) filter of x.
- $x_I, x_O \in X_G$ are the *input* and the *output* node of Γ, respectively.

We say that $card(X_G)$ is the size of Γ. If α is a constant function, then the network is said to be *homogeneous*. In the theory of networks some types of

underlying graphs are common, e.g., *rings, stars, grids*, etc. Networks of evolutionary processors with underlying graphs having these special forms have been considered in [1, 2, 14, 3]. We focus here on *complete* ANSPs, i.e., ANSPs having a complete underlying graph denoted by K_n, where n is the number of vertices.

A *configuration* of a ANSP Γ as above is a mapping $C : X_G \longrightarrow 2^{V^*}$ which associates a set of words with every node of the graph. A configuration may be understood as the sets of words which are present in any node at a given moment. Given a word $w \in V^*$, the initial configuration of Γ on w is defined by $C_0^{(w)}(x_I) = w$ and $C_0^{(w)}(x) = \emptyset$ for all $x \in X_G - \{x_I\}$.

A configuration can change either by a *splicing step* or by a *communication step*. When changing by a splicing step, each component $C(x)$ of the configuration C is changed in accordance with the set of splicing rules M_x associated with the node x and the set A_x. Formally, we say that the configuration C' is obtained in *one splicing step* from the configuration C, written as $C \Longrightarrow C'$, iff

$$C'(x) = S_x(A_x, C(x)) \text{ for all } x \in X_G.$$

When changing by a communication step, each node processor $x \in X_G$ sends one copy of each word it has, which is able to pass the output filter of x, to all the node processors connected to x and receives all the words sent by any node processor connected with x providing that they can pass its input filter.

Formally, we say that the configuration C' is obtained in *one communication step* from configuration C, written as $C \vdash C'$, iff

$$C'(x) = (C(x) - \tau_x(C(x))) \ \cup \bigcup_{\{x,y\} \in E_G} (\tau_y(C(y)) \cap \rho_x(C(y))) \text{ for all } x \in X_G.$$

Let Γ be an ANSP, the computation of Γ on the input word $w \in V^*$ is a sequence of configurations $C_0^{(w)}, C_1^{(w)}, C_2^{(w)}, \ldots$, where $C_0^{(w)}$ is the initial configuration of Γ on w, $C_{2i}^{(w)} \Longrightarrow C_{2i+1}^{(w)}$ and $C_{2i+1}^{(w)} \vdash C_{2i+2}^{(w)}$, for all $i \geq 0$. By the previous definitions, each configuration $C_i^{(w)}$ is uniquely determined by the configuration $C_{i-1}^{(w)}$. Otherwise stated, each computation in an ANSP is deterministic. A computation *halts* (and it is said to be *finite*) if one of the following two conditions holds:

(i) There exists a configuration in which the set of words existing in the output node x_O is non-empty. In this case, the computation is said to be an *accepting computation*.

(ii) There exist two consecutive identical configurations.

The *language accepted* by Γ is

$$L_a(\Gamma) = \{w \in V^* \mid \text{ the computation of } \Gamma \text{ on } w \text{ is an accepting one}\}.$$

We say that an ANSP Γ decides the language $L \subseteq V^*$, and write $L(\Gamma) = L$ iff $L_a(\Gamma) = L$ and the computation of Γ on every $x \in V^*$ halts.

The reader is referred to [10, 11] for the classical time and space complexity classes defined on the standard computing model of Turing machine.

In a similar way, we define two computational complexity measures using ANSP as the computing model. To this aim we consider an ANSP Γ with the input alphabet V that halts on every input. The *time complexity* of the finite computation $C_0^{(x)}$, $C_1^{(x)}$, $C_2^{(x)}$, $\dots C_m^{(x)}$ of Γ on $x \in V^*$ is denoted by $Time_\Gamma(x)$ and equals m. The *length complexity* of the above computation is defined by

$$Length_\Gamma(x) = \max\{|w| : w \in C_i^{(x)}(z), 1 \leq i \leq m, z \in X_G\}.$$

The time complexity of Γ is the partial function from $\mathbf{N}$ to $\mathbf{N}$,

$$Time_\Gamma(n) = \max\{Time_\Gamma(x) \mid x \in V^*, |x| = n\}.$$

Analogously, the length complexity of Γ is the partial function from $\mathbf{N}$ to $\mathbf{N}$,

$$Length_\Gamma(n) = \max\{Length_\Gamma(x) \mid x \in V^*, |x| = n\}.$$

It is easy to note that for any ANSP Γ if $Time_\Gamma$ is bounded by a linear polynomial, then $Length_\Gamma$ is also linearly bounded.

3 Solving Problems with ANSPs

We discuss very briefly and informally how ANSPs could be used as problem solvers. A possible correspondence between decision problems and languages can be done via an encoding function which transforms an instance of a given decision problem into a word, see, e.g., [9]. We say that a decision problem is solved in linear time by ANSPs if the following conditions are satisfied:

1. The encoding function can be computed by a deterministic Turing machine in linear time. Therefore each instance of the problem is linearly related to its associated word.
2. For each instance of the problem one can effectively construct an ANSP which decides in linear time the word encoding the given instance. This means that the word is accepted if and only if the solution to the given instance of the problem is "YES". This effective construction is called a linear time solution to the considered problem.

In this section we present linear solutions to two NP-complete problems: SAT (Satisfiability) and HPP (Hamiltonian Path Problem).

Satisfiability is perhaps the best studied NP-complete problem because one arrives at it from a large number of practical problems. It has direct applications in mathematical logic, artificial intelligence, VLSI engineering, computing theory, etc. It can also be met indirectly in the area of constraint satisfaction problems.

An instance of SAT consists of a formula E with n variables and m clauses. More precisely, the formula E is a conjunction (i.e., $\wedge$) of m clauses, with each being the disjunction (i.e., $\vee$) of several different variables or their negations (i.e., $\bar{\cdot}$) from a set of n variables. We naturally assume that each variable or its

negation appear in at least one clause. The problem asks whether or not there exists an assignment of the n boolean variables such that the m clauses are all satisfied.

Theorem 1. SAT *can be solved in linear time by ANSPs. Furthermore, the other resources (size, total number of symbols, number of splicing rules and auxiliary words associated with any node) of the ANSPs solving a given instance of* **SAT** *are linearly bounded by the size of the instance.*

Proof. Let V be the set of variables, $V = \{x_1, x_2, \ldots, x_n\}$ and $\phi = (C_1) \wedge (C_2) \wedge \ldots (C_m)$ be a boolean formula, where the negation of a variable x_i is denoted by $\bar{x}_i$. Each such formula may be viewed as a word over the alphabet $V \cup \bar{V} \cup \{\wedge, \vee, (,)\}$, where $\bar{V} = \{\bar{x} \mid x \in V\}$. We define the alphabet

$$W = \{[x_i = 1], [x_i = 0] \mid 1 \leq i \leq n\} \cup V \cup \bar{V} \cup \{\vee, \wedge, (,), \$, \#, \uparrow\}.$$

We now consider the ANSP $\Gamma = (V, W, K_{2n+2}, \mathcal{N}, \alpha, In, Out)$, where K_{2n+2} is the complete graph with the $2n+2$ nodes In, Out, $(x_i \leftarrow 0)$, $(x_i \leftarrow 1)$, $1 \leq i \leq n$, and the other parameters are defined as follows:

- $\underline{In}$:
  - $S_{In} = \{[(\$[x_n = b], \varepsilon); (\$, ()] \mid b \in \{0,1\}\} \cup \{[(\$[x_i = b], \varepsilon); (\$, [x_{i+1} = c]) \mid b, c \in \{0,1\}, 1 \leq i \leq n-1\}$,
  - $A_{In} = \{\$[x_i = b] \mid b \in \{0,1\}, 1 \leq i \leq n\}$,
  - $PI_{In} = \emptyset$, $FI_{In} = W$, $PO_{In} = \{[x_1 = 0], [x_1 = 1]\}$, $FO_{In} = \emptyset$, $\alpha(In) = (4)$.
- $\underline{(x_i \leftarrow 0), 1 \leq i \leq n}$:
  - $S_{(x_i \leftarrow 0)}$ contains all splicing rules of the form $[(\uparrow, C'\beta\#); ([x_n = b], (C)\beta\#)]$, where

$$(i)\ \ b \in \{0,1\}, C)\beta \text{ is a suffix of } \phi, \text{ with } C \in (V \cup \bar{V} \cup \{\vee\})^+,$$

$$(ii)\ \ C' = \begin{cases} \varepsilon, \text{ if } C \text{ starts with } \bar{x}_i, \\ (\gamma), \text{ if } C = x_i \vee \gamma, \gamma \in (V \cup \bar{V})^+, \\ \uparrow, \text{ if } C = x_i \text{ or } C \text{ starts with } x_j \text{ or } \bar{x}_j \text{ for some } j \neq i \end{cases}$$

  - $A_{(x_i \leftarrow 0)} = \{\uparrow C'\beta\# \mid \ C' \text{ and } \beta \text{ are defined as above }\}$,
  - $PI_{(x_i \leftarrow 0)} = \{[x_i = 0]\}$, $FI_{(x_i \leftarrow 0)} = \{\uparrow\}$, $PO_{(x_i \leftarrow 0)} = \emptyset$, $FO_{(x_i \leftarrow 0)} = \emptyset$, $\alpha((x_i \leftarrow 0)) = (1)$.
- $\underline{(x_i \leftarrow 1), 1 \leq i \leq n}$:
  - $S_{(x_i \leftarrow 1)}$ contains all splicing rules of the form $[(\uparrow, C'\beta\#); ([x_n = b], (C)\beta\#)]$, where

$$(i)\ \ b \in \{0,1\}, C)\beta \text{ is a suffix of } \phi, \text{ with } C \in (V \cup \bar{V} \cup \{\vee\})^+,$$

$$(ii)\ \ C' = \begin{cases} \varepsilon, \text{ if } C \text{ starts with } x_i, \\ (\gamma), \text{ if } C = \bar{x}_i \vee \gamma, \gamma \in (V \cup \bar{V})^+, \\ \uparrow, \text{ if } C = \bar{x}_i \text{ or } C \text{ starts with } x_j \text{ or } \bar{x}_j \text{ for some } j \neq i \end{cases}$$

  - $A_{(x_i \leftarrow 1)} = \{\uparrow C'\beta\# \mid \ C' \text{ and } \beta \text{ are defined as above }\}$,

- $PI_{(x_i \leftarrow 1)} = \{[x_i = 1]\}$, $FI_{(x_i \leftarrow 1)} = \{\uparrow\}$, $PO_{(x_i \leftarrow 1)} = \emptyset$, $FO_{(x_i \leftarrow 1)} = \emptyset$, $\alpha((x_i \leftarrow 1)) = (1)$.

– *Out*:
- $S_{Out} = A_{Out} = PI_{Out} = PO_{Out} = \emptyset$,
- $FI_{Out} = V \cup \bar{V} \cup \{(,)\vee, \wedge, \uparrow\}, FO_{Out} = W$, $\alpha(Out) = (1)$.

Let us outline the working mode of Γ on the input word $w = \$\phi\#$. We can assume that there are no identical clauses in ϕ. In the initial configuration the word w lies in the input node In. In the first $2n-1$ computational steps, out of which n are splicing ones, no word can be communicated since no word can leave the node In. More precisely, after k splicing steps all words

$$\$[x_{n-k+1} = b_k] \dots [x_n = b_1]\phi\#$$

with $b_j \in \{0,1\}$, $1 \leq j \leq k$ are in In. After the first $2n-1$ steps all these words are communicated to all the other nodes. All these words have two parts: a prefix where either 0 or 1 is assigned to each variable, called the value-prefix, and another part consisting of a word representing a suffix of the input formula, called the formula-suffix. All words of this form will be referred to as *correct* words. Every word that contains $[x_k = b_k], k \in \{1, \dots, n\}$, in its value-prefix can enter the node $N(x_k \leftarrow 1)$, if $b_k = 1$, or the node $N(x_k \leftarrow 0)$, otherwise.

Let us suppose it enters $N(x_k \leftarrow 1)$. If the input formula has the form $(x_k \vee F) \wedge G$, then the formula-suffix of the word is replaced by G, which still represents a formula-suffix. If the input formula has the form $(\bar{x}_k \vee F) \wedge G$, then the formula-suffix of the word is replaced by $(F) \vee G$, which is still a formula-suffix. Finally, if the formula-suffix has the form $(\bar{x}_k)G$, then it is replaced by $\uparrow G$ which is not a formula-suffix anymore, so that this is an incorrect word. It is easy to note that all incorrect words are lost as soon as they leave the node where they were produced. The special form of the splicing rules require that the splicing operation can be done between the value-prefix and the formula-suffix of a correct word only. By this reason, any word having a formula-suffix that starts with x_j or $\bar{x}_j$ is transformed into an incorrect word as soon as it enters a node $N(x_k \leftarrow b)$ with $k \neq j$. Now the process is iterated; it is plain that after each such pair of steps (splicing/communication) a correct word is transformed into either a correct word with a shorter formula-suffix or an incorrect word. Therefore, the node Out contains a word after at most $2n + 2|\phi|$, where $|\phi|$ denotes the length of the formula ϕ, if and only if there exists an assignment of the n variables which satisfies the given formula ϕ. If such an assignment does not exist, then the ANSP halts after $2n + 2|\phi| + 1$ steps having the output node Out empty.

We want to stress that also the other resources of Γ are linearly bounded: this is trivial for the size of Γ and the number of symbols while the number of auxiliary words and splicing rules in every node is linearly bounded by the length of the input formula. □

It is worth mentioning the fact that the underlying structure does not change if the number of variables in the given instance remains the same. We also can

say that the network, excepting the input and output nodes, may be viewed as a "program": we choose the filters, the splicing words and rules, we "guess" a value assignment for the variables, and then we compute the formula by replacing the variables with their values, one by one, from left to right. The ANSP presented in the previous proof is not homogeneous but the reader can easily construct a homogeneous ANSP having two extra nodes able to solve any instance of SAT as efficiently as the above one.

The HPP is to decide whether or not a given directed graph has a Hamiltonian path. A Hamiltonian path in a directed graph is a path which contains all vertices exactly once. It is known that the HPP is an NP-complete problem.

Theorem 2. HPP *can be solved by ANSPs in linear time. Furthermore, the other resources (size, total number of symbols, number of splicing rules and auxiliary words associated with any node) of the ANSP solving a given instance of* **HPP** *are linearly bounded by the number of nodes of the given graph.*

Proof. Let us consider a directed graph $\gamma = (V, E)$, with $V = \{x_1, x_2, \ldots, x_n\}$ for which we are looking for a Hamiltonian path starting with x_1. First we define the alphabet $U = V \cup \{\$, \#, \perp\}$ and the the homogeneous ANSP $\Gamma = (V, U, K_{n+1}, \mathcal{N}, \alpha, In, Out)$, where K_{n+1} is the complete graph with the nodes In, $y_2, y_3, \ldots, y_n$, Out, and the other parameters are defined as follows:

- $\underline{In}$:
  - $S_{In} = \{[(\$, x_1\#^{n-1}); (\perp, \#)]\}$,
  - $A_{In} = \{\$x_1\#^{n-1}\}$,
  - $PI_{In} = \emptyset$, $FI_{In} = V$, $PO_{In} = FO_{In} = \emptyset$, $\alpha(In) = (1)$.
- $\underline{y_i}$, $2 \leq i \leq n$:
  - $S_{y_i} = \{[(\$, x_i\#^{p-1}); (x_j, \#^p)] \mid (x_j, x_i) \in E, p \geq 1\} \cup \{[(\$, \$); (x_j, \#^p)] \mid (x_j, x_i) \notin E, p \geq 1\}$.
  - $A_{y_i} = \{\$x_i\#^p \mid p \geq 0\} \cup \{\$\$\}$.
  - $PI_{y_i} = \emptyset, FI_{y_i} = \{x_i, \$\}, PO_{y_i} = FO_{y_i} = \emptyset$, $\alpha(y_i) = (1)$.
- $\underline{Out}$:
  - $S_{Out} = A_{Out} = \emptyset, PI_{Out} = \emptyset, PO_{Out} = \emptyset$,
  - $FI_{Out} = \{\$, \#\}, FO_{Out} = U$, $\alpha(Out) = (1)$.

The above construction needs a very short explanation. One starts with the input word $\perp\#^n$ in the input node In. After a splicing step, two words go out from In: $\$\#^n$ and $\perp x_1\#^{n-1}$. The first one is lost while a copy of the second one enters each node y_i, $2 \leq i \leq n$. Actually, it is easy to note that any word containing the symbol \$ produced by splicing in any node is lost in the communication step. Let us follow a copy of $\perp x_1\#^{n-1}$ that entered y_i for some i. The next splicing step produces a word that can continue the computational process, namely $\perp x_1 x_i\#^{n-2}$, if and only if $(x_1, x_i) \in E$. The words obtained in this way contain, between $\perp$ and the first occurrence of #, paths in G. Note that the number of occurrences of # in the suffix of these words stores the number of nodes which are still needed for completing a Hamiltonian path. Furthermore, a word containing a symbol x_j, for some j, cannot enter again the node

y_j. By these explanations, after exactly $2n$ steps *Out* contains all Hamiltonian paths, if any, or the computation halts after at most $2n-2$, if the graph has no Hamiltonian path.

It is obvious that also the other resources of Γ are linearly bounded by the number of nodes of the given graph. It is worth mentioning the fact that the underlying structure of Γ does not change if the number of nodes in the given graph remains the same. □

References

1. J. Castellanos, C. Martin-Vide, V. Mitrana, J. Sempere, Solving NP-complete problems with networks of evolutionary processors, *IWANN 2001* (J. Mira, A. Prieto, eds.), LNCS 2084, Springer-Verlag, 2001, 621–628.
2. J. Castellanos, C. Martin-Vide, V. Mitrana, J. Sempere, Networks of evolutionary processors, *Acta Informatica* 39 (2003), 517-529.
3. J. Castellanos, P. Leupold, V. Mitrana, Descriptional and computational complexity aspects of hybrid networks of evolutionary processors, *Theoretical Computer Science*, in press.
4. E. Csuhaj-Varjú, L. Kari, G. Păun, Test tube distributed systems based on splicing, *Computers and AI* 15, 2-3(1996), 211–232.
5. E. Csuhaj-Varjú, A. Salomaa, Networks of parallel language processors. In: *New Trends in Formal Languages* (Gh. Păun, A. Salomaa, eds.), LNCS 1218, Springer Verlag, 1997, 299–318.
6. E. Csuhaj-Varjú, V. Mitrana, Evolutionary systems: a language generating device inspired by evolving communities of cells, *Acta Informatica* 36 (2000), 913–926.
7. L. Errico, C. Jesshope, Towards a new architecture for symbolic processing. In *Artificial Intelligence and Information-Control Systems of Robots '94* (I. Plander, ed.), World Sci. Publ., Singapore, 1994, 31–40.
8. S. E. Fahlman, G. E. Hinton, T. J. Seijnowski, Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. In *Proc. AAAI National Conf. on AI*, William Kaufman, Los Altos, 1983, 109–113.
9. M. Garey, D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-completeness*, Freeman, San Francisco, CA, 1979.
10. J. Hartmanis, P.M. Lewis II, R.E. Stearns, Hierarchies of memory limited computations. *Proc. 6th Annual IEEE Symp. on Switching Circuit Theory and Logical Design*, 1965, 179 - 190.
11. J. Hartmanis, R.E. Stearns, On the computational complexity of algorithms, *Trans. Amer. Math. Soc.* 117 (1965), 533–546.
12. W. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, 1985.
13. F. Manea, C. Martín-Vide, V. Mitrana, Solving 3CNF-SAT and HPP in Linear Time Using WWW, *Proceedings of MCU 2004*, in press.
14. C. Martin-Vide, V. Mitrana, M. Perez-Jimenez, F. Sancho-Caparrini, Hybrid networks of evolutionary processors, *Proc. of GECCO 2003*, LNCS 2723, Springer Verlag, Berlin, 401 - 412.
15. G. Păun, Distributed architectures in DNA computing based on splicing: Limiting the size of components, *Unconventional Models of Computation*, Springer Verlag, Berlin 1998, 323–335.
16. D. Sankoff et al. Gene order comparisons for phylogenetic inference:Evolution of the mitochondrial genome. *Proc. Natl. Acad. Sci. USA*, 89 (1992) 6575–6579.

Hilbert's Tenth Problem and Paradigms of Computation

Yuri Matiyasevich

Steklov Institute of Mathematics
Laboratory of Mathematical Logic
27, Fontanka
St.Petersburg, Russia 191023
yumat@pdmi.ras.ru
http://logic.pdmi.ras.ru/~yumat

Abstract. This is a survey of a century long history of interplay between Hilbert's tenth problem (about solvability of Diophantine equations) and different notions and ideas from the Computability Theory.

1 Statement of the Problem: Intuitive Notion of Algorithm

In the year 1900 the prominent German mathematician David Hilbert delivered to the *Second International Congress of Mathematicians* (held in Paris) his famous lecture titled *Mathematische Probleme* [12]. There he put forth 23 (groups of) problems which were, in his opinion, the most important open problems in mathematics that the pending 20th century would inherit from passing 19th century. Problem number 10 was stated as follows:

> **10. Entscheidung der Lösbarkeit einer diophantischen Gleichung.**
>
> Eine diophantische Gleichung mit irgendwelchen Unbekannten und mit ganzen rationalen Zahlkoefficienten sei vorgelegt : *man soll ein Verfahren angeben, nach welchem sich mittels einer endlichen Anzahl von Operationen entscheiden läßt, ob die Gleichung in ganzen rationalen Zahlen lösbar ist.* [1]

A *Diophantine equation* is an equation of the form

$$P(x_1, \ldots, x_m) = 0 \tag{1}$$

[1] **10. Determination of the Solvability of a Diophantine Equation.** Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: *Devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.*

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 310–321, 2005.

where P is a polynomial with integer coefficients. Hilbert raised the question about solving Diophantine equations in *"rational integers"* which were nothing else but numbers $0, \pm 1, \pm 2, \ldots$; without loss of generality in this paper we will deal with solving Diophantine equations in natural numbers so all lower-case Latin letters will range over $0, 1, 2, \ldots$

Since Diophantus's time (3rd century A.D.) number-theorists have found solutions for plenty of Diophantine equations and also have proved the unsolvability of a large number of other equations. However, for different classes of equations, or even for different individual equations, one had to invent different specific methods. In the 10th problem Hilbert asked for a *universal* method for recognizing the solvability of Diophantine equations, i.e., in modern terminology the 10th problem is a *decision problem* (the only one among the 23 problems).

Note that Hilbert did not use the word "algorithm" in his statement of the tenth problem. Instead, he used the rather vague wording "*a process according to which it can be determined by a finite number of operations ...*". Although he could have used the word "algorithm," it would not really have helped much to clarify the statement of the problem because, at that time, there was no rigorous definition of the general notion of an algorithm. What existed was a number of examples of particular mathematical algorithms (such as celebrated Euclid's algorithm for finding the greatest common divisor of two integers), and an intuitive conception of an algorithm in general.

Does it imply that Hilbert's tenth problem was ill-posed? Not at all. The absence of a general definition of an algorithm was not in itself an obstacle to finding a positive solution of Hilbert's tenth problem. If somebody invented the required "*process*", it should be clear that in fact this process does the job, so an intuitive conception of an algorithm would be sufficient for positive solution of the tenth problem which was, most likely, Hilbert's expectation.

2 Davis's Conjecture: Are All Effectively Enumerable Sets Diophantine?

The first investigations aimed at a proof of algorithmic undecidability of Hilbert's tenth problem appeared at the beginning of 1950's. In particular, at that time Martin Davis considered *Diophantine sets* which are sets of natural numbers having *Diophantine representations*, i.e., definitions of the form

$$a \in \mathfrak{M} \Longleftrightarrow \exists x_1 \ldots x_m [P(a, x_1, \ldots, x_m) = 0] \tag{2}$$

where P is again a polynomial with integer coefficients one of the variables of which, a, is now a *parameter*. Davis's aim was to give a characterization of the whole class of Diophantine sets. The computability theory immediately puts a condition which is necessary for a set to be Diophantine: every Diophantine set is, evidently, effectively enumerable. Davis conjectured ([5, 6]) that this necessary condition is also sufficient:

Davis's conjecture. *A set of natural numbers is Diophantine if and only if it is effectively enumerable.*

Effectively enumerable sets can be defined via the notion of an algorithm, but the things can be taken in the reversed order: having given an independent definition of a effectively enumerable set, one can develop the whole theory of computability in terms of effectively enumerable sets instead of algorithms; examples of such an approach can be found in G.S. Tseitin's paper [38] and P. Martin-Löf's book [25]. Thus Davis's conjecture opened a way to base the computability theory on the number-theoretical notion of a Diophantine set.

3 Davis's Conjecture: First Step to the Proof via Arithmetization

Martin Davis's made the first step to proving his conjecture by showing in [6] that every effectively enumerable set $\mathfrak{M}$ has an almost Diophantine representation:

Theorem (Martin Davis). *Every effectively enumerable set $\mathfrak{M}$ has a representation of the form*

$$a \in \mathfrak{M} \Longleftrightarrow \exists z \forall y_{\leq z} \exists x_1 \ldots x_m [P(a, x_1, \ldots, x_m, y, z) = 0] \tag{3}$$

where P is a polynomial with integer coefficients and $\forall y_{\leq z}$ is the bounded universal quantifier "for all y not greater than z".

A representation of this type became known as the *Davis normal form*. To obtain it, Davis started in [6] with a representation of the set $\mathfrak{M}$ by an arbitrary arithmetical formula with any number of bounded universal quantifiers. The existence of such arithmetical formulas for every effectively enumerable set was demonstrated by Kurt Gödel in his classical paper [10]. Thanks to the bound on the universal quantifiers, every such formula defines an effectively enumerable and hence these formulas could be used for foundation of the Computability Theory.

4 Original Proof of Davis: Post's Normal Forms

According to a footnote in Davis' paper [6], the idea of obtaining the representation (3) by combining universal quantifiers from a general arithmetic representation was due to the (anonymous) referee of the paper. The original proof of Davis (outlined in [5] and given with details in [8]) was quite different. Namely, Davis managed to arithmetize *Post normal forms* using only one universal quantifier. These forms are a special case of more general *canonical forms* introduced by Emil L. Post [36] as a possible foundation of computability theory (and the above cited book [25] uses just Post canonical forms).

5 Davis's Conjecture Proved: Effectively Enumerable Sets Are Diophantine

It took two decades before Davis's conjecture became a theorem (for historical details see, for example, [29]; for an extensive bibliography on Hilbert's tenth problem visit [43]). The following weaker result due to Martin Davis, Hilary Putnam, and Julia Robinson [9] was a mile-stone on the way to the proof of Davis's conjecture:

DPR-Theorem. *For every effectively enumerable set $\mathfrak{M}$ there exists a representation of the form*

$$a \in \mathfrak{M} \Longleftrightarrow \exists x_1 \ldots x_m [E(a, x_1, x_2, \ldots, x_m) = 0] \tag{4}$$

where E is an exponential polynomial*, i.e., an expression constructed by combining the variables and particular integers using the traditional rules of addition, multiplication and exponentiation.*

The last step in the proof of Davis's conjecture was done in [26], and nowadays corresponding theorem is often called

DPRM-Theorem. *The notions of a Diophantine set and the notion of an effectively enumerable set coincide.*

Thus a (seemingly narrow) notion from the Number Theory turned out to be equivalent to the very general notion from the Computability Theory.

6 Existential Arithmetization I: Turing Machines

Already the very first proof of the DPRM-theorem given in [26] was constructive in the sense that as soon as a set $\mathfrak{M}$ is presented in any standard form, it is possible to find corresponding Diophantine representation (2). This was done in the following four steps:

1. construction of an arithmetical formula with many bounded universal quantifiers;
2. transformation of this formula into Davis normal form (3);
3. elimination the single bounded universal quantifier at the cost of passing to exponential Diophantine equations, getting an exponential Diophantine representation (4);
4. elimination of the exponentiation.

Now that we know that in fact no universal quantifier is necessary at all, it would be more natural to try to perform the whole arithmetization by using only purely existential formulas. From technical point of view for the success of this approach it is crucial to select an appropriate device for the initial representation of the set $\mathfrak{M}$.

For the first time such a purely existential arithmetization was done in [28] with the set $\mathfrak{M}$ being recognized by a Turing machine; a simplified way of constructing Diophantine representation by arithmetization of Turing machines is presented in [29]; yet another construction based on Turing machines is given in [39].

7 Existential Arithmetization II: Register Machines

When arithmetizing Turing machine, one has first to introduce a method to represent the content of the tape of the machine by numbers. In this respect another kind of abstract computing devices, *register machines*, turned out to be more suitable as a starting point for constructing Diophantine representations. Register machines were introduced almost simultaneously by several authors: J. Lambek [22], Z. A. Melzak [32], M. L. Minsky [33, 34], and J. C. Shepherdson and H. E. Sturgis [37]. Like Turing machines, register machines have very primitive instructions but, in addition, they deal directly with numbers rather than with words. This led to a "visual proof" of simulation of register machines by Diophantine equations (see [17, 18, 31]).

8 Existential Arithmetization III: Partial Recursive Functions

Another classical tool for the foundations of the Computability Theory are *partial recursive functions*. Existential arithmetization of these functions was done in [30] where Diophantine representations are constructed inductively, alongside construction of a partial recursive function from the initial functions. In order to deal with the primitive recursion and with the operator of minimization it turned out useful to generalize the notion of a partial recursive function: instead of dealing, say, with one-argument function f it was more convenient to work with a function F, defined on arbitrary n-tuples of natural number by

$$F(\langle a_1, \ldots, a_n \rangle) = \langle f(a_1), \ldots, f(a_n) \rangle. \tag{5}$$

9 Universality in Number Theory: Collapse of Diophantine Hierarchy

The DPRM-theorem allows a transfer of a number of ideas from the Computability Theory to the Number Theory. One example of such a transfer is the existence of a *universal Diophantine equation*, i.e., an equation

$$U(a, k, y_1, \ldots, y_M) = 0 \tag{6}$$

with the following property: *for arbitrary Diophantine equation*

$$P(a, x_1, \ldots, x_m) = 0 \tag{7}$$

there exist (effectively calculable) number k_P such that for arbitrary value of the parameter a the equation (7) *has a solution in $x_1, \ldots, x_m$ if and only if equation*

$$U(a, k_P, y_1, \ldots, y_M) = 0 \tag{8}$$

has a solution in $y_1, \ldots, y_M$. This implies that traditional number-theoretical hierarchy of Diophantine equations of degree 1, 2, ... with 1, 2 , ... unknowns collapses at some level. While the existence of (6) immediately follows from DPRM-theorem and the existence of, say, a universal Turing machine, the mere idea of the existence of a such universal object in the theory of Diophantine equations looked quite implausible not only for number-theorists, but for some logicians also (see [21]).

The existence of a universal Diophantine equation is an example of a result which is number-theoretical by its statement, but which was originally proved by tools from Computability Theory; today such an equation (6) can be constructed by purely number-theoretical methods (see [29]).

10 Growth of Solution: Speeding Up Diophantine Equations

Another example of a transfer of ideas from Computability Theory to Number Theory is as follows. M.Davis [7] used the DPRM-theorem in order to get for Diophantine equations an analog of a speed-up theorem of Manuel Blum [3]. Namely, *for every total computable function $\alpha(a, x)$ one can construct two one-parameter Diophantine equations*

$$P_1(a, x_1, \ldots, x_k) = 0, \qquad P_2(a, x_1, \ldots, x_k) = 0 \tag{9}$$

such that

(i) for every value of the parameter a exactly one of these two equations has a solution;

(ii) if Diophantine equations

$$Q_1(a, y_1, \ldots, y_l) = 0, \qquad Q_2(a, y_1, \ldots, y_l) = 0 \tag{10}$$

are solvable for the same values of the parameter a as, respectively, equations (9)*, then one can construct a third pair of Diophantine equations*

$$R_1(a, z_1, \ldots, z_m) = 0, \qquad R_2(a, z_1, \ldots, z_m) = 0 \tag{11}$$

such that

- *these equations are again solvable for the same values of the parameter a as, respectively, equations* (9)*;*
- *for all sufficiently large values of the parameter a for every solution $y_1, \ldots, y_l$ of one of the equations* (10) *there exists a solution $z_1, \ldots, z_m$ of the corresponding equation* (11) *such that*

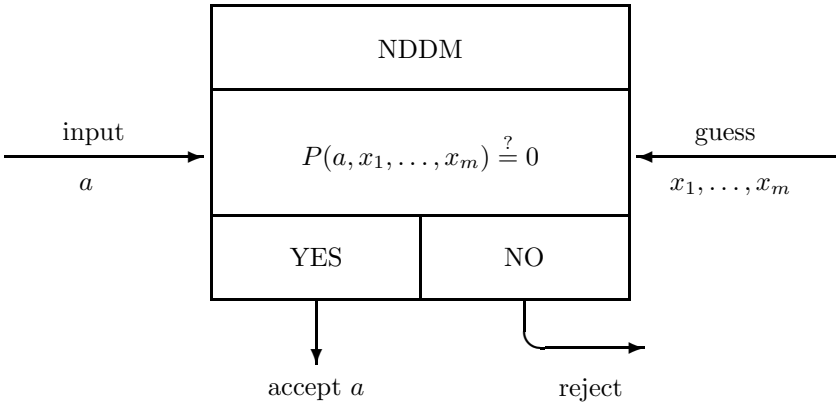
$$y_1 + \cdots + y_l > \alpha(a, z_1 + \cdots + z_m). \tag{12}$$

These formulation of a Diophantine speed-up contains, for the sake of the most generality, the notion of a total computable function; by substituting for α any particular (fast growing) total computable function, one would obtain a purely number-theoretic result which, however, has never been imagined by number-theorist.

11 Diophantine Machines: Capturing Nondeterminism

The DPRM-theorem allows one to treat Diophantine equations as computing devices. This was done in a picturesque form by Leonard Adleman and Kenneth Manders in [2]. Namely, they introduced the notion of *Non-Deterministic Diophantine Machine*, NDDM for short.

A NDDM is specified by a parametric Diophantine equation (7) and works as follows: on input a it guesses the numbers $x_1, \ldots, x_m$ and then checks (7); if the equality holds, then a is accepted.



The DPRM-theorem is exactly the statement that NDDMs are as powerful as, say, Turing machines, i.e., every set acceptable by a Turing machine is accepted by some NDDM, and, of course, *vice versa*.

The idea behind the introduction of a new computing device was as follows: in NDDM we have full separation of guessing and deterministic computation, and the latter is very simple—just the calculation of the value of a polynomial.

12 Unambiguity: Equations with Unique Solution

NDDMs are essentially non-deterministic computing devices. For such devices non-determinism is sometimes fictitious in the sense that at most one path can lead to accepting; if this is so one speaks about *unambiguous computations*. Corresponding property for (exponential) Diophantine representations was called

single-foldness: a representation (2) or (4) is called *single-fold representation* if for given value of the parameter a there exists at most one choice of the unknowns $x_1, \ldots, x_m$.

The existence of single-fold exponential Diophantine representations for every effectively enumerable set was established in [27] and later was improved to the existence single-fold exponential Diophantine representations with only 3 existential variables (see [14, 29]).

The existence of single-fold (or even weaker *finite-fold*) Diophantine representations is a major open problem; the positive answer would shed light on some difficulties met in Number Theory in connection with effectivisation of some results about Diophantine equations (for more details see, for example, [27, 29]).

Single-fold exponential Diophantine representations found applications in the descriptional complexity (see below).

13 Diophantine Complexity: D Versus NP

While the DPRM-theorem implies that NDDMs are as powerful as any other abstract computational device, the intriguing crucial question remains open: *how efficient are the NDDMs?* Adleman and Manders supposed that in fact NDDMs are as efficient as Turing machines.

For the latter there are two natural complexity measures: TIME and SPACE. For NDDMs there is only one natural complexity measure which plays the role of both TIME and SPACE. This measure is SIZE, which is the size (in bits) of the smallest solution of the equation (it is not essential whether we define this solution as the one with the smallest possible value of $\max\{x_1, \ldots, x_m\}$, or of $x_1 + \cdots + x_m$).

Adleman and Manders obtained in [2] the first results comparing the efficiency of NDDMs and Turing machines by estimating the SIZE of a NDDM simulating a Turing machine with TIME in special ranges.

Imposing bounds on the SIZE, we can define a corresponding complexity class. It was shown by A.K. Vinogradov and N.K. Kossovskii [40] that in this way one can define all Grzegorczyk classes starting from $\mathcal{E}^3$. Of course, the lower classes are of greater interest, and, what is typical, they turned out to be more difficult.

Adleman and Manders [1] also introduced the class **D** consisting of all sets $\mathfrak{M}$ having representations of the form

$$a \in \mathfrak{M} \iff \exists x_1 \ldots x_m \big[P(a, x_1, \ldots, x_m) = 0 \,\&\, |x_1| + \cdots + |x_m| \le |a|^k\big]$$

where $|a|$ denotes, as usual, the (binary) length of a. It is easy to see that $\mathbf{D} \subseteq \mathbf{NP}$ and the class **D** is known (see [24]) to contain **NP**-complete problems but otherwise the class **D** is little understood. Adleman and Manders asked whether in fact $\mathbf{D} = \mathbf{NP}$. Recently Chris Pollett [35] showed that this is so provided that $\mathbf{D} \subseteq$ **co-NLOGTIME**, and indicated a number of other ways to tackle $\mathbf{D} = \mathbf{NP}$ question.

An arithmetical definitions of the class **NP** via bounded analog of Davis normal form (3) were given by Bernhard R. Hodgson and Clement F. Kent [19, 13] and by Stasis Yukna [41, 42].

Helger Lipmaa [23] introduced **PD**, the "deterministic part" of the class **D**, and used Diophantine equations for secure information exchange protocols.

14 Random Diophantine Equations: Complexity on Average

The class **NP** contains thousands of equivalent problems which are supposed to be difficult (unless $\mathbf{P} = \mathbf{NP}$). However, only few problems from **NP** were proved to be of the maximal difficulty *on average*. Ramarathanam Venkatesan and Sivaramakrishnan Rajagopalan considered the *Randomized Diophantine Problem* and proved that it is average-case complete; unfortunately, their proof is conditional, and their assumption (on existence of a Diophantine equation with a special property) is equivalent to $\mathbf{D} = \mathbf{NP}$.

15 Parallel Computations: Calculation of a Polynomial on a Petri Net

Petri nets and *systems of vector addition* were introduced as tools for describing parallel computations. Michael Rabin used the undecidability of (exponential) Diophantine equation to prove that some relations between systems of vector addition (and hence also between Petri nets, because the latter easily simulate systems of vector addition) are not recognizable (see paper of Michel Hack [11] where a stronger result was obtained, or [29–Section 10.2]). The crucial point was a definition (introduced by Rabin) of a calculation of the values of (exponential) polynomials by systems of vector addition.

16 A Step Above Hilbert's Tenth Problem: Computational Chaos in Number Theory

Diophantine equations are undecidable. However, every Diophantine set is effectively enumerable and hence its *descriptive complexity* is the least possible: for every polynomial P the initial segment of the set $\mathfrak{M}$ from (2), i.e., the intersection of the set $\mathfrak{M}$ with the set

$$\{a \mid a \le N\}, \tag{13}$$

can be coded by $O(\log(N))$ bits only. However, we can reach the maximal descriptive complexity by considering questions which are only slightly more complicated than those from Hilbert's tenth problem. Gregory Chaitin [4] constructed a one-parameter exponential Diophantine equation such that the set

$$\{a \mid \exists^{\infty} x_1 \ldots x_m [E(a, x_1, x_2, \ldots, x_m) = 0]\} \tag{14}$$

requires N bits (up to an additive constant) for *prefix-free coding* of its intersection with the set (13); here $\exists^{\infty}$ means the existence of infinitely many solutions of the equation. Informally, one can say that the set (14) is completely chaotic.

More recently Toby Ord and Tien D. Kieu [20] constructed another exponential Diophantine equation which for every value of a has only finitely many solutions but the parity of the number of solutions again has completely chaotic behavior in the sense of the descriptive complexity. I was able to generalize this result in the following way: instead of asking about the parity of the number of solutions one can ask whether the number of solutions belongs to any fixed decidable infinite set with infinite complement.

All these results were obtained for exponential Diophantine equations because they are based on the existence of single-fold exponential Diophantine representations; the existence of similar chaos among genuine Diophantine equations is a major open question.

References

1. L. Adleman, K. Manders. Computational complexity of decision procedures for polynomials. In: *16th Annual Symposium on Foundations of Computer Science*, pages 169–177, 1975.
2. L. Adleman, K. Manders. Diophantine complexity. In: *17th Annual Symposium on Foundations of Computer Science*, pages 81–88, Houston, Texas, 25-26 October 1976. IEEE.
3. M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14(2):322–336, 1967.
4. G. Chaitin. *Algorithmic Information Theory.* Cambridge, England: Cambridge University Press (1987).
5. M. Davis, Arithmetical problems and recursively enumerable predicates (abstract). *Journal of Symbolic Logic*, 15(1):77–78, 1950.
6. M. Davis. Arithmetical problems and recursively enumerable predicates. *J. Symbolic Logic*, 18(1):33–41, 1953.
7. M. Davis. Speed-up theorems and Diophantine equations. In Randall Rustin, editor, *Courant Computer Science Symposium 7: Computational Complexity*, pages 87–95, 1973. Algorithmics Press, New York.
8. M. Davis. *Computability and Unsolvability.* Dover Publications, New York, 1982.
9. M. Davis, H. Putnam and J. Robinson, The decision problem for exponential Diophantine equations, *Ann. Math.* (2), 74, 425-436, 1961 (Reprinted in *The collected works of Julia Robinson*, S. Feferman, Ed., *Collected Works*, 6, 1996, xliv+338pp. American Mathematical Society, Providence, RI. ISBN: 0-8218-0575-4).
10. K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. I. *Monatsh. Math. und Phys.*, 38(1):173–198, 1931.
11. M. Hack. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2(1), 77–95, 1976.

12. D. Hilbert, Mathematische Probleme. Vortrag, gehalten auf dem internationalen Mathematiker Kongress zu Paris 1900, *Nachr. K. Ges. Wiss., Göttingen, Math.-Phys.Kl.* (1900), 253-297. See also David Hilbert, *Gesammelte Abhandlungen*, Berlin : Springer, vol. 3 (1935), 310 (Reprinted: New York : Chelsea (1965)). English translation: *Bull. Amer. Math. Soc.*, 8 (1901-1902), 437-479. Reprinted in : *Mathematical Developments arising from Hilbert problems*, Proceedings of symposia in pure mathematics, vol.28, American Mathematical Society, Browder Ed., 1976, pp.1-34.
13. B. R. Hodgson and C. F. Kent. A normal form for arithmetical representation of NP-sets. *Journal of Computer and System Sciences*, 27(3):378–388, 1983.
14. J. P. Jones and Ju. V. Matijasevič. Exponential Diophantine representation of recursively enumerable sets. In J. Stern, editor, *Proceedings of the Herbrand Symposium: Logic Colloquium '81*, volume 107 of *Studies in Logic and the Foundations of Mathematics*, pages 159–177, Amsterdam. North Holland, 1982.
15. J. P. Jones and Ju. V. Matijasevič. A new representation for the symmetric binomial coefficient and its applications. *Les Annales des Sciences Mathématiques du Québec*, 6(1):81–97, 1982.
16. J. P. Jones and Yu. V. Matijasevich. Direct translation of register machines into exponential Diophantine equations. In L. Priese, editor, *Report on the 1st GTI-workshop*, number 13, pages 117–130, Reihe Theoretische Informatik, Universität-Gesamthochschule Paderborn, 1983.
17. J. P. Jones and Y. V. Matijasevič. Register machine proof of the theorem on exponential Diophantine representation of enumerable sets. *J. Symbolic Logic*, 49(3):818–829, 1984.
18. J. P. Jones, Y. V. Matijasevič. Proof of recursive unsolvability of Hilbert's tenth problem. *Amer. Math. Monthly* 98(8):689–709, 1991.
19. C. F. Kent and B. R. Hodgsont. An arithmetical characterization of NP. *Theoretical Computer Science*, 21(3), 255–267, 1982.
20. T. Ord and T. D. Kieu. On the existence of a new family of Diophantine equations for Ω. Fundam. Inform. 56, No.3, 273-284 (2003).
21. G. Kreisel, "Davis, Martin; Putnam, Hilary; Robinson, Julia. The decision problem for exponential Diophantine equations." *Mathematical Reviews*, 24#A3061:573, 1962.
22. J. Lambek. How to program an infinite abacus. *Canad. Math. Bull.*, 4:295–302, 1961.
23. H. Lipmaa. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. Lecture Notes in Computer Science, v. 2894, 2003, 398–415.
24. K. L. Manders and L. Adleman. NP-complete decision problems for binary quadratics. *J. Comput. System Sci.*, 16(2):168–184, 1978.
25. P. Martin-Löf. *Notes on Constructive Mathematics*. Almqvist & Wikseil, Stockholm, 1970.
26. Yu. V. Matiyasevich. Diofantovost' perechislimykh mnozhestv. *Dokl. AN SSSR*, 191(2):278–282, 1970. Translated in: *Soviet Math. Doklady*, 11(2):354-358, 1970.
27. Yu. V. Matiyasevich. Sushchestvovanie neèffektiviziruemykh otsenok v teorii èksponentsial'no diofantovykh uravneniĭ. *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR (LOMI)*, 40:77–93, 1974. (Translated in: *Journal of Soviet Mathematics*, 8(3):299–311, 1977.)

28. Yu. Matiyasevich. Novoe dokazatel'stvo teoremy ob èksponentsial'no diofantovom predstavlenii perechislimykh predikatov. *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR (LOMI)*, 60:75–92, 1976. (Translated in: *Journal of Soviet Mathematics*, 14(5):1475–1486, 1980.)
29. Yu. Matiyasevich. *Desyataya Problema Gilberta.* Moscow, Fizmatlit, 1993. English translation: Hilbert's tenth problem. MIT Press, 1993. French translation: Le dixième problème de Hilbert, Masson, 1995. URL: `http://logic.pdmi.ras.ru/~yumat/H10Pbook`, mirrored at `http://www.informatik.uni-stuttgart.de/ifi/ti/personen/Matiyasevich/H10Pbook`.
30. Yu. Matiysevich. A direct method for simulating partial recursive functions by Diophantine equations. Annals Pure Appl. Logic, 67, 325–348, 1994.
31. Yu. Matiyasevich, *Hilbert's tenth problem: what was done and what is to be done.* Contemporary mathematics, 270, pp. 1–47, 2000.
32. Z. A. Melzak. An informal arithmetical approach to computability and computation. *Canad. Math. Bull.*, 4:279–294, 1961.
33. M. L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in the theory of Turing machines. *Ann. of Math. (2)*, 74:437–455, 1961.
34. M. L. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, Englewood Cliffs; New York, 1967.
35. C. Pollett. On the Bounded Version of Hilbert's Tenth Problem. Arch. Math. Logic, vol. 42. No. 5. 2003. pp. 469–488.
36. E. L. Post. Formal reductions of the general combinatorial decision problem. Amer. J. Math., v. 65 (1943), 197-215. Reprinted in "The collected works of E. L. Post", M. Davis, editor. Birkhäuser, Boston, 1994.
37. J. C. Shepherdson and H. E. Sturgis. Computability of recursive functions, *J. ACM* 10(2):217–255, 1963.
38. G.S.Tseitin. Odin sposob izlozheniya teorii algorifmov i perechislimykh mnozhestv. Trudy Matematicheskogo instituta im. V. A. Steklova 72 (1964) 69–99. English translation in "Proceedings of the Steklov Institute of Mathematics".
39. P. van Emde Boas. Dominos are forever. In L. Priese, editor, *Report on the 1st GTI-workshop*, number 13, pages 75–95, Reihe Theoretische Informatik, Universität-Gesamthochschule Paderborn, 1983.
40. A. K. Vinogradov and N. K. Kosovskiĭ. Ierarkhiya diofantovykh predstavleniĭ primitivno rekursivnykh predikatov. *Vychislitel'naya tekhnika i voprosy kibernetiki*, no. 12, 99–107. Lenigradskiĭ Gosudarstvennyĭ Universitet, Leningrad 1975.
41. S.Yukna. Arifmeticheskie predstavleniya klassov mashinnoĭ slozhnosti. *Matematicheskaya logika i eë primeneniya*, no. 2:92–107. Institut Matematiki i Kibernetiki Akademii Nauk Litovskoĭ SSR, Vil'nyus, 1982.
42. S.Yukna. Ob arifmetizatsii vychisleniĭ. *Matematicheskaya logika i eë primeneniya*, no. 3:117–125. Institut Matematiki i Kibernetiki Akademii Nauk Litovskoĭ SSR, Vil'nyus, 1983.
43. `http://logic.pdmi.ras.ru/Hilbert10`.

On Some Relations Between Approximation Problems and PCPs over the Real Numbers

Klaus Meer*

Department of Mathematics and Computer Science,
Syddansk Universitet, Campusvej 55, 5230 Odense M, Denmark
meer@imada.sdu.dk

Abstract. In [7] it was recently shown that $\mathrm{NP}_{\mathbb{R}} \subseteq \mathrm{PCP}_{\mathbb{R}}(poly, O(1))$, i.e.the existence of transparent long proofs for $\mathrm{NP}_{\mathbb{R}}$ was established. The latter denotes the class of real number decision problems verifiable in polynomial time as introduced by Blum, Shub and Smale [6].

The present paper is devoted to the question what impact a potential full real number $\mathrm{PCP}_{\mathbb{R}}$ theorem $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$ would have on approximation issues in the BSS model of computation. We study two natural optimization problems in the BSS model. The first, denoted by MAX-QPS, is related to polynomial systems; the other, MAX-q-CAP, deals with algebraic circuits. Our main results combine the PCP framework over $\mathbb{R}$ with approximation issues for these two problems. The first main result characterizes validity of a full $\mathrm{PCP}_{\mathbb{R}}$ theorem by the existence of a certain reduction from MAX-QPS to MAX-q-CAP. The second result proves non-existence of particular approximation algorithms if we assume $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$ to hold.

1 Introduction

One of the certainly most important achievements in Theoretical Computer Science within the last decade is the PCP theorem [2, 3]. It characterizes NP by the class $\mathrm{PCP}(O(\log n), O(1))$ of problems accepted by a verifier that generates a logarithmic number of random bits and then checks a constant number of proof components. The PCP theorem has shown tremendous impact on (non-)approximability results in combinatorial optimization. For a short introduction see the next section, full details can be found in [4].

A branch of complexity theory that has seen increasing interest in recent years is real number complexity. A machine model, called Blum-Shub-Smale (for short: BSS) machine, was introduced together with a real analogue $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NP}_{\mathbb{R}}$? of the classical P versus NP question in [6]; see also [5] for a detailed presentation. Considering the PCP theorem and its importance in classical complexity theory

* Partially supported by the EU Network of Excellence PASCAL Pattern Analysis, Statistical Modelling and Computational Learning and by the Danish Natural Science Research Council SNF. This publication only reflects the author's views.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 322–331, 2005.

on the one hand side and real number computations on the other hand it is a natural question to ask whether some kind of PCP theorems hold as well in the BSS model, for example in relation with the class $\mathrm{NP}_{\mathbb{R}}$. In [7], see also [8], the investigation of such questions was started by showing the existence of transparent long proofs for $\mathrm{NP}_{\mathbb{R}}$; in precise terms, the inclusion $\mathrm{NP}_{\mathbb{R}} \subseteq \mathrm{PCP}_{\mathbb{R}}(poly, O(1))$ was proven. The question whether a "full" $\mathrm{PCP}_{\mathbb{R}}$ theorem is true remains a challenging open problem:

$$\mathbf{PCP}_{\mathbb{R}}\text{-Conjecture: } \mathrm{NP}_{\mathbb{R}} \;=\; \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1)).$$

Though this at the moment only is a conjecture we could ask what impact its potential truth would have on approximation problems over the reals. In this paper we shall study such consequences. We investigate the following optimization problems over the reals:

Definition 1. *a) The MAX-Quadratic Polynomial Systems MAX-QPS optimization problem is defined as follows: The input consists of two numbers $n, m \in \mathbb{N}$ together with m polynomials $p_1, \ldots, p_m \in \mathbb{R}[x_1, \ldots, x_n]$, all p_i of degree of most two and depending on at most three variables among $x_1, \ldots, x_n$. The task is to compute the maximal number of polynomials among the p_i's that simultaneously can be made zero by an assignment $y \in \mathbb{R}^n$.*

The corresponding decision problem, i.e. deciding whether there is a common zero for all p_i, is denoted by QPS.

b) The MAX-q-Circuit Acceptance Problem MAX-q-CAP is defined as follows. Let $q \in \mathbb{N}$ be fixed. Input for the problem are natural numbers n, m with $q \leq n$ together with m algebraic circuits $C_1, \ldots, C_m$, (see [5] for more details on algebraic circuits). Each C_i has q input nodes that are labelled with indices $i_1, \ldots, i_q \in \{1, \ldots, n\}$. All circuits compute a result in $\{0, 1\}$. Moreover, the circuits have additional special constant input nodes through which real circuit constants are introduced into the circuit's computation.

The optimization problem is to find the maximal number of circuits that commonly are satisfiable using an input vector $y \in \mathbb{R}^n$. Here, C_i is satisfiable by y iff C_i computes the result '0' in case it takes the value y_{i_j} as input value for input node $i_j, 1 \leq j \leq q$.

Remark 2. Both the MAX-QPS and the MAX-q-CAP problem for $q \geq 3$ are $\mathrm{NP}_{\mathbb{R}}$-hard. For the former this follows directly from $\mathrm{NP}_{\mathbb{R}}$-completeness of the related decision problem. For the latter use the fact that the evaluation of polynomials can be represented by the computation of an algebraic circuit of polynomial size in the size of the given polynomial and having the same number of inputs as the polynomial has variables. The transformation between BSS computations and algebraic circuits is explained in detail in Chapter 18 of [5].

The MAX-QPS problem is a natural real analogue of the MAX-3-SAT problem in combinatorial optimization that asks for maximizing the number of clauses in a 3-SAT formula that can be commonly satisfied. The classical PCP theorem implies that there is no PTAS (polynomial time approximation scheme)

for MAX-3-SAT unless P = NP. We show that for MAX-QPS a similar though weaker statement holds if the above conjecture is true: There is no $\mathrm{FPTAS}_{\mathbb{R}}$ (fully polynomial time approximation scheme) for MAX-QPS unless $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$. With respect to MAX-$q$-CAP more can be shown. We prove that validity of the $\mathrm{PCP}_{\mathbb{R}}$-conjecture is **equivalent** to the existence of a certain reduction from the QPS decision problem to MAX-q-CAP realizing a gap (as in the classical setting with respect to 3-SAT and MAX-3-SAT, see [1]). This result implies that no $\mathrm{PTAS}_{\mathbb{R}}$ for MAX-q-CAP exists unless $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$ if the full real $\mathrm{PCP}_{\mathbb{R}}$ theorem is true.

2 Basic Notions

In this section we very briefly define the notion of real verifiers and $\mathrm{PCP}_{\mathbb{R}}$ classes. We expect the reader to be familiar with the BSS model and the classical PCP theorem. Full details of all that can be found in [4, 5, 7].

Definition 3. *a) Let $r, q : \mathbb{N} \mapsto \mathbb{N}$ be two functions. A $(r(n), q(n))$-restricted verifier V in the Turing model is a particular randomized Turing machine working as follows. For an input $x \in \{0,1\}^*$ of size n and another vector $y \in \{0,1\}^*$ representing a potential membership proof of x in a certain language, the verifier first produces a sequence of $r(n)$ many random bits (under the uniform distribution on $\{0,1\}^{r(n)}$). Given x and these $r(n)$ many random bits V computes in deterministic polynomial time in n the indices of $q(n)$ many components of y. Finally, V uses the input x together with the values of the chosen components of y in order to perform a deterministic polynomial time algorithm. At the end of this algorithm V either accepts or rejects (x, y).*

b) Part a) can be adapted almost word by word in order to define verifiers for the BSS model. The randomized part will be a real number algorithm that tosses coins. The input x and the verification y now belong to the set $\mathbb{R}^* := \bigcup_{n=1}^{\infty} \mathbb{R}^n$. The bit-length of $x \in \mathbb{R}^n$ is replaced by its algebraic size $size_{\mathbb{R}}(x) := n$.

We denote by $V(x, y, \rho)$ the result of V supposed the random sequence generated for input (x, y) was $\rho \in \{0,1\}^{r(size_{\mathbb{R}}(x))}$.

Definition 4. *Let $r, q : \mathbb{N} \mapsto \mathbb{N}$; a real number decision problem $L \subseteq \mathbb{R}^*$ is in the real number complexity class $PCP_{\mathbb{R}}(r(n), q(n))$ iff there exists a $(r(n), q(n))$-restricted verifier V such that conditions i) and ii) below hold:*

i) For all $x \in L$ there is a $y \in \mathbb{R}^*$ such that for all randomly generated strings $\rho \in \{0,1\}^{r(size_{\mathbb{R}}(x))}$ the verifier accepts:

$$\Pr_{\rho}\{V(x, y, \rho) = \ 'accept'\} = 1.$$

ii) For any $x \notin L$ and for each $y \in \mathbb{R}^*$ it is

$$\Pr_{\rho}\{V(x, y, \rho) = \ 'reject'\} \geq \frac{1}{2}.$$

The probability is chosen uniformly over all strings $\rho \in \{0,1\}^{r(size_{\mathbb{R}}(x))}$.

The PCP framework is closely related to approximation issues. Thus, we have to introduce the following notions of approximability for certain optimization problems in the real number setting. These definitions are motivated by the corresponding notions of the classes APX, PTAS and FPTAS used in combinatorial optimization, see [4]. However, we mention that such a generalization is not completely analogous due to the real number framework, see Remark 6 below.

Definition 5. *Let $f : \mathbb{R}^* \times \mathbb{R}^* \mapsto \mathbb{R}^*$ and consider the following maximization problem: Given an instance $\mathcal{I} \in \mathbb{R}^*$ find*

$$\max_{x \in \mathbb{R}^*} f(\mathcal{I}, x) =: OPT(\mathcal{I}) \ .$$

a) The maximization problem belongs to class $APX_{\mathbb{R}}$ (it has an approximation scheme) over $\mathbb{R}$ if and only if there is a constant $r \geq 1$ together with a BSS algorithm $\mathcal{A}$ running in polynomial time in $size_{\mathbb{R}}(\mathcal{I})$ such that for each problem instance $\mathcal{I} \in \mathbb{R}^*$ algorithm $\mathcal{A}$ computes a value $\mathcal{A}(\mathcal{I})$ satisfying

$$\frac{OPT(\mathcal{I})}{\mathcal{A}(\mathcal{I})} \leq r$$

and there exists an $x \in \mathbb{R}^*$ with $f(\mathcal{I}, x) = \mathcal{A}(\mathcal{I})$ (i.e. $\mathcal{A}(\mathcal{I})$ occurs as a function value of f).

b) The maximization problem belongs to class $PTAS_{\mathbb{R}}$ (it has a polynomial time approximation scheme) over $\mathbb{R}$ if there is a BSS algorithm $\mathcal{A}$ that for all problem instances $\mathcal{I} \in \mathbb{R}^*, \epsilon > 0$ computes a value $\mathcal{A}(\mathcal{I}, \epsilon)$ in polynomial time in $size_{\mathbb{R}}(\mathcal{I})$ satisfying

$$\frac{OPT(\mathcal{I})}{\mathcal{A}(\mathcal{I})} \leq 1 + \epsilon \ .$$

Moreover, we require that there exists an $x \in \mathbb{R}^*$ with $f(\mathcal{I}, x) = \mathcal{A}(\mathcal{I})$.

c) The maximization problem belongs to class $FPTAS_{\mathbb{R}}$ (it has a fully polynomial time approximation scheme) over $\mathbb{R}$ if there is a BSS algorithm $\mathcal{A}$ as in part b), but this time the running time of $\mathcal{A}$ depends polynomially both on $size_{\mathbb{R}}(\mathcal{I})$ and on $\frac{1}{\epsilon}$.

Remark 6. We do not intend to present here a more detailed study of approximation classes in the BSS model. Instead, we only consider approximation properties of some specific maximization problems related to $\mathrm{PCP}_{\mathbb{R}}$ classes. This is, for example, the reason why above we are not more specific about the kind of maximization (or minimization) problems to study (as it is done by defining the class NPO of combinatorial optimization problems). In a more detailed development of such an approach first the class $\mathrm{NPO}_{\mathbb{R}}$ would have to be defined carefully. Here, we just want to point out that there are some differences with respect to the classical definitions of these classes in combinatorial optimization. Whereas

in the latter to each problem instance $\mathcal{I}$ there is attached a finite (though exponential in cardinality) set of feasible solutions, for our problems the search space might be uncountable. As a consequence we do not require our approximation algorithms to produce a feasible solution, but only a value guaranteed to come from a feasible solution. In general, the construction of a feasible solution might even be impossible.

3 Results

In this section we analyse relations between a full $\mathrm{PCP}_{\mathbb{R}}$ theorem over the reals and approximation issues to obtain the main results of this paper.

3.1 The MAX-3-QPS Problem

Starting point for our considerations is the following well known result in the Turing model.

Theorem 7. *([1]) The following two statements are equivalent:*

a) NP= PCP$(O(\log n), O(1))$;
b) There exist an $\epsilon > 0$ *and a polynomial time reduction* Φ *from 3-SAT to MAX-3-SAT such that for each instance* $\mathcal{I}$ *for 3-SAT we have*
  i) if $\mathcal{I}$ *is satisfiable, then all clauses in* $\Phi(\mathcal{I})$ *are commonly satisfiable;*
  ii) if $\mathcal{I}$ *is not satisfiable, then at most a fraction of* $\frac{1}{1+\epsilon}$ *many clauses in* $\Phi(\mathcal{I})$ *are commonly satisfiable.*

Whereas the implication b) $\Rightarrow$ a) is easy the reverse one needs the PCP theorem in order to construct a gap between $\Phi(\mathcal{I})$ for satisfiable and for unsatisfiable $\mathcal{I}$'s. In particular, the above theorem implies that there is no PTAS for MAX-3-SAT unless P = NP.

We now want to study whether a similar statement holds in the BSS model over $\mathbb{R}$. So far, the existence of long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$ is known: $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(poly, O(1))$ (see [7]). It is an open problem whether also a full $\mathrm{PCP}_{\mathbb{R}}$ theorem is true: Is $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$?

Therefore it is interesting to analyze whether one can characterize such a result as well by means of a reduction property similar to the one given in Theorem 7. Moreover, what implications would result from a full $\mathrm{PCP}_{\mathbb{R}}$ theorem with respect to certain approximation properties over the reals?

A natural problem to start with is the MAX-QPS problem from Definition 1. It was used in [7] to establish the existence of transparent long proofs for $\mathrm{NP}_{\mathbb{R}}$.

We shall see that for the MAX-QPS problem proving a similar statement to Theorem 7 is not obvious. The following result (in particular its proof) indicates what might go wrong.

Theorem 8. *Suppose the real* $PCP_{\mathbb{R}}$*-conjecture holds, i.e. the equation* $NP_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), O(1))$ *is true. Then there exists no fully polynomial time approximation scheme* ($FPTAS_{\mathbb{R}}$) *for MAX-QPS in the BSS model unless* $P_{\mathbb{R}} = NP_{\mathbb{R}}$.

Proof. The proof at the beginning follows the one in [2] showing that MAX-3-SAT does not have a PTAS unless P = NP. However, major differences occur due to the semi-algebraic framework our problem is situated in.

Let L be an arbitrary, fixed $\mathrm{NP}_{\mathbb{R}}$-complete decision problem and let $x \in \mathbb{R}^n$ be an input for deciding whether $x \in L$. Consider a $(O(\log n), O(1))$-restricted verifier V for problem L. Such a verifier V is existing according to the assumption that the $\mathrm{PCP}_{\mathbb{R}}$-conjecture is true. If $c \cdot \log n$ ($c > 0$ constant) many random bits are produced and q components are inspected, then we introduce $q \cdot n^c$ many real variables $z_1, \ldots, z_{q \cdot n^c}$. An assignment for $z_1, \ldots, z_{q \cdot n^c}$ represents a potential membership proof for $x \in L$. For a fixed choice $r \in \{0,1\}^{c \cdot \log n}$ for the random bits the verifier chooses q among the components of $z_1, \ldots, z_{q \cdot n^c}$. Let us denote the values of the chosen components by $z_1(r), \ldots, z_q(r)$. Given these real values together with the input x the verifier determines in deterministic polynomial time whether it accepts $(x, z_1(r), \ldots, z_q(r))$. Let $A(x) \subseteq \mathbb{R}^q$ denote the semi-algebraic set of values $(z_1(r), \ldots, z_q(r))$ such that the verifier accepts $(x, z_1(r), \ldots, z_q(r))$ if and only if the components $z_1(r), \ldots, z_q(r)$ belong to $A(x)$.

Next, we apply the reduction procedure in [6] to construct a QPS instance $\mathcal{P}(r)$ of polynomially many polynomial equations:

$$\begin{aligned} \mathcal{P}(r) \equiv \quad & p_1^r(x, z_1(r), \ldots, z_q(r), t_1^r, \ldots, t_k^r) = 0 \quad \wedge \ldots \wedge \\ & \wedge \ \ p_\ell^r(x, z_1(r), \ldots, z_q(r), t_1^r, \ldots, t_k^r) = 0. \end{aligned}$$

The system $\mathcal{P}(r)$ has the following properties:

- all p_i^r are polynomials of degree at most 2;
- all p_i^r depend on at most three variables among the $z_1(r), \ldots, z_q(r)$, $t_1^r, \ldots, t_k^r$;
- both the numbers ℓ of equations and k of newly introduced variables during the reduction are polynomially bounded in n, say both are at most $p(n)$ for a polynomial p;
- for fixed x the system has a solution with respect to $z_1(r), \ldots, z_q(r)$, $t_1^r, \ldots, t_k^r$ iff the verifier accepts $(x, z_1(r), \ldots, z_q(r))$ iff $(z_1(r), \ldots, z_q(r)) \in A$.

In particular, if V does not accept $(x, z_1(r), \ldots, z_q(r))$, then at most $p(n) - 1$ of the polynomials in $\mathcal{P}(r)$ do have a common root (without loss of generality we can assume that $\mathcal{P}(r)$ precisely has $p(n)$ many equations). Note as well that $\mathcal{P}(r)$ is a QPS instance.

Next, we construct another QPS instance $\mathcal{P}$ by building all conjunctions of the $\mathcal{P}(r)$'s:

$$\mathcal{P} := \bigwedge_{r \in \{0,1\}^{c \cdot \log n}} \mathcal{P}(r)$$

$\mathcal{P}$ has $p(n) \cdot n^c$ many equations. Moreover, if an instance x belongs to L all equations have a solution because the verifier accepts the correct proof for all random strings. Contrary, if $x \notin L$ the verifier rejects each proof for at least half of the random strings. This means that at most

$$p(n) \cdot n^c - \frac{n^c}{2} \;=\; n^c \cdot (p(n) - \frac{1}{2})$$

many equations in $\mathcal{P}$ can be solved simultaneously.

Finally, if there were a $\mathrm{FPTAS}_{\mathbb{R}}\mathcal{A}$ for MAX-QPS we could solve the $\mathrm{NP}_{\mathbb{R}}$-complete problem L in polynomial time as follows. For input x construct in polynomial time the above instance $\mathcal{P}$ as well as an $\epsilon \leq \frac{1}{2 \cdot p(n)}$. Compute an ϵ-approximation of the MAX-QPS instance $\mathcal{P}$ using $\mathcal{A}$ in polynomial time in $n := size_{\mathbb{R}}(x)$ and $(\epsilon)^{-1}$, i.e. in polynomial time in n. Easy calculation shows that for the above choice of ϵ algorithm $\mathcal{A}$ will yield a result $> n^c \cdot (p(n) - 1)$ if and only if the original instance x belongs to L. □

Let us comment on the above proof. The problem when comparing it with Theorem 7 is the following. It shows that a real version of the full PCP theorem implies the existence of a polynomial p together with a polynomial time BSS reduction from the QPS decision problem to the MAX-QPS maximization problem such that

i) if $\mathcal{P}$ is an instance for QPS that has a solution (i.e. the involved polynomials have a common zero), then the same is true for all the polynomials in $\Phi(\mathcal{P})$;
ii) if $\mathcal{P}$ is an instance with no solution, then at most a fraction of $1 - \frac{1}{2 \cdot p(n)}$ many polynomials of $\Phi(\mathcal{P})$ have a common zero.

The point here is that our proof cannot generate a constant gap independent of n. This is due to the fact that the reduction above uses the polynomial time computation of the verifier to produce a semi-algebraic description of the set A of suitable values for the q components of a potential proof y. Contrary to the situation in the corresponding proof for MAX-3-SAT in the Turing model, where there is a constant description of A independently of the (polynomial) running time of the verifier simply because A is finite of cardinality at most 2^q, this seems to be not the case in the real framework. There is no obvious way to obtain a constant length description of A only depending on q.

This fact also 'destroys' equivalence in the above statement. If we suppose the reduction Φ to exist, then the natural way to construct a verifier out of it is the following. For a set of polynomials representing an instance for MAX-QPS and for a guess $y \in \mathbb{R}^*$, pick one polynomial at random and evaluate it in y. Repeat this procedure as many times as it is necessary to detect a fault in the verification y with probability at least $\frac{1}{2}$. The problem is that now this needs polynomially many attempts, so we do not obtain a verifier that reads a constant number of components in y, only.

This leads us to the following

Problem 9. Assuming $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$, is there a polynomial time BSS reduction from QPS to MAX-QPS that realizes a constant gap in the above sense?

It is even unclear whether MAX-QPS admits approximation algorithms with any fixed error guarantee:

Problem 10. Does MAX-QPS belong to $\mathrm{APX}_{\mathbb{R}}$, i.e. is there a constant $r > 1$ and a polynomial time (in the algebraic size of an input) BSS algorithm that computes for an instance $\mathcal{P}$ of QPS the maximal number of commonly feasible polynomials up to factor r?

Whereas for MAX-3-SAT such an algorithm is trivially designed the question for MAX-QPS seems to be much more involved. If we allow the m polynomials involved to have altogether $O(m^2)$ terms instead of at most $O(m)$ (as is the case with instances for QPS), then the following negative result can be obtained. Note that in the QPS problem we can assume $m \geq \frac{n}{3}$ since each variable occurs at least once.

Theorem 11. *Consider the following approximation problem over the reals: Given a system of polynomials $p_1, \ldots, p_m \in \mathbb{R}[x_1, \ldots, x_n]$ in n variables such that*

- *each p_i has degree 2 and*
- *the total number of non-vanishing terms in all p_i is bounded by $O(m^2)$.*

Then there is no polynomial time BSS algorithm approximating the value $\max_{x \in \mathbb{R}^n} |\{i | p_i(x) = 0\}|$ within a constant factor $r > 1$ unless $P_{\mathbb{R}} = NP_{\mathbb{R}}$.

Proof. Suppose such an approximation algorithm exists. We then show how to decide the $\mathrm{NP}_{\mathbb{R}}$-complete QPS decision problem in polynomial time. Let $\mathcal{Q} = (Q_1, \ldots, Q_s)$ be an input system for QPS such that each Q_i has degree 2 and depends on at most 3 of the variables $x_1, \ldots, x_n$. Clearly, the question to decide whether $\max_{x \in \mathbb{R}^n} |\{i | Q_i(x) = 0\}| = s$ is $\mathrm{NP}_{\mathbb{R}}$-complete as well. Consider a polynomial time approximation algorithm $\mathcal{A}$ that approximates the above maximum within a factor $r > 1$, i.e. $\mathcal{A}$ computes a result $\mathcal{A}(\mathcal{Q}) \in \mathbb{N}$ with

$$\frac{\max_{x \in \mathbb{R}^n} |\{i | Q_i(x) = 0\}|}{\mathcal{A}(\mathcal{Q})} \leq r \ .$$

Thus, if all s polynomials Q_i have a common zero, then $\mathcal{A}$ produces a result $\geq \frac{s}{r}$ on input $\mathcal{Q}$. We construct a new polynomial system with the same set of variables and $r \cdot s$ many polynomials. Choose a matrix $A \in \mathbb{R}^{r \cdot s \times s}$ such that each (s, s) submatrix of A is regular. For example, A can be taken as a Vandermonde matrix with $r \cdot s$ rows with entries $(i, i^2, \ldots, i^s), i \in \{1, \ldots, r \cdot s\}$. The new system consists of the $r \cdot s$ polynomials obtained when multiplying $A \cdot (Q_1(x), \ldots, Q_s(x))^T$; it has $\leq 3 \cdot s$ many terms per polynomial, thus $O(r \cdot s^2)$ altogether. It then follows:

i) If the original polynomial system has a solution x^*, then the system $A \cdot (Q_1(x), \ldots, Q_s(x))^T = 0$ as well has the solution x^*. Consequently, all $r \cdot s$ many polynomials of that system have a common solution and the (exact) result of the maximization problem is $r \cdot s$.

ii) If the original system has no common solution, then among the polynomials in $A \cdot (Q_1(x), \ldots, Q_s(x))^T$ at most $s - 1$ can have a common zero. This is true because regularity of all (s, s) submatrices of A implies that for each such

submatrix $\tilde{A}$ the system $\tilde{A} \cdot y = 0$ only has the trivial solution $y = 0 \in \mathbb{R}^n$. But this solution cannot be obtained as $y = (Q_1(x), \ldots, Q_s(x))^T$ for an $x \in \mathbb{R}^n$ because of the assumption. In this case, the maximum value of the optimization problem therefore is $\leq s - 1$.

Algorithm $\mathcal{A}$ in case i) has to give a result $\geq \frac{r \cdot s}{r} = s$, in case ii) the result will be $\leq s - 1$. We can decide solvability of the original polynomial system $\mathcal{Q}$ in polynomial time by applying $\mathcal{A}$ to the new instance $A \cdot (Q_1(x), \ldots, Q_s(x))^T$ and deciding, whether the result is $\geq s$ or not. □

Remark 12. The above proof can be used to obtain a slightly sharper result. The proof works as long as we introduce a new system that has a polynomial in s number of polynomials. Thus, the matrix A constructed can have $p(s)$ many rows instead of only $r \cdot s$ many, where p is an arbitrary polynomial. The corresponding calculations then will yield a slight improvement on the previous result.

3.2 The MAX-q-CAP Problem

The previous subsection leads to the question whether there are other approximation problems over the reals than MAX-QPS that do satisfy a statement like the MAX-3-SAT problem in Theorem 7. The problem in our proof for MAX-QPS was the reduction used. If we are able to define a problem for which the similar reduction only results in a constant number of introduced 'similar' objects, then we might hope for a stronger result. The MAX-q-CAP problem from Definition 1 has this property.

Theorem 13. *The following two statements are equivalent:*

a) $NP_{\mathbb{R}} = PCP_{\mathbb{R}}(O(\log n), O(1))$;
b) There exist an $\epsilon > 0$ *and a polynomial time BSS reduction* Φ *from the QPS decision problem to the MAX-3-CAP optimization problem such that for each instance* $\mathcal{P}$ *for QPS we have*
  i) if the polynomials in $\mathcal{P}$ *are commonly satisfiable, then all circuits in* $\Phi(\mathcal{P})$ *are commonly satisfiable;*
  ii) if not all polynomials in $\mathcal{P}$ *are commonly satisfiable, then at most a fraction of* $\frac{1}{1+\epsilon}$ *many circuits in* $\Phi(\mathcal{P})$ *are commonly satisfiable.*

Proof. For the implication $a) \Rightarrow b)$ we consider once again the proof of Theorem 8. Given an input x for QPS and the verifier we define the set $A \subseteq \mathbb{R}^q$ as above. Now instead of applying the reduction to QPS we just perform the transformation of the verifier's polynomial time computation on x and an $r \in \{0,1\}^{c \cdot \log n}$ as constants and a $z \in \mathbb{R}^q$ as input to one algebraic circuit of polynomial size, see [5]. This circuit has q many input nodes and gives result 0 on z iff the verifier accepts (x, z, r). Thus, we can construct an instance of the MAX-q-CAP problem that has n^c many circuits. The main point is that we introduce just one circuit for one choice of r. Following the same arguments as before if the verifier rejects a guess y it rejects it for at least $\frac{1}{2}$ of the random strings, and this implies the

same for the MAX-q-CAP instance we obtain. This yields the existence of the required reduction with $\epsilon := 1$.

The converse follows by producing a MAX-q-CAP instance using the assumed reduction. We pick at random a circuit and check for a guess y, whether the circuit computes result 0. If not we reject. Note that this happens with probability at least $1 - \frac{1}{1+\epsilon}$ if the verification was false. We repeat that procedure until the failure probability is less than $\frac{1}{2}$. This gives the required verifier. □

Corollary 14. *Assume that the real PCP$_{\mathbb{R}}$-conjecture is true and that $P_{\mathbb{R}} \neq NP_{\mathbb{R}}$. Let $q \geq 3$ be fixed. Then there exists* **no** *polynomial time approximation scheme PTAS$_{\mathbb{R}}$ for MAX-q-CAP.*

Proof. Assume that a PTAS$_{\mathbb{R}}$ $\mathcal{A}$ exists. Applying $\mathcal{A}$ to an input $\mathcal{P} = (n, m, p_1, \ldots, p_m)$ of MAX-QPS and an $\epsilon \leq 1$ we could decide solvability of the MAX-QPS decision problem (i.e. whether the p_i have a common zero) in polynomial time as follows: By using Theorem 13, part b) we produce a MAX-q-CAP instance $\Phi(\mathcal{P})$ in polynomial time. Let ℓ be the number of circuits in the instance $\Phi(\mathcal{P})$. Then we apply $\mathcal{A}$ to $(\Phi(\mathcal{P}), 1)$ and check whether the result is less than $\frac{\ell}{2}$ or not. If it is less we reject, otherwise we accept. □

In this paper approximation issues for some natural real number maximization problems were studied. We have seen that there is a closed relation between a potential full PCP$_{\mathbb{R}}$ theorem for NP$_{\mathbb{R}}$ and (non-)approximability properties of such real number problems. This substantiates that working towards proving the conjecture NP$_{\mathbb{R}}$ = PCP$_{\mathbb{R}}(O(\log n), O(1))$ is a major challenge in this area of real number complexity theory.

References

1. S. Arora, C. Lund: Hardness of Approximation. In: Approximation Algorithms for NP-hard problems. D. Hochbaum (ed.), PWS Publishing, Boston, 399–446, 1996.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy: Proof verification and hardness of approximation problems. Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 14–23, 1992.
3. S. Arora, S. Safra: Probabilistic checking proofs: A new characterization of NP. Journal of the ACM 45, 70–122, 1998. Preliminary version: Proc. of the 33rd Annual IEEE Symposium on the Foundations of Computer Science, 2–13, 1992.
4. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, 1999.
5. L. Blum, F. Cucker, M. Shub, S. Smale: Complexity and Real Computation. Springer, 1998.
6. L. Blum, M. Shub, S. Smale: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. Bull. Amer. Math. Soc., vol. 21, 1–46, 1989.
7. K. Meer: Transparent long proofs: A first PCP theorem for NP$_{\mathbb{R}}$. Extended abstract in: Proc. ICALP 2004, Springer LNCS 3142, 959–970, 2004.
8. K. Meer: Probabilistically checkable proofs over the reals. To appear in: Electronic Notes in Theoretical Computer Science.

Correlation Dimension and the Quality of Forecasts Given by a Neural Network

Krzysztof Michalak and Halina Kwasnicka

Faculty Division of Computer Science,
Wroclaw University of Technology, Wroclaw, Poland

Abstract. The problem addressed in this paper is searching for a dependence between the correlation dimension of a time series and the mean square error (MSE) obtained when predicting the future time series values using a multilayer perceptron. The relation between the correlantion dimension and the ability of a neural network to adapt to sample data represented by in-sample mean square error is also studied. The dependence between correlation dimension and in-sample and out-of-sample MSE is found in many real-life as well as artificial time series. The results presented in the paper were obtained using various neural network sizes and various activation functions of the output layer neurons.

1 Introduction

The correlation dimension is one of the measures that can be used to describe the behaviour of a dynamic system. For a countable set of points $x_1, x_2, \ldots \in R^k$ the correlation dimension is usually defined using the correlation integral:

$$C(r) = \lim_{n\to\infty} \frac{1}{n^2} \sum_{i,j=1}^{n} H(r - |x_i - y_j|) \ . \tag{1}$$

where H(z) = 1 for positive z, and 0 otherwise. The correlation dimension D_{corr} is then defined as:

$$D_{\text{corr}} = \lim_{r\to 0} \frac{log(C(r))}{log(r)} \ . \tag{2}$$

Given the time series representing the output of a dynamic system one can compute the correlation dimension of the attractor representing the dynamics of the underlying system [1]. Due to the fact that the estimation of the correlation dimension is relatively easy, this measure is widely used to describe the dynamics of a time series in a quantitative way.

Apart from being a fundamental dynamical invariant correlation dimension is often used for estimation of the complexity of the model of a dynamic system, such as an autoregresive (AR) model. When neural networks are concerned the correlation dimension is often used to estimate the optimal size of the network

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 332–341, 2005.

[2, 3]. In such an approach, especially when implications of an embedding theorem by Takens [4] are concerned only large-scale differences in correlation dimension are considered important.

In this paper some more subtle effects are studied. Time series used in experiments had values of correlation dimension estimate falling into a range of length of about 3. They were fed into a multilayer perceptron with fixed structure and for each time series values of in-sample and out-of-sample mean square error were recorded. It is shown, that both in case of real-life and artificial data a dependence between the correlation dimension and MSE values can be observed.

This paper is structured as follows. Section 2 describes data used and defines the problem. Section 3 describes the experiments performed on real-life data. Experiments performed on artificial data are described in Section 4. Finally, Section 5 concludes the paper.

2 Data Description and Problem Definition

This paper addreses the issue of measuring prediction error made when using a multilayer perceptron for predicting future values of some time series $\{p_t\}$. In order to feed a time series into the neural net it must be preprocessed to form sets of input and output vectors. Usually, a set of time lagged vectors $x_t = (p_{t-d+1}, p_{t-d+2}, \ldots, p_t)$ is used as an input set. The most common approach is to use one-dimensional output vectors, so the output vector corresponding to the input vector x_t is $y_t = (p_{t+1})$.

Considering the definition of input and output data the neural network is required to have $N_{\text{in}} = d$ neurons in the input layer and $N_{\text{out}} = 1$ neuron in the output layer. The number of neurons in the hidden layer N_{hid} can vary arbitrarily.

In order to measure how well a neural network adapts to training data and how good a prediction of future time series values is, the values of in-sample mean square error MSE_{in} and out-of-sample mean square error MSE_{out} respectively are used.

Given a set of time-lagged vectors one can calculate the correlation dimension D_{corr} of the set consisting of consecutive vectors x_d, x_{d+1}, ... in an embedding space R^d. Note, that in general one can calculate the value of D_{corr} using embedding space of other dimensionality than was used to create input data set for the neural net. Nevertheless, in the experiments presented in this paper the dimension d of the embedding space used to calculate D_{corr} was always set to be equal to the number of inputs of the neural network N_{in}. Thus, the same set of time lagged vectors was fed to the network and was used to calculate the correlation dimension. For calulating the correlation dimension a well-known Grassberger-Procaccia algorithm [1] was used.

In order to calculate the MSE_{in} and MSE_{out} values obtained when processing time series with different values of D_{corr} sets of 10 to 50 time series were tested. These time series were chosen so that the behaviour of the series in a given set was expected to be "similar" to each other. Values of MSE_{in} and

MSE_{out} obtained from the time series from a given set were then plotted against values of D_{corr}. To the data points obtained linear or quadratic approximation was applied.

3 Real-Life Data

In order to prepare sets of real-life time series showing similar behaviour average monthly land-surface temperature series were used. Source data was obtained from [5]. These data sets consist of time series of temperatures recorded at some fixed points on Earth surface. Recording stations are spaced evenly every 0.5 degree latitude and longitude. Each time series contains temperature values from years 1930-2000. It was assumed that time series recorded at stations which are placed at close locations behave in a similar way. Especially, sets of measurements from stations placed at the same latitude were considered interesting. Table 1 summarizes the data sets used in the experiments.

Table 1. Summary of real-life data

Set name	Latitude	Longitude	Number of time series
Victoria Island	69.75	100.25 to 119.75	40
Canada	52.25	100.25 to 119.75	40
Azores	40.25	20.25 to 43.25	40
Beaufort Sea	69.75	120.25 to 139.75	40

Each time series $\{p_t\}$ contains $N = 852$ points. Values in each series were normalized to [0, 1]. Tests were performed using many different values for the number of input neurons N_{in}, the number of hidden neurons N_{hid} and various activation functions in the output layer. Some example parameter sets are summarized in Tab. 2.

Table 2. Parameters of neural networks used for prediction

Set name	N_{in}	N_{hid}	Output activation function
Victoria Island	16	100	sigmoid
Victoria Island	16	32	sigmoid
Victoria Island	16	100	linear
Victoria Island	6	20	sigmoid
Canada	16	100	sigmoid
Azores	16	100	sigmoid
Beaufort Sea	16	100	sigmoid

Networks of other sizes using *sigmoid* and *linear* output activation functions produced results very similar to these presented in the paper. Other output

activation functions such as *softmax* and *tanh* were also tested but learning process convergence in case of these functions was very poor.

For each data set the correlation dimension of each individual time series calculated using the embedding space with dimension $d = N_{in}$ falls within the range [1.9, 3.5].

A set of time-lagged vectors $(p_{t-N_{in}+1}, \ldots, p_t)$ for $t = N_{in}, \ldots, N-1$ was constructed from each of the series in the set using the sliding window technique. The length of the window was equal to the number of input neurons N_{in}. Respective desired output for each input vector was p_{t+1}.

The learning process was performed using all but the last 150 vectors of each input set. Initial weights of each perceptron were drawn from a zero-mean unit variance gaussian. Then, weights optimization using a scaled conjugate gradient algorithm was performed. Optimization was stopped when a change of all of the weigths in a single optimization step was smaller than 10^{-15}, but no later than after 1000 iterations.

After the weight optimization was complete all the training input vectors were forwarded through the network so that the MSE_{in} could be measured. Then, the last 150 input vectors which had not been used for network training were forwarded and the MSE_{out} was calculated. For each time series the whole process starting with random weights initialization was performed 10 times and mean values of MSE_{in} andMSE_{out} were recorded. For all tested time series the value of MSE_{in} fell within the range [0.0008, 0.003] and the value of MSE_{out} fell within the range [0.001, 0.008].

Results obtained are presented on Fig. 1-6. For all data sets linear trends were plotted. Note that due to space limitations not all data sets listed in Tab. 2 were illustrated on the figures.

The results illustrated on Fig. 1 and 2 were obtained using relatively large networks. Very similar behaviour was observed for most of experiments performed using neural networks of similar size regardless of the data set used. As it can clearly be seen the linear fit in these cases is almost perfect.

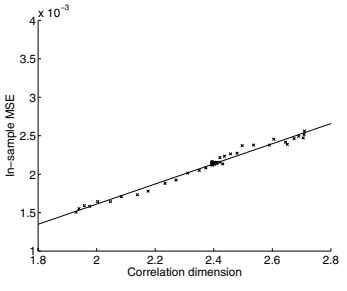


Fig. 1. MSE_{in} for Victoria Island data set (Lat. = 69.75, Lon. $\in$ [100.25, 119.75]) obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 100$ and linear activation function

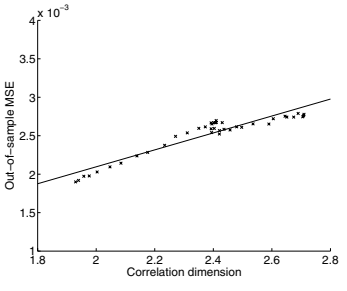


Fig. 2. MSE_{out} for Victoria Island data set (Lat. = 69.75, Lon. $\in$ [100.25, 119.75]) obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 100$ and linear activation function

The correlation dimension of time series in Victoria Island data set ranges from 2.1 to 2.7. From the Takens embedding theorem, it follows, that the underlying dynamical system could in this case be adequately represented by a model with a number of degrees of freedom between 5.2 and 6.4. As an integer is required, $N_{\text{in}} = 6$ was used. In this case a good linear approximaxtion could also be found albeit some outliers appeared as it can be seen on Fig. 3 and 4.

Values of mean square errors for the Azores data set are clearly much more spreaded than for the other sets as it is shown on Fig. 5 and 6. However, in this case also a relatively good linear fit could be found.

It is interesting to plot all the results on one chart. Of course, to obtain a consistent plot, it is required to use the results obtained using a fixed network structure. On Fig. 7 and 8 the results for MSE_{in} and MSE_{out} respectively, obtained for $N_{\text{in}} = 16$ and $N_{\text{hid}} = 100$ for data sets described in Tab. 1 are

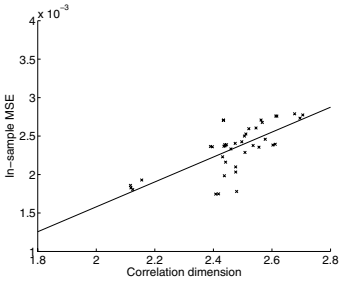


Fig. 3. MSE_{in} for Victoria Island data set (Lat. = 69.75, Lon. $\in$ [100.25, 119.75]) obtained for $N_{\text{in}} = 6$, $N_{\text{hid}} = 20$ and sigmoid activation function

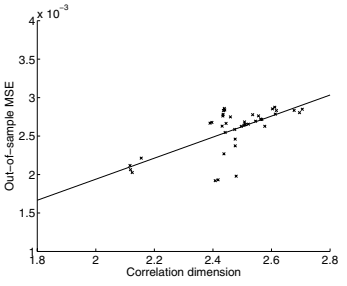


Fig. 4. MSE_{out} for Victoria Island data set (Lat. = 69.75, Lon. $\in$ [100.25, 119.75]) obtained for $N_{\text{in}} = 6$, $N_{\text{hid}} = 20$ and sigmoid activation function

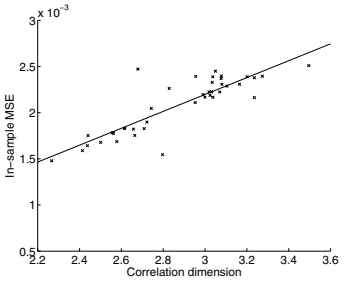


Fig. 5. MSE_{in} for Azores data set (Lat. = 40.25, Lon. $\in$ [20.25, 43.25]) obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 100$ and sigmoid activation function

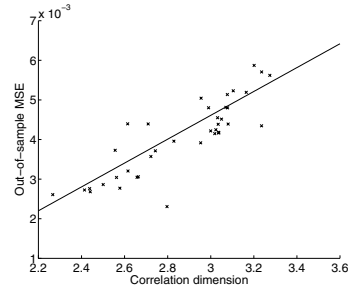


Fig. 6. MSE_{out} for Azores data set (Lat. = 40.25, Lon. $\in$ [20.25, 43.25]) obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 100$ and sigmoid activation function

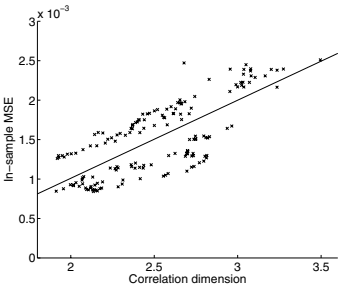


Fig. 7. MSE $_{\text{in}}$for all meteorological data sets obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 100$ and sigmoid activation function

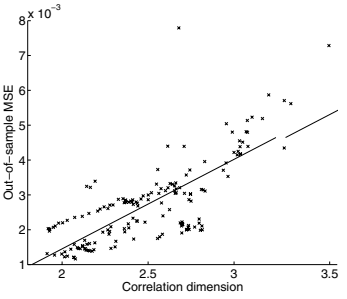


Fig. 8. MSE_{out} for all meteorological data sets obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 100$ and sigmoid activation function

presented. In all cases a sigmoid activation function was used. Observations obtained for all data sets using the same network structure follow the linear trend closely.

Tests performed on real-life data, presented in this section suggest that the ability of a neural network to adapt to training data and the quality of the forecasts given by a network is related to the correlation dimension of the input time series. However, the parameters of the linear approximation of the results are in each case different. It is thus apparent that the characteristics of the relationship may vary and to obtain consistent results the time series in each set must be in some way "similar" to each other. In case of meteorological data this similarity is ensured by using test sets containing temperature time series recorded at neigbouring stations.

4 Artificial Data

In the previous section it was shown that in meteorological data a relationship between the correlation dimension of data and the ability of a neural network to adapt to data and to predict future values can be observed. However, it was also noted that for this relationship to be clearly visible the test time series must be in some way "similar". One of the ways to achieve such similarity is to exploit additional information from the problem domain. In case of meteorological data this additional information is the geographic location at which the temperature series was recorded. Unfortunately, such kind of information does not give any clues of what features of the time series itself are responsible for two time series being "similar" or "dissimilar".

To provide some more insight into this issue tests on artificially generated data were performed. Usually, artificial time series are generated by iterating a model of a dynamical system. By changing parameters of the model a wide

variety of time series can usually be generated. However, due to the chaotic nature of nonlinear dynamical systems commonly used for this task, there is very little or no control over how similar any of these time series are, even if only slight modifications in model parameters are introduced. For example it was observed that if the MSE values are plotted against D_{corr} for the time series generated by a Mackey-Glass model, the results do not form any recognizable structure no matter what sets of parameters are applied to the model. The time series generated by this model for various parameters are too "dissimilar".

Therefore, to generate suitable time series sets other methods have to be employed. In this paper artificial data sets were constructed by transforming existing real-life time series, namely return rates of 5 companies from the Warsaw Stock Exchange. The length of a single time series varies from about 1420 to about 2660 depending how long a company has been present at the stock market. The longest time series cover the period since July 1994 till November 2003. Details of stock price time series used are presented in Tab. 3.

Table 3. Summary of stock price data

Set name	Company name	Number of samples
AMERBANK	AmerBank	2296
DZPOLSKA	DZ Bank Polska	2310
FORTISPL	Fortis Bank Polska	2223
ODLEWNIE	Odlewnie Polskie	1415
TONSIL	Tonsil	2661

Test data sets were created using two methods: by adding random noise and by applying a step function to each base time series. In the random noise approach random values drawn from a zero-mean gaussian were added to each element of the base series. Apart from original series the test set contained series created using gaussian distribution with 0.00001, 0.00002, 0.00005, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2 and 0.5 variance. Thus, each test set contained 16 time series. In the step function approach base time series were transformed using a step function returning 0 and 1 with the threshold value 0.0100, 0.0125, 0.0150, 0.0175, 0.0200, 0.0225, 0.0250, 0.0275, 0.0300, 0.0325, 0.0350, 0.0375, 0.0400, 0.0425, 0.0450, 0.0475 and 0.0500. Base time series containing untransformed return rates was not included in the test set in this approach, so the total number of series in each set was 17.

The tests performed on noised data were identical to those performed on real-life data. The network used was a two-layer perceptron with $N_{\mathrm{in}} = 16$, $N_{\mathrm{hid}} = 50$ and a sigmoid activation function. The results obtained are also very similar to the ones obtained using the real-life data. In case of noised data the points obtained are more spreaded and the ranges for the MSE and D_{corr} are wider. Some of the results are shown on Fig. 9-12. Results for other data sets are very similar in following the linear trend to those visualised on figures, even though values of the correlation dimension and MSE sometimes fall into different ranges.

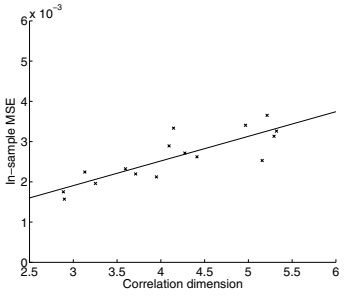


Fig. 9. MSE_{in} for noised AMERBANK data set obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

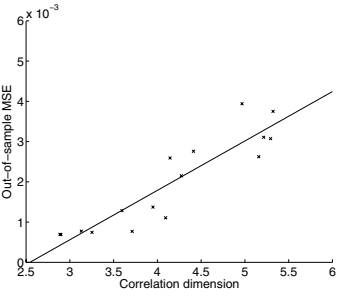


Fig. 10. MSE_{out} for noised AMERBANK data set obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

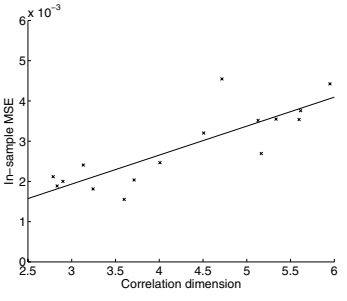


Fig. 11. MSE_{in} for noised DZPOLSKA data set obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

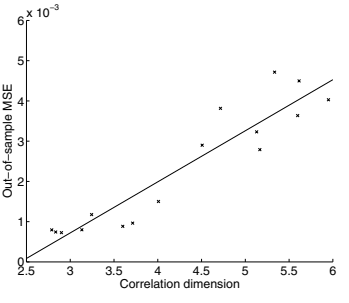


Fig. 12. MSE_{out} for noised DZPOLSKA data set obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

For artificial data created by adding noise to base time series of stock prices a linear dependence between MSE and D_{corr} can be observed. However, the observations are more spreaded than in case of meteorological data. It is worth noticing, that the linear dependence holds regardless of how big the MSE is. Similar characteristics is observed when the MSE is below 0.004 and for MSE reaching 0.05. Due to the fact that for different data sets the values of the MSE fall into different ranges when all recorded results are plotted on one chart no particular structure can be observed.

The other tests were performed on sets of time series created using step function. These tests were identical to those performed on real-life data, with the only difference that in case of step function approach the output of the neural network was passed through the step function with threshold 0.5 before calculating the MSE. The network used in each case was a two-layer perceptron with $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and with sigmoid activation function.

In this case the correlation dimension of the time series was very small, ranging from 0.0025 to 0.05. This was caused by the fact that the step function

preprocessing generates binary vectors in R^d which form the set with dimensionality very close to zero.

On the other hand, values of both in-sample MSE and out-of-sample MSE were relatively high, the MSE_{in} ranging from 0 to 0.1 and MSE_{out} ranging from 0.03 to 0.5. Of course, the values of the MSE cannot be compared to those obtained using other approaches because a step function on the network output allows the error on single sample to be 0 or 1 only.

The most interesting effect that can be observed is that while the MSE_{out} is still linearly dependent of the D_{corr} the values of MSE_{in} plotted against D_{corr} look more like a quadratic function. The results for AMERBANK data set are presented on Fig. 13 and 14. Results for other data sets look very much similar.

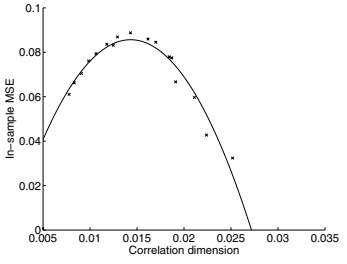


Fig. 13. MSE_{in} for step-function transformed AMERBANK data set obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

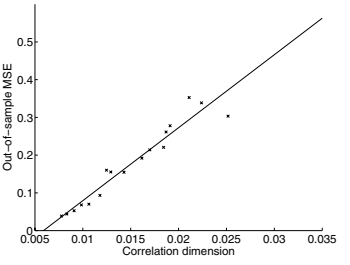


Fig. 14. MSE_{out} for step-function transformed AMERBANK data set obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

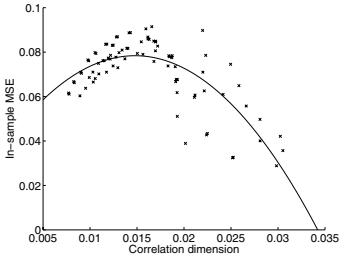


Fig. 15. MSE_{in} for all step-function transformed stock prices data sets obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

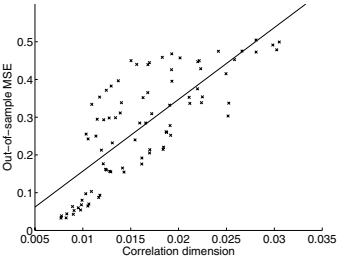


Fig. 16. MSE_{out} for all step-function transformed stock prices data sets obtained for $N_{\text{in}} = 16$, $N_{\text{hid}} = 50$ and sigmoid activation function

For artificial data created by transforming a base time series of stock prices with a step function a quadratic dependence between MSE_{in} and D_{corr} can be observed. Nevertheless, for MSE_{out} the characteristics is still linear. This linear

dependence in case of step-function transformed data is observed for values of D_{corr} from a quite different range than in other cases. As, similarly to the results for meteorological data, the values of MSE for step-function transformed data fall into the same range for all data sets, all of these results can be plotted on one chart. As can be seen on Fig. 15 and 16 the results are somewhat spreaded but they form distinct structures around the proposed approximation curves.

5 Conclusion

The experiments presented in this paper were aimed at finding dependence between the correlation dimension of a time series and the ability of a neural network to predict future values of the time series. Experiments were performed on real-life data as well as some artificially constructed data sets. Neural networks used were of different sizes and various output activation functions were studied. Also, two different methods of pre- and post processing were used.

In all presented cases a clear dependence between the in-sample and out-of-sample MSE and the correlation dimension of the time-series was found. This dependence was observed for various values of the correlation dimension and the MSE. The characteristics of this dependence varies between data sets which suggests that the time series in a single data set must be in some way "similar". Finding proper measures of such similarity requires further study. Also, more theoretical studies are required to explain how the two factors in question are related.

References

1. Grassberger P., Procaccia I.: Characterization of strange attractors. Physical Review Letters, **Volume 50, Issue 5** (1983) 346–349
2. Camastra F., Vinciarelli A.: Intrinsic Dimension Estimation of Data: An Approach Based on Grassberger-Procaccia's Algorithm. Neural Processing Letters, **Volume 14, Issue 1** (2001) 27–34
3. Zhang Sheng et al.: Determining the input dimension of a neural network for nonlinear time series prediction. Chinese Physics, **Volume 12, Issue 6** (2003) 594–598.
4. Takens F.: Detecting Strange Attractors in Turbulence. Proceedings of the Symposion on Dynamical Systems and Turbulence (1983)
5. Willmott, Matsuura and Collaborators' Global Climate Resource Pages. `http://climate.geog.udel.edu/~climate/`

The Computational Complexity of One-Dimensional Sandpiles

Peter Bro Miltersen*

Dept. of Computer Science,
University of Aarhus, Denmark
bromille@brics.dk

Abstract. We prove that the one-dimensional sandpile prediction problem is in $\mathbf{AC}^1$. The previously best known upper bound on the $\mathbf{AC}^i$-scale was $\mathbf{AC}^2$. We also prove that it is not in $\mathbf{AC}^{1-\epsilon}$ for any constant $\epsilon > 0$.

1 Introduction

In physics, a *complex dynamical* system can be informally defined as a system operating on the "*edge of chaos*" (a phrase coined by Langton [7]). Thus, complexity is neither rigid order, with the future development of the system being easily and completely understood, nor chaos, with the system being virtually unpredictable and exhibiting "quasi-random" behavior. For the computer scientist, complex dynamical systems are interesting as they are exactly the physical systems capable of performing non-trivial computation.

In an attempt to put the above considerations on more rigorous ground, Machta, Moore and collaborators have, in a series of papers [5, 8, 9, 12, 11, 13], put forward a general program seeking to capture and measure the physicists' informal notion of the "complexity" of a system by the rigorous notion of the *computational* complexity of the problem of predicting the behavior of the system. This approach gives us the opportunity of getting a fine-grained view of the complexity of a system by using the hierarchy of complexity classes of computational complexity theory. As a simple example, if the prediction problem for a system is complete for $\mathbf{P}$, we can say that the system is capable of efficiently emulating general sequential computation, while, if the prediction problem is in, say, $\mathbf{NL}$, the system can only perform computations obeying a severe space bound and thus not all computations, assuming $\mathbf{P} \neq \mathbf{NL}$.

An important model in complex systems theory is the Bak-Tang-Wiesenfeld *sandpile* model [1, 2]. The prediction problem for this model was considered from the computational complexity point of view by Moore and Nilsson [10]. The problem is defined for any dimension $d \geq 1$, but in this paper we concentrate on the one-dimensional case.

* Peter Bro Miltersen is supported by BRICS, Basic Research in Computer Science, a centre of the Danish National Research Foundation and by a grant from the Danish Research Council. He would like to thank Cris Moore for helpful discussions.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 342–348, 2005.

The one-dimensional sandpile prediction problem can be described as follows (the description of Moore and Nilsson is slightly different but can be easily seen to be equivalent to the following). A one-dimensional sandpile is a map $h : \mathbf{Z} \to \{0, 1, 2, \ldots\}$. The value $h(i)$ is interpreted as the height of a pile of sand at position i. A pile of height 2 or more is *unstable*, and may *topple*, distributing sand evenly to the two neighboring positions. Formally, if map $h' : \mathbf{Z} \to \{0, 1, 2, \ldots\}$ satisfies that for some i, $h'(i) = h(i) - 2, h'(i-1) = h(i-1) + 1, h'(i+1) = h(i+1) + 1$, and $h'(x) = h(x)$ for all $x \notin \{i-1, i, i+1\}$, we'll say that h' is a possible successor to h, or $h \to h'$. If there is no possible successor to h, we'll say that h is *stable*. If h is zero outside some interval I, say, $I = \{1, 2, \ldots, n\}$, it can be shown that there is a unique stable g, so that $h \to^* g$. We will denote this unique g by h^*. The one-dimensional sandpile prediction problem is the following: Given $h : \{1, 2, \ldots, n\} \to \{0, 1, 2\}$, find h^*. A similar definition can be made for the d-dimensional sandpile problem for any $d \geq 1$, but in this paper, we concentrate on the one-dimensional case.

Following their general program, Moore and Nilsson showed that the d-dimensional sandpile prediction problem is in $\mathbf{P}$ for all d, that it is $\mathbf{P}$-complete for $d \geq 3$ and that the 1-dimensional problem is in $\mathbf{AC}^1[\mathbf{NL}] \subseteq \mathbf{AC}^2 \subseteq \mathbf{NC}^3$. They also present an efficient sequential algorithm for the 1-dimensional problem with a time bound of $O(n \log n)$.

The purpose of this paper is to present two pieces of information about the complexity of the one-dimensional sandpile prediction problem, narrowing down the possible range of its exact complexity considerably.

First we show the following improvement of the parallel upper bound of Moore and Nilsson.

Theorem 1. *The one-dimensional sandpile prediction problem is in* $\mathbf{AC}^1 \subseteq \mathbf{NC}^2$.

Conceptually, the algorithm is much simpler than the parallel algorithm of Moore and Nilsson. It is based on refining their *sequential* algorithm. and then noticing that the refined algorithm can be carried out by a deterministic poly-time logspace Turing machine with access to an auxiliary pushdown store. It is known that the class of languages so computed is **LOGDCFL** [15], the class of languages logspace reducible to a deterministic context-free language, and as $\mathbf{LOGDCFL} \subseteq \mathbf{AC}^1$ [14], the result follows.

Second, we prove the following hardness result.

Theorem 2. *The one-dimensional sandpile prediction problem is hard for* $\mathbf{TC}^0$ *with respect to constant depth reductions.*

That the one-dimensional sandpile prediction problem is hard for $\mathbf{TC}^0$ means that the one-dimensional sandpile is sufficiently complex to carry out at least somes slightly non-rudimentary computation, such as computing the parity or majority of n bits. This provides formal justification for the statement of Moore and Nilsson that the dynamics of the one-dimensional sandpile problem is "non-trivial". It also implies the following lower bound:

Corollary 1. *The one-dimensional sandpile prediction problem is* not *in* $\mathbf{AC}^{1-\epsilon}$ *for any constant* $\epsilon > 0$.

As $\mathbf{AC}^i$ is the class of problems solvable by CRCW (concurrent-read, concurrent-write) PRAMs (parallel random access machines) with polynomially many processors in time $O((\log n)^i)$, we thus have fairly tight upper and lower bounds for solving the one-dimensional sandpile problem in this model of computation.

1.1 Organization of the Paper

In Section 2, we sketch the proof of Theorem 1 and in Section 3, we sketch the proofs of Theorem 2 and corollary 3. We conclude with some open problems in Section 4.

2 The Upper Bound

In this section, we sketch the proof of Theorem 1.

In their paper, Moore and Nilsson show that the one-dimensional sandpile problem can be solved by the following sequential algorithm. Given an input sandpile $h : \{1, \ldots, n\} \to \{0, 1, 2\}$, extend it to $\mathbf{Z}$ by making zero the values of h outside the interval from 1 to n. We maintain two subsets T and N of the integers. Initially T is the set $\{i \in \mathbf{Z} | h(i) = 2\}$ and N is the set $\{i \in \mathbf{Z} | h(i) = 0\}$. Note that while N is an infinite set, it has an obvious finite representation. Now, while T is not empty repeat the following two steps, a *round*:

1. Pick an $t \in T$. Find $z_1, z_2 \in N$ so that $z_1 \leq t < z_2$.
2. Let $N = (N - \{z_1, z_2\}) \cup \{z_1 + z_2 - t\}$. Let $T = T - \{t\}$.

When T is empty, h' defined to be 0 on N and 1 on $\mathbf{Z} - N$ is the unique stable successor of h. We refer the reader to the argument for this in Moore and Nilsson.

Moore and Nilsson suggest implementing the above algorithm directly by maintaining the "finite part" of the set N in sorted order, and, for each $t \in T$, do a binary search in N to find z_1 and z_2. Our first observation is that this binary search can be eliminated when going through the list T in increasing order. Indeed, we can maintain the following *invariant*. Between rounds, if $T = \{t_1 < t_2 < \ldots < t_m\}$, we can maintain a partition of N into N_1 and N_2 in such a way that the following properties are true:

1. All elements in N_1 are smaller than each element in N_2.
2. The elements z_1, z_2 so that $z_1 \leq t_1 < z_2$ are either
   (a) The largest element of N_1 and the smallest element of N_2 or
   (b) Both in N_2.

First note that we can easily establish the invariant at the beginning of the computation. Now given the invariant, we want to perform one round of the algorithm of Moore and Nilsson. We should find an appropriate z_1, z_2 so that

$z_1 \leq t_1 < z_2$. If case (a) applies, we remove t_1 from T (so t_2 will be the "new" t_1 in the next round), z_1 from N_1 and z_2 from N_2. Note that $z_1 < z_1 + z_2 - t_1 \leq z_2$ and also note the t_2 is greater than the new largest element of N_1 (since t_1 was). Thus, if we insert $z_1 + z_2 - t_1$ as the new largest element of N_1 if $z_1 + z_2 - t_1 \leq t_2$ and we insert $z_1 + z_2 - t_1$ as the new smallest element of N_2 if $t_2 < z_1 + z_2 - t_1$, we have maintained the invariant. If case (b) applies, we move elements from N_2 to N_1 until case (a) applies, reducing it to this case. This completes the description of the algorithm.

Because of case (b), the reader may conclude that we have replaced binary search with linear search which does not sound like such a grand idea, but note that the size of N_2 is never increased during a round. Thus, the total cost of moving elements from N_2 during the entire execution of the algorithm is linear.

Concerning the complexity of the refined algorithm, viewed as a sequential algorithm we have to be somewhat careful about the exact model of computation. On a log cost RAM, the unrefined algorithm of Moore and Nilsson has complexity $O(n \log n)$ because of

1. The binary searches which cost $O(\log n)$ time each.
2. Adding and subtracting $O(\log n)$-bit integers, each operation costing $O(\log n)$ time.

The refined algorithm also has complexity $O(n \log n)$ in the log-cost model. However, it is common practice to *only* use the log-cost time measure when integers of bit length much bigger than the log of the input are involved (e.g., as in case of the problem of multiplying two n-bit integers). When only $O(\log n)$-bit length integers are involved, the *unit cost* RAM model is much more commonly used. In the unit cost RAM model, the refined algorithm has complexity $O(n)$ while the Moore-Nilsson version keeps having complexity $O(n \log n)$ because of the binary searches.

More important than its sequential complexity in various RAM models is the consequences of the refined algorithm for the parallel complexity of the sandpile prediction problem. Indeed, we next note that the refined algorithm can be implemented by a polynomial time, logspace Turing machine with access to an auxiliary pushdown store (i.e., an extra "free" tape, where a tape cell is erased when the head moves from the cell to its left neighbor). Because of the robustness of logarithmic space, we just have to argue that the algorithm can be implemented by a while-program using $O(1)$ variables, each holding an integer of $O(\log n)$ bits and an auxiliary object STACK where such $O(\log n)$ bit integers can be pushed and popped. We show how each variables in the refined algorithms can be represented using such objects.

N_1 will be represented by the STACK object and a single integer variable v. The invariants of the representation are the following:

1. N_1 is exactly $\{\ldots, -v-2, -v-1, -v\} \cup \{\text{the elements in STACK}\}$.
2. The elements of STACK are sorted, with the largest at the top.

With this representation, we can remove the largest element from N_1, and add an element to N_1 larger than the largest one. These are the only operations we

need to perform on N_1 when running the refined algorithm. We can also easily initialize the representation to the correct content in the beginning of the refined algorithm.

N_2 will be represented by three integer variables l, i and w. The invariant of the representation is that $N_2 = \{l\} \cup \{j \geq i | H[i] = 0\} \cup \{j \in \mathbf{Z} | j \geq w\}$. Here $H[1..n]$ is the structure holding in the input, representing the map $h : \{1, \ldots, n\} \rightarrow \{0, 1, 2\}$. With this representation, we can replace the smallest element of N_2 by another element and remove the smallest element from N_2. These are the only two operations the refined algorithm uses.

T is represented in a way similar to N_2.

We have now shown that the refined algorithm for one-dimensional sandpile prediction can be implemented by a deterministic, polynomial time, logspace Turing machine with access to an auxiliary pushdown store. At the end of the algorithm, the output is contained in the representation of N_1 and N_2. If we are interested in the i'th bit of the output for some i, we can find it by either inspecting the structure for N_2 or the structure for N_1, in the last case popping a sufficient number of elements from the stack. Thus the language 1SANDPILE = $\{\langle h, i\rangle |$ the i'th bit in h^* is $1\}$ can be decided by a deterministic logspace Turing machine with an auxiliary pushdown store. By a result of Sudborough [15], this means that the language is in **LOGDCFL** which, by a result of Ruzzo [14] is a subset of $\mathbf{AC}^1$. The functional version of the problem, i.e. the map $h \rightarrow h^*$ itself is therefore computable in the same class.

3 The Lower Bound

In this section, we sketch the proofs of Theorem 2 and Corollary 3.

Moore and Nilsson showed **P**-completeness of higher-dimensional versions of the sandpile prediction problem; here we show hardness for much lower complexity classes of the one-dimensional problem. When considering **P**-completeness, logspace reductions are most often used. However, these are not meaningful for classes below **L**, as those classes are not closed under logspace reductions. Here, we use a weaker notion of reductions, DLOGTIME-uniform constant depth reductions [4, 3]. All classes considered here are closed under those reductions. A language π_1 reduces to an language to π_2 by such reductions if we can build DLOGTIME-uniform, constant depth, polynomial size circuit for π_1, using unbounded fan-in AND-gates, unbounded fan-in OR-gates, NEGATION-gates, and *oracle*-gates computing π_2.

Let MAJORITY be the problem of deciding whether a string of n input bits (n odd) have more 1's than zeros. We shall show that MAJORITY reduces to 1SANDPILE. Since MAJORITY is complete for $\mathbf{TC}^0$ by constant depth reductions (indeed, the *definition* of $\mathbf{TC}^0$ is that it is the closure of MAJORITY under constant depth reductions), it follows that 1SANDPILE is hard for $\mathbf{TC}^0$ under constant depth reductions.

Let PARITY be the problem of deciding whether a string of n input bits have an odd number of ones. As PARITY is in $\mathbf{TC}^0$, PARITY is not in $\mathbf{AC}^{1-\epsilon}$

for any $\epsilon > 0$ [6], and $\mathbf{AC}^{1-\epsilon}$ is closed under constant depth reductions, we get the corollary that 1SANDPILE is not in $\mathbf{AC}^{1-\epsilon}$ for any constant $\epsilon > 0$.

It remains to show that MAJORITY constant depth reduces to 1SANDPILE. We have to construct a uniform circuit for MAJORITY using unbounded fan-in AND gates, unbounded fan-in OR gates and 1SANDPILE oracle-gates.

Given an input $x_1x_2 \ldots x_n$ of the MAJORITY problem, we can assume that n is odd. Our circuit first constructs an instance $1y_{11}y_{12}y_{13}y_{21}y_{22}y_{23} \cdots y_{n1}y_{n2}y_{n3}$ of the sandpile problem where $y_{i1} = x_i$, $y_{i2} = 1 - x_i$ and $y_{i3} = 2$, except for $y_{n3} = 1$. For example, we reduce 0101011 to 1 012 102 012 102 012 101. Applying the Moore-Nilsson algorithm, we see that this instance reduces to a stable value with exactly one zero with an index between 1 and $3n$, this index being $n + 1 +$ #ones in $x_1x_2 \ldots x_n$. Thus, to find the majority of $x_1x_2 \ldots x_n$, we just need to check if the index of the unique one in the reduced pile is bigger than $\frac{3n}{2} + 1$. This is easily checked with a uniform constant depth circuit.

4 Discussion and Open Problems

In general, to carry out the program of making formal the informal notion of the "complexity" of a complex dynamical system by identifying "complexity" with the computational complexity of the prediction problem, it seems appropriate to use the finest scale available in the theory of computational complexity.

We have shown a $\mathbf{TC}^0$ lower bound and a **LOGDCFL** upper bound for the one-dimensional sandpile prediction problem. Obviously, getting tighter bounds would be desirable. In particular, is the one-dimensional problem hard for $\mathbf{NC}^1$? Is it in **L** or **NL**?

Moore and Nilsson classified the d-dimensional sandpile prediction problem as **P**-complete for $d \geq 3$ and left open whether the 2-dimensional sandpile prediction problem is also hard for **P**. We may note that the reduction used by Moore and Nilsson to show that the 3-dimensional problem is hard for **P** also establishes that the 2-dimensional problem is hard for $\mathbf{NC}^1$: Their reduction is a reduction from the monotone circuit value problem and the third dimension is only used when implementing crossovers of wires. Thus, the monotone *planar* circuit value problem reduces to 2-dimensional sandpile prediction, and this problem is hard for $\mathbf{NC}^1$. Getting a better lower bound than $\mathbf{NC}^1$ or a better upper bound than **P** for the 2-dimensional sandpile prediction problem would be most interesting.

References

1. P. Bak, C. Tang and K. Wiesenfeld. Self-organized criticality: an explanation of $1/f$ noise. *Physical Review Letters* **59** (1987) 381–384.
2. P. Bak, C. Tang and K. Wiesenfeld. Self-organized criticality. *Physical Review A* **38** (1988) 364–374.
3. D. A. M. Barrington, N. Immerman and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306.

4. A.K. Chandra, L. Stockmeyer and U. Vishkin. Constant depth reducibility. SIAM Journal on Computing **13** (1984) 423–439.
5. D. Griffeath and C. Moore. Life without death is P-complete. *Complex Systems* **4** (1990) 299–318.
6. J. Håstad. *Computational Limitations of Small-Depth Circuits.* ACM doctoral dissertation award 1986. MIT Press, 1987.
7. C. G. Langton. Computation at the edge of chaos. *Physica D* **42** (1990).
8. J. Machta. The computational complexity of pattern formation. *Journal of Statistical Physics* **77** (1994) 755.
9. J. Machta and R. Greenlaw. The computational complexity of generating random fractals. *Journal of Statistical Physics* **82** (1996) 1299
10. C. Moore and M. Nilsson. The Computational Complexity of Sandpiles. *Journal of Statistical Physics* **96** (1999) 205–224. Also available on `http://www.santafe.edu/~moore/`.
11. C. Moore and M.G. Nordahl. Predicting lattice gasses is P-complete. *Santa Fe working Paper* 97-04-034.
12. C. Moore. Majority-vote cellular automata, Ising dynamics, and P-completeness. *Journal of Statistical Physics* **88** (1997) 795–805.
13. K. Moriarty and J. Machta. The computational complexity of the Lorentz lattices gas. *Journal of Statistical Physics* **87** (1997) 1245.
14. W. Ruzzo. Tree-size bounded alternation. *Journal of Computer and Systems Sciences* 21:218-235, 1980.
15. I. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the Association for Computing Machinery*, 25:405-414, 1978.

Categoricity in Restricted Classes

Andrey Morozov

Sobolev Institute of Mathematics,
Koptyug prosp. 4, 630090, Novosibirsk, Russia
morozov@math.nsc.ru

It is well known that only a few structures can be described up to isomorphism by their elementary theories in the class of all models of fixed cardinality. Unfortunately, most of the natural and interesting structures fail to have this property.

The situation changes if we restrict the class of models by adding some more requirements. S.Tennenbaum [4] was the first to prove the standard model of arithmetics is finitely axiomatizable and categorical in the class of all computable models. Later on, A.Morozov [1, 3], A.Nies [2], and others proved categoricity and finite axiomatizability of some other structures in natural restricted classes.

In my talk I will present a survey of results on categoricity and finite axiomatizability of some structures in restricted classes of models and outline main ideas of proofs. It is remarkable that these results essentially use computability theory.

References

1. A.Morozov. On the theories of classes of recursive permutation groups. *Siberian Advances in Mathematics*, 1(1):138–153, 1991.
2. A.Nies. Separating classes of groups by first–order sentences. *Internat. J. Algebra Comput.*, 13:287–302, 2003.
3. A.S.Morozov and A.Nies. Finitely generated groups and first order logic. *Journal of the London mathematical society.* to appear.
4. S. Tennenbaum. Non-archimedean models for arithmetic. *Notices of the American Mathematical Society, p. 270.*, 6:270, (June 1959).

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, p. 349, 2005.

Recursion and Complexity*

Yiannis N. Moschovakis

Department of Mathematics,
University of California, Los Angeles, CA 90095-1555, USA
and
Department of Mathematics,
Graduate Program in Logic, Algorithms and Computation,
University of Athens, Athens, Greece
ynm@math.ucla.edu

My purpose in this lecture is to explain how the representation of algorithms by recursive programs can be used in complexity theory, especially in the derivation of lower bounds for worst-case time complexity, which apply to *all*—or, at least, a very large class of—algorithms. It may be argued that recursive programs are not a new computational paradigm, since their manifestation as Herbrand-Gödel-Kleene systems was present at the very beginning of the modern theory of computability, in 1934. But they have been dissed as tools for complexity analysis, and part of my mission here is to rehabilitate them.

I will draw my examples primarily from van den Dries' [1] and the joint work in [3, 2], incidentally providing some publicity for the fine results in those papers. Some of these results are stated in Section 3; before that, I will set the stage in Sections 1 and 2, and in the last Section 4 of this abstract I will outline very briefly some conclusions about recursion and complexity which I believe that they support.

1 Partial Algebras

A (pointed) *partial algebra* is a structure of the form

$$\boldsymbol{A} = (A, 0, 1, \boldsymbol{\Phi}) = (A, 0, 1, \{\phi^{\boldsymbol{A}}\}_{\phi\in\Phi}), \tag{1}$$

where 0, 1 are distinct points in the universe A, and for every $\phi \in \Phi$,

$$\phi^{\boldsymbol{A}} : A^n \rightharpoonup A$$

is a partial function of some arity n associated by the *signature* Φ with the symbol ϕ. Typical example is the structure of arithmetic

$$\boldsymbol{N} = (\mathbb{N}, 0, 1, =, +, \cdot),$$

* This is an outline of a projected lecture, in which I will refer extensively to and draw conclusions from results already published in [2]; and to make it as self-contained as possible, it has been necessary to quote extensively from [2], including the verbatim repetition of some of the basic definitions.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 350–357, 2005.

which happens to be *total*, i.e., the symbols '=', '+' and '·' are interpreted by total functions, the characteristic function of the identity in the first case, and addition and multiplication for the other two. Genuinely partial algebras typically arise as *restrictions* of total algebras, often to finite sets: if $\{0,1\} \subseteq B \subseteq A$, then

$$\boldsymbol{A} \restriction B = (B, 0, 1, \{\phi^{\boldsymbol{A}} \restriction B\}_{\phi \in \Phi}),$$

where, for any $f : A^n \rightharpoonup A$,

$$f \restriction B(x_1, \ldots, x_n) = w \iff x_1, \ldots, x_n, w \in B \ \&\ f(x_1, \ldots, x_n) = w.$$

An *imbedding* $\iota : \boldsymbol{B} \rightarrowtail \boldsymbol{A}$ from one partial algebra into another (of the same signature) is any injective (total) map

$$\iota : B \rightarrowtail A,$$

such that $\iota(0) = 0$, $\iota(1) = 1$, and for all $\phi \in \Phi$, $\boldsymbol{x} = (x_1, \ldots, x_n), w$ in B,

$$\text{if } \phi^{\boldsymbol{B}}(\boldsymbol{x}) = w, \text{ then } \phi^{\boldsymbol{A}}(\iota(\boldsymbol{x})) = \iota(w),$$

where, of course, $\iota(x_1, \ldots, x_n) = (\iota(x_1), \ldots, \iota(x_n))$. If the identity $\iota(x) = x$ is an imbedding of $\boldsymbol{B}$ into $\boldsymbol{A}$, we call $\boldsymbol{B}$ a *partial subalgebra* of $\boldsymbol{A}$ and write $\boldsymbol{B} \subseteq_p \boldsymbol{A}$. Notice that the definitions here are in the spirit of graph theory, not model theory, i.e., we do not insist that partial subalgebras be closed under the given operations; in particular, for any $B \subseteq A$,

$$(\{0,1\}, --) \subseteq_p \boldsymbol{A} \restriction B \subseteq_p \boldsymbol{A},$$

where $(\{0,1,\}, --))$ is the trivial algebra with universe $\{0,1\}$ and all symbols in Φ interpreted by completely undefined partial functions.

For any $X \subseteq A$ and any number m, we define the set $G_m(X)$ *generated in* $\boldsymbol{A}$ *from* X *in* m *steps* in the obvious way:

$$\begin{aligned} G_0(X) &= \{0,1\} \cup X, \\ G_{m+1}(X) &= G_m(X) \cup \{\phi^{\boldsymbol{A}}(\boldsymbol{x}) \mid \boldsymbol{x} \in G_m(X), \phi \in \Phi, \phi^{\boldsymbol{A}}(\boldsymbol{x}) \downarrow\}. \end{aligned} \tag{2}$$

Notice that if X is finite, then each $G_m(X)$ is a finite set. The set generated by X is the union

$$G(X) = \bigcup\nolimits_{m \in \mathbb{N}} G_m(X),$$

and it determines the partial subalgebra $\boldsymbol{A} \restriction G(X)$ of $\boldsymbol{A}$ generated by X.

The Complexity of Values

Suppose now that $\boldsymbol{A}$ is a partial algebra as in (1) and

$$\chi : A^n \to A$$

is an n-ary function on A which we want to compute *from the givens* $\{\phi^{\boldsymbol{A}}\}_{\phi \in \Phi}$ of $\boldsymbol{A}$—*and nothing else*. It is natural to suppose that this cannot be done unless

each value $\chi(\boldsymbol{x})$ can be generated from the arguments $\boldsymbol{x}$ by successively applying the givens, i.e.,

$$\chi(\boldsymbol{x}) \in G(\boldsymbol{x}) \quad (\boldsymbol{x} \in A^n);$$

and that if this holds and we set

$$g_\chi(\boldsymbol{x}) = \text{the least } m \text{ such that } \chi(\boldsymbol{x}) \in G_m(\boldsymbol{x}),$$

then any algorithm which computes $\chi(\boldsymbol{x})$ will need at least $g_\chi(\boldsymbol{x})$ steps. This can be argued very generally, and it can be used to derive hard lower bounds for all algorithms which compute $\chi(\boldsymbol{x})$ from specific givens. Van den Dries used it in [1] to derive a triple logarithmic lower bound for the function

$$\gcd(x, y) = \text{the greatest common divisor of } x \text{ and } y \quad (x, y \in \mathbb{N}, x \geq y \geq 1)$$

from addition, subtraction and division with remainder, i.e., the two functions

$$\mathrm{iq}(x, y) = q, \quad \mathrm{rem}(x, y) = r,$$

where q and r are the unique natural numbers such that

$$x = yq + r \quad 0 \leq r < y.$$

His proof introduced some ingenious ideas from number theory (which we will mention further down), and the result was at least a first step in an effort to establish that the classical Euclidean algorithm is optimal; the Euclidean, of course, has single logarithmic complexity, and it uses only the remainder function $\mathrm{rem}(x, y)$.

The big advantage of the complexity function $g_\psi(\boldsymbol{x})$ is that lower bound results about it apply to all algorithms, whatever algorithms are. On the other hand, it cannot be used to establish lower bounds for decision problems, where the function that we want to compute takes on only the values 0 or 1: for that we need to make some assumptions about algorithms, which we do next.

2 Recursive Programs

The *terms* of the language $L(\Phi)$ of programs in the signature Φ are defined by the recursion

$$E :\equiv 0 \mid 1 \mid \mathsf{v}_i \mid \phi(E_1, \ldots, E_n) \mid \mathsf{p}_i^n(E_1, \ldots, E_n) \mid \text{if } (E_0 = 0) \text{ then } E_1 \text{ else } E_2,$$

where v_i is one of a list of *individual variables*; p_i^n is one of a list of n-ary (partial) *function variables*; and ϕ is any n-ary symbol in Φ. These terms are interpreted as usually in any Φ-partial algebra $\boldsymbol{A}$ and relative to any assignment π which assigns some $\pi(\mathsf{v}_i) \in A$ to each individual variable v_i, and some n-ary partial function $\pi(\mathsf{p}_i^n) : A^n \rightharpoonup A$ to each function variable p_i^n:

$$[\![E]\!](\pi) = [\![E]\!](\boldsymbol{A}, \pi) = \text{the value (if defined) of } E \text{ in } \boldsymbol{A} \text{ for the assignment } \pi;$$

and if the variables which occur in E are in the list $(\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{p}_1, \ldots, \mathsf{p}_m)$, then E defines a (partial) *functional*

$$F_E(\boldsymbol{x}, \boldsymbol{p}) = [\![E]\!](\{(\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{p}_1, \ldots, \mathsf{p}_m) := \boldsymbol{x}, \boldsymbol{p}\}) \tag{3}$$

which is monotone and continuous.

A *recursive* (or McCarthy) *program* of $L(\Phi)$ is any system of *recursive term equations*

$$\alpha : \quad \begin{cases} p_\alpha(\boldsymbol{x}) = E_0 \\ p_1(\boldsymbol{x}_1) = E_1 \\ \qquad \vdots \\ p_K(\boldsymbol{x}_K) = E_K \end{cases} \tag{4}$$

such that $p_\alpha, p_1, \ldots, p_K$ are distinct function variables; $p_1, \ldots, p_K$ are the only function variables which occur in $E_0, \ldots, E_K$; and for each i, the free, individual variables in each E_i are in the list $\boldsymbol{x}_i$. The term E_0 is the *head* of α, the remaining terms are its *body*, and we may describe the mutual recursive definition expressed by α by the simple notation

$$\alpha \equiv E_0 \text{ where } \{p_1 = E_1, \ldots, p_K = E_K\}.$$

The function variables $p_1, \ldots, p_K$ are bound in this expression.

To interpret a program α on a Φ-structure $\boldsymbol{A}$, we observe that its parts define the system of mutually recursive equations

$$\begin{aligned} p_\alpha(\boldsymbol{x}) &= F_{E_0}(\boldsymbol{x}, p_1, \ldots, p_K), \\ p_1(\boldsymbol{x}_1) &= F_{E_1}(\boldsymbol{x}_1, p_1, \ldots, p_K), \\ &\vdots \\ p_K(\boldsymbol{x}) &= F_{E_K}(\boldsymbol{x}_K, p_1, \ldots, p_K), \end{aligned}$$

using (3), which by the usual methods has a set of least, mutual solutions

$$\overline{p}_\alpha, \overline{p}_1, \ldots, \overline{p}_K;$$

we then let

$$[\![\alpha]\!] = [\![\alpha]\!](\boldsymbol{A}) = \overline{p}_\alpha : A^n \rightharpoonup A,$$

so that the partial function "computed" by α on $\boldsymbol{A}$ is the component of the tuple of least solutions of the mutual recursion determined by α which corresponds to the head term.

A partial function $f : A^n \rightharpoonup A$ is $\boldsymbol{A}$-*recursive* if it is computed by some recursive program.

Except for the notation, these are the programs introduced by John McCarthy in [6]. McCarthy proved that if $\boldsymbol{N}_0 = (\mathbb{N}, 0, 1, S, P)$ is the simplest structure on the natural numbers with just the successor and the predecessor operations as given, then the $\boldsymbol{N}_0$-recursive partial functions are exactly the Turing-computable ones. To justify the connection with computability in general,

one must of course explain how recursive programs compute partial functions, in effect to construct an *implementation* of $L(\Phi)$ on an arbitrary partial Φ-algebra $\boldsymbol{A}$; but it is well-known how to do this (in many ways), and we will not be concerned with it.

We note two basic lemmas, whose proofs are very easy:

Lemma 1 (Imbedding). *If $\iota : \boldsymbol{B} \to \boldsymbol{A}$ is an imbedding from one Φ-algebra into another, then for every Φ-program α and all $\boldsymbol{x}, w \in B$,*

$$\text{if } [\![\alpha]\!](\boldsymbol{B}, \boldsymbol{x}) = w, \quad \text{then } [\![\alpha]\!](\boldsymbol{A}, \iota(\boldsymbol{x})) = \iota(w).$$

In particular, if $\boldsymbol{B} \subseteq_p \boldsymbol{A}$, then for every Φ-program α and all $\boldsymbol{x}, w \in B$,

$$\text{if } [\![\alpha]\!](\boldsymbol{B}, \boldsymbol{x}) = w, \quad \text{then } [\![\alpha]\!](\boldsymbol{A}, \boldsymbol{x}) = w.$$

Lemma 2 (Finiteness). *For every Φ-algebra $\boldsymbol{A}$, every recursive Φ-program α and all $\boldsymbol{x}, w \in A$, if $[\![\alpha]\!](\boldsymbol{A}, \boldsymbol{x}) = w$, then there exists some m such that*

$$w \in G_m(\boldsymbol{x}) \quad \text{and} \quad [\![\alpha]\!](\boldsymbol{A} \restriction G_m(\boldsymbol{x}), \boldsymbol{x}) = w.$$

The Finiteness Lemma expresses the simple proposition that computations are finite, and so they live in some finite subset $G_m(\boldsymbol{x})$ of A generated by the input; but it leads directly to the next, fundamental notion of complexity.

The Basic (Structural) Complexity

For each recursive program α, each partial algebra $\boldsymbol{A}$, and each $\boldsymbol{x}$ such that $[\![\alpha]\!](\boldsymbol{x})\downarrow$, we set

$$C_\alpha(\boldsymbol{x}) = \text{the least } m \text{ such that } [\![\alpha]\!](\boldsymbol{A} \restriction G_m(\boldsymbol{x}), \boldsymbol{x}) = [\![\alpha]\!](\boldsymbol{x}).$$

Roughly speaking, the basic complexity $C_\alpha(\boldsymbol{x})$ measures the minimum number of nested calls to the givens which is required for the computation of $[\![\alpha]\!](\boldsymbol{x})$. It cannot be realistically attained by any actual implementation of α, but it is a plausible lower bound for the time complexity of any implementation.

The Finiteness Lemma now yields immediately the key tool for deriving lower bound results about the basic complexity of recursive programs:

Lemma 3 (The Imbedding Test). *Let $\boldsymbol{A}$ be a partial algebra as in* (1)*, suppose that $\chi : A^n \to A$, and assume that for some $\boldsymbol{x} \in A^n$ and some m, there is an imbedding*

$$\iota : \boldsymbol{A} \restriction G_m(\boldsymbol{x}) \to \boldsymbol{A}$$

such that

$$\chi(\iota(\boldsymbol{x})) \neq \iota(\chi(\boldsymbol{x}));$$

it follows that for every recursive program α which computes χ in $\boldsymbol{A}$,

$$C_\alpha(\boldsymbol{x}) > m.$$

3 Two Lower Bound Results About Coprimeness

We quote here from [2] two lower bound results about the relation of *coprimeness*

$$x \perp y \iff x, y \geq 1 \ \& \ (\forall d > 1)[d \nmid x \vee d \nmid y],$$

which are obtained using the Imbedding Test, Lemma 3.

For the first, let

$$\mathbf{Lin}_0 = \{=, <, \text{parity}, 2\cdot, \frac{1}{2}\cdot, +, \dot{-}\} \tag{5}$$

where the relations stand for their characteristic functions and

$$2 \cdot (x) = 2x, \quad \frac{1}{2} \cdot (x) = \text{iq}(x, 2).$$

Knuth [5] describes the *binary algorithm* algorithm of Stein which computes the gcd (and hence decides coprimeness) from $\mathbf{Lin}_0$ in logarithmic time.[1] The Stein algorithm is optimal among recursive algorithms (up to a multiplicative constant), because of the following:

Theorem 4 ([2]). *If a recursive program α decides the coprimeness relation in the algebra $\boldsymbol{A}_0 = (\mathbb{N}, \mathbf{Lin}_0)$, then for all $a > 2$,*

$$C_\alpha(a, a^2 - 1) \geq \frac{1}{10} \log(a^2 - 1). \tag{6}$$

The proof goes by showing that if $m < \frac{1}{10} \log(a^2 - 1)$, then there is an imbedding

$$\iota : \boldsymbol{A}_0 \restriction G_m(a, a^2 - 1) \to \boldsymbol{A}$$

such that for some λ,

$$\iota(a) = \lambda a, \quad \iota(a^2 - 1) = \lambda(a^2 - 1),$$

so that ι carries the coprime pair $(a, a^2 - 1)$ to a pair of numbers which are not coprime, and hence $C_\alpha(a, a^2 - 1) > m$ by the Imbedding Test. It is not possible to describe in this abstract how this imbedding is defined, but it is quite simple. Not so for the next result—the Main Theorem of [2]—where the same idea is used, but the relevant imbedding is much harder to define and depends on Liouville's Theorem on "good approximations" of algebraic irrationals, cf. [4]:

Theorem 5 ([2]). *If a recursive program α decides the coprimeness relation in the algebra $\boldsymbol{A}_1 = (\mathbb{N}, \mathbf{Lin}_0, \text{iq}, \text{rem})$, then for infinitely many coprime pairs $a > b > 1$,*

$$C_\alpha(a, b) \geq \frac{1}{10} \log \log a. \tag{7}$$

In fact, (7) holds for all coprime $a > b > 1$ such that

[1] This is also specified in [2].

$$\left|\frac{a}{b} - \sqrt{2}\right| < \frac{1}{b^2} \tag{8}$$

(and there are infinitely many such a, b by a classical result).

The target here is the Euclidean algorithm which decides coprimeness in logarithmic time, and so the theorem is one log short of what is needed to establish the (plausible) optimality of the Euclidean. But it is as good as any known lower bound for coprimeness from its givens, and perhaps unique in its uniformity—it yields the same lower bound, on the same inputs for all recursive programs.

4 Inessential (Logical) Extensions of Partial Algebras

The next natural question is whether Theorems 4 and 5 also hold for other "computational paradigms", for example random access machines. Of course they do, and by more-or-less the same proofs, adapted to the idiosyncracies of each model which do not affect the basic, arithmetical facts that enter into the arguments. Rather than do this one computation model at a time, however, we look for a general result which might suggest that these lower bounds hold *for all algorithms*.

Let $\boldsymbol{A} = (A, 0, 1, \{\phi^{\boldsymbol{A}}\}_{\phi\in\Phi})$ be a partial algebra. An *inessential extension* of $\boldsymbol{A}$ is any partial algebra

$$\boldsymbol{B} = (B, 0, 1, \{\phi^{\boldsymbol{A}}\}_{\phi\in\Phi}, \{\psi^{\boldsymbol{B}}\}_{\psi\in\Psi})$$

with the following properties:

(IE1) $A \subseteq B$, and $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same 0 and 1;

(IE2) every permutation π of A fixing 0 and 1 can be extended to a permutation π^B of B such that for every "new given" $\psi = \psi^{\boldsymbol{B}}$ of arity n and all $x_1, \ldots, x_n \in B$,

$$\pi^B\psi(x_1, \ldots, x_n) = \psi(\pi^B x_1, \ldots, \pi^B x_n). \tag{9}$$

Here we view each "old given" $\phi^{\boldsymbol{A}}$ as a partial function on B, undefined when one of its arguments is not in A.

We might also call these extensions *logical*, since the property we demand of the new givens in $\boldsymbol{B}$ is reminiscent (or better: a relativization to the given algebra $\boldsymbol{A}$) of Tarski's *logical functions*. In any case, Lemma 3 extends directly (and easily) to them:

Lemma 6 (The Extended Imbedding Test, [2]). *Let $\boldsymbol{A}$ be a partial algebra as in* (1)*, suppose that $\chi : A^n \to A$, and assume that for some $\boldsymbol{x} \in A^n$ and some m, there is an imbedding*

$$\iota : \boldsymbol{A} \restriction G_m(\boldsymbol{x}) \to \boldsymbol{A}$$

such that

$$\chi(\iota(\boldsymbol{x})) \neq \iota(\chi(\boldsymbol{x}));$$

it follows that for every recursive program α which computes χ in some inessential extension $\boldsymbol{B}$ of $\boldsymbol{A}$,

$$C_\alpha(\boldsymbol{x}) > m.$$

And, of course, random access machines relative to any set $\boldsymbol{\Phi}$ of functions on $\mathbb{N}$ can be faithfully represented by recursive programs on inessential extensions of $(\mathbb{N}, 0, 1, \boldsymbol{\Phi})$, as are all computational models *relative to* $\boldsymbol{\Phi}$; and so Theorems 4 and 5 also hold for them.

In fact, the familiar computational paradigms for computation on $\mathbb{N}$ accept some fixed, number-theoretic functions $\boldsymbol{\Phi}$ as givens (the successor and predecessor, equality, addition, etc.), and they also assume some "computational constructs" which are characteristic of them and independent of $\boldsymbol{\Phi}$, e.g., branching, recursion, reading from and writing to registers or stacks, higher-type logical operations like λ-abstraction and β-conversion, etc. In [7, 8] it is argued that all algorithms from given operations can be faithfully represented by suitable recursive programs on the structure determined by the givens: now the results we have discussed here suggest the following, more concrete interpretation of that proposal for algorithms from first-order givens on some set A:

Refined Church-Turing Thesis for Algorithms, ([2]). *Every algorithm α from a set $\boldsymbol{\Phi}$ of partial functions and relations on a set A can be represented faithfully by a recursive program β on some inessential extension $\boldsymbol{B}$ of the partial algebra $\boldsymbol{A} = (A, 0, 1, \boldsymbol{\Phi})$.*

If we assume this Thesis, then the Extended Imbedding Test makes it possible to establish lower bounds for all algorithms from a set of first-order givens $\boldsymbol{\Phi}$ on A, whenever we can produce the appropriate imbeddings.

References

1. Lou van den Dries. Generating the greatest common divisor, and limitations of primitive recursive algorithms. *Foundations of computational mathematics*, 3:297–324, 2003.
2. Lou van den Dries and Yiannis N. Moschovakis. Is the Euclidean algorithm optimal among its peers? *The Bulletin of Symbolic Logic*, 10:390–418, 2004.
3. Lou van den Dries and Yiannis N. Moschovakis. Arithmetic complexity. 200? in preparation.
4. G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Clarendon Press, Oxford, fifth edition (2000). originally published in 1938.
5. D. E. Knuth. *The Art of Computer Programming. Fundamental Algorithms*. Addison-Wesley, second edition, 1973.
6. J. McCarthy. A basis for a mathematical theory of computation. In P. Braffort and D Herschberg, editors, *Computer programming and formal systems*, pages 33–70. North-Holland, 1963.
7. Yiannis N. Moschovakis. On founding the theory of algorithms. In H. G. Dales and G. Oliveri, editors, *Truth in mathematics*, pages 71–104. Clarendon Press, Oxford, 1998.
8. Yiannis N. Moschovakis. What is an algorithm? In B. Engquist and W. Schmid, editors, *Mathematics unlimited – 2001 and beyond*, pages 919–936. Springer, 2001.

FM-Representability and Beyond

Marcin Mostowski[1] and Konrad Zdanowski[2]

[1] Department of Logic, Institute of Philosophy, Warsaw University,
Krakowskie Przedmieście 3, 00-047 Warszawa, Poland
m.mostowski@uw.edu.pl
[2] Institute of Mathematics, Polish Academy of Science,
ul. Śniadeckich 8., P.O.Box 21,
00-956 Warszawa 10, Poland
kz@impan.gov.pl

Abstract. This work concerns representability of arithmetical notions in finite models. It follows the paper by Marcin Mostowski [1], where the notion of FM–*representability* has been defined. We discuss how far this notion captures the methodological idea of representing infinite sets in finite but potentially infinite domains.

We consider mainly some weakenings of the notion of FM–representability. We prove that relations *weakly* FM–*representable* are exactly those being Σ^0_2–definable. Another weakening of the notion, namely *statistical representability*, turns out to be equivalent to the original one. Additionally, we consider the complexity of sets of formulae naturally defined in finite models. We state that the set of sentences true in almost all finite arithmetical models is Σ^0_2–complete and that the set of formulae FM–representing some relations is Π^0_3–complete.

1 Introduction

This work concerns mainly the following problem.

> *Let us suppose that our world is finite, but not of a restricted size. It means that everytime it can be enlarged by a finite number of new entities. This assumption says, in Aristotelian words (see [2], Physics, book* 3*), that the world is finite but potentially infinite. Then, which infinite sets can be reasonably described in our language?*

For simplifying the problem we restrict our attention to sets (and relations) of natural numbers and we assume that our world contains only natural numbers.

Technically, the problem appears when one is trying to transfer some classical ideas into finite–models theoretic framework. It requires frequently a uniform representation for various infinite relations in finite models. As a rule, uniformity means that the representation of a relation is given by one formula. Of course in a single finite model only a finite approximation of any infinite relation can be defined. Therefore we have to consider representability in infinite classes of

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 358–367, 2005.

finite models — intuitively *finite but potentially infinite models.*[1] In the paper [1] an attempt to make the notion precise has been made and FM–representability theorem has been proved (see Theorem 5).[2]

Let R be a set of natural numbers. Then we say that R is FM–represented by a formula $\varphi(x)$ if for each initial segment I of natural numbers $\varphi(x)$ correctly describes R for all elements from I in all sufficiently large finite interpretations. Originally the notion was motivated by an attempt to transfer the Tarski's method of classifying concepts by means of truth definitions to finite models.[3] In this case we have to describe syntax of considered languages in finite models. Needed syntactical relations are essentially infinite. Therefore, the notion of FM–representability appeared as an answer to this problem.

In this paper we concentrate on the notion of FM–representability and some possible weakenings of it. We show that, in a sense, the notion captures strongly the idea of representing relations in finite models.

2 Basic Notions

We start with the crucial definition of FM–domain.

Definition 1. *Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be a finite set of arithmetical relations on ω. By an $\mathcal{R}$–domain we mean the model $\mathcal{A} = (\omega, R_1, \ldots, R_k)$. We consider finite initial segments of these models. Namely, for $n \geq 1$, by $\mathcal{A}_n$ we denote the structure*

$$\mathcal{A}_n = (\{0, \ldots, n-1\}, R_1^n, \ldots, R_k^n),$$

where, for $i = 1, \ldots, k$, the relation R_i^n is the restriction of the relation R_i to the set $\{0, \ldots, n-1\}$. We treat n-ary functions as $n+1$-ary relations.

The FM*–domain of $\mathcal{A}$ (or* FM*–domain of $\mathcal{R}$), denoted by* FM($\mathcal{A}$)*, is the family $\{\mathcal{A}_n : n \in \omega\}$.*

Throughout this paper we are interested mainly in the family FM($\mathbb{N}$), for $\mathbb{N} = (\omega, +, \times)$. By arithmetical formulae we mean first order formulae with addition and multiplication treated as ternary predicates. The standard ordering $x \leq y$ is definable by the formula $\exists z\, x + z = y$. Its strict version, $x < y$, is defined as $x \leq y \wedge x \neq y$. The constants 0 and MAX are defined respectively as $\leq$–smallest and $\leq$–greatest elements. For each $n \in \omega$, by $\bar{n}$ we mean the constant denoting the n-th element in the ordering $\leq$ counting from 0. If there is no such

[1] In the context of foundations of mathematics a very similar approach to potential infinity is presented by Jan Mycielski in [3].

[2] Some consequences of this idea are also discussed in [4], [5].

[3] The basics of the method of truth definitions in finite models were formulated in [6]. The paper [1] covers [6], giving additionally some refinement of the method. It was applied then in [4], and [7] for classifying finite order concepts in finite models. Some applications of the method for classifying computational complexity classes can be found in [8].

element we take $\bar{n} = \mathrm{MAX}$. We write $x|y$ for $\exists z \leq y(1 < z \wedge zx = y)$. It is known that all these notions are definable by bounded formulae. Thus, their interpretations conform to their intended meaning also in models from $\mathrm{FM}(\mathbb{N})$.

Let us mention, that in [4] a finite axiomatization ST has been presented which characterizes, up to isomorphism, the family $\mathrm{FM}(\mathbb{N})$ within the class of all finite models.

The other notions which we use here are fairly standard, one can consult e.g. [9] and [10] for model or recursion theoretic concepts, respectively. We write $\{e\}$ to denote the partial function computed by the Turing machine with the index e. $\{e\}(n)\uparrow$ means that the function $\{e\}$ is not defined on n, and $\{e\}(n)\downarrow$ means that $\{e\}(n)$ is defined. We put $W_e = \{n \in \omega : \{e\}(n)\downarrow\}$.

We consider the family of Σ^0_n (Π^0_n) relations which are definable in $\mathbb{N}$ by Σ^0_n (Π^0_n) formulae. Δ^0_n are relations which are definable by Σ^0_n and Π^0_n formulae.

$R \subseteq \omega^r$ is many one reducible to $S \subseteq \omega^s$ ($R \leq_m S$) if there exists a total recursive function f such that for all $a_1, \ldots, a_r \in \omega$,

$$(a_1, \ldots, a_r) \in R \text{ if and only if } f(a_1, \ldots, a_r) \in S.$$

A relation S is complete for a class $\mathcal{K}$ if $S \in \mathcal{K}$ and for any other $R \in \mathcal{K}$, $R \leq_m S$.

We say that R is Turing reducible to S ($R \leq_T S$) if there is an oracle Turing machine which decides R using S as an oracle. R and S are Turing equivalent if $R \leq_T S$ and $S \leq_T R$. The degree of R, denoted by $\deg(R)$, is the class of all relations which are Turing equivalent to R. In particular, $\mathbf{0}'$ is the degree of any recursively enumerable (RE) complete set, and $\mathbf{0}''$ is the degree of any Σ^0_2–complete set.

We use bald characters, e.g. $\mathbf{a}$, for valuations in a given model $\mathcal{A}$. We write $|\mathcal{A}|$ for the universe of a model $\mathcal{A}$. If $\varphi(x_1, \ldots, x_k)$ is a formula in the vocabulary of $\mathcal{A}$ with all free variables between $x_1, \ldots, x_k$ we write $\mathcal{A} \models \varphi[a_1, \ldots, a_k]$, for $a_1, \ldots, a_k \in |\mathcal{A}|$, when φ holds in $\mathcal{A}$ under any valuation $\mathbf{a}$ for which $\mathbf{a}(x_i) = a_i$, for $i = 1, \ldots, k$.

Definition 2. *Let $\varphi(x_1, \ldots, x_r)$ be an arithmetical formula and $a_1, \ldots, a_r \in \omega$. We say that φ is true of $a_1, \ldots, a_r$ in all sufficiently large finite models ($\models_{\mathrm{sl}} \varphi[a_1, \ldots, a_r]$) if and only if $\exists k \forall n \geq k\ \mathbb{N}_n \models \varphi[a_1, \ldots, a_r]$ (or, in other words, if φ is true of $a_1, \ldots, a_r$ in almost all finite models from $\mathrm{FM}(\mathbb{N})$).*

For each unbounded family of finite models $\mathcal{K}$, by $\mathrm{sl}(\mathcal{K})$ we denote the set of formulae which are true in almost all models from $\mathcal{K}$. In particular, $\models_{\mathrm{sl}} \varphi$ means that $\varphi \in \mathrm{sl}(\mathrm{FM}(\mathbb{N}))$.

Definition 3. *We say that $R \subseteq \omega^r$ is* FM*–represented by a formula $\varphi(x_1, \ldots, x_r)$ if and only if for each $a_1, \ldots, a_r \in \omega$ both of the following conditions hold:*

(i) $\models_{sl} \varphi[a_1, \ldots, a_r]$ *if and only if* $R(a_1, \ldots, a_r)$.
(ii) $\models_{sl} \neg\varphi[a_1, \ldots, a_r]$ *if and only if* $\neg R(a_1, \ldots, a_r)$.

We say that R is FM*–representable if there is an arithmetical formula φ which* FM*–represents R.*

The notion of FM–representability has been defined in [1] in a slightly different way. We summarize various equivalent conditions in the following

Proposition 4. *Let $R \subseteq \omega^r$ and $\varphi(x_1, \ldots, x_r)$ be a formula in a vocabulary of* FM($\mathbb{N}$). *The following conditions are equivalent:*

1. *$\varphi(x_1, \ldots, x_r)$* FM*–represents R,*
2. *for each m there is k such that for all $a_1, \ldots, a_r \leq m$,*

$$R(a_1, \ldots, a_r) \text{ if and only if } \mathbb{N}_i \models \varphi[a_1, \ldots, a_r],$$

for all $i \geq k$.

The second condition expresses the intuition that φ FM–represents R in FM($\mathbb{N}$) if each finite fragment of R is correctly described by φ in all sufficiently large models from FM($\mathbb{N}$).

The main characterization of the notion of FM–representability is given by the following

Theorem 5 (FM**–representability theorem, see [1]).** *Let $R \subseteq \omega^r$. R is* FM*–representable if and only if R is of degree $\leq \mathbf{0}'$ (or, equivalently, is Δ^0_2–definable).*

The theorem does not depend on the strength of the underlying logic provided that the truth relation for this logic restricted to finite models is recursive and it contains first order logic. On the other hand, it is surprising that the theorem requires relatively weak arithmetical notions. In [11] it is proved that it holds in FM–domain of multiplication. It is improved in [12] to the divisibility relation.

3 Weak FM-Representability

As the most natural weakening of the notion of the notion of FM–representability we consider the following:

Definition 6. *A relation $R \subseteq \omega^r$ is weakly* FM*–representable if there is a formula $\varphi(x_1, \ldots, x_r)$ with all free variables among $x_1, \ldots, x_r$ such that for all $a_1, \ldots, a_r \in \omega$,*

$$(a_1, \ldots, a_r) \in A \text{ if and only if } \models_{sl} \varphi[a_1, \ldots, a_r].$$

Since the definition of $\models_{\mathrm{sl}} \varphi$ can be expressed as an Σ^0_2–sentence the following holds.

Fact 7. *Let $R \subseteq \omega^r$. If R is weakly* FM*–representable, then $R \in \Sigma^0_2$.*

The reverse of the implication from Fact 7 will be proved after evaluating the degree of the theory sl(FM($\mathbb{N}$)).

As an analogue of the relation between FM–representability and weak FM–representability we recall the relation between strong and weak representability

in Peano arithmetic. We say that a relation $R \subseteq \omega^r$ is strongly PA–representable if there is a PA–formula $\varphi(x_1, \ldots, x_r)$ with all free variables among $x_1, \ldots, x_r$ such that for all $n_1, \ldots, n_r \in \omega$,

$$(n_1, \ldots, n_r) \in R \iff \mathrm{PA} \vdash \varphi(\bar{n}_1, \ldots, \bar{n}_r)$$
$$(n_1, \ldots, n_r) \notin R \iff \mathrm{PA} \vdash \neg\varphi(\bar{n}_1, \ldots, \bar{n}_r).$$

$R \subseteq \omega^r$ is weakly PA–representable if there is a PA–formula $\varphi(x_1, \ldots, x_r)$ with all free variables among $x_1, \ldots, x_r$ such that for all $n_1, \ldots, n_r \in \omega$,

$$(n_1, \ldots, n_r) \in R \iff \mathrm{PA} \vdash \varphi(\bar{n}_1, \ldots, \bar{n}_r).$$

A relation R is strongly PA–representable if and only if it is decidable. R is weakly PA–representable if and only if R is recursively enumerable. If R and its complement are weakly PA–representable, then R is strongly PA–representable. We state the analogous fact for FM–representability and weak FM–representability. It follows easily from the known relations between the classes Σ^0_2 and Δ^0_2.

Fact 8. *Let $R \subseteq \omega^r$. R and $\omega^r - R$ are weakly* FM*–representable if and only if R is* FM*–representable.*

Below, we prove the stronger fact that weakly FM–representable relations are exactly the Σ^0_2 relations.

Firstly, we consider some properties of coding computations and the formula $\mathrm{Comp}(e, c)$ which says that c is a finished computation of the machine e. (Here and in what follows by a Turing machine we mean a deterministic Turing machine.) We construct $\mathrm{Comp}(e, c)$ using Kleene predicate $\mathrm{T}(e, x, c)$, which means that c is a finished e–computation with the input x. It is known that this predicate is definable by an arithmetical formula with all quantifiers bounded by c. Moreover, if $\mathrm{T}(e, x, c)$ then $e < c$ and $x < c$. We define $\mathrm{Comp}(e, c)$ as $\exists x < c\, \mathrm{T}(e, x, c)$.

Now let us state a few facts about the formula $\mathrm{Comp}(e, c)$. All quantifiers in $\mathrm{Comp}(e, c)$ are bounded by c. It follows that the truth value of $\mathrm{Comp}(e, c)$ in a given model M does not depend on the elements in M greater than c and that $\mathrm{Comp}(e, c)$ will hold in a given model $M \in \mathrm{FM}(\mathbb{N})$ as soon as the code of the computation appears in M.

Now, we state the lemma summarizing these considerations.

Lemma 9. *There is a formula* $\mathrm{Comp}(x, y)$ *such that*

$$\forall e\, \forall c\, \forall M \in \mathrm{FM}(\mathbb{N})(\mathrm{card}(M) > c \Rightarrow$$
$$(c \textit{ is a finished computation of } e \iff M \models \mathrm{Comp}[e, c])).$$

Definition 10. *By* Fin *we mean the set of indices of Turing Machines having finite domains, i. e.*

$$\mathrm{Fin} = \{e \in \omega : \exists n \in \omega\ \mathrm{card}(W_e) = n\}.$$

By a well known result from recursion theory (see e.g. [10]) Fin is Σ_2^0–complete.

Lemma 11. Fin *is weakly* FM*–representable.*

Proof. Let $\varphi(x)$ be the formula $\neg\mathrm{Comp}(x, \mathrm{MAX})$, where $\mathrm{Comp}(x, y)$ is the formula from the last lemma. By properties of Comp stated there, for all e,

$$e \in \mathrm{Fin} \text{ if and only if } \models_{sl} \varphi[e].$$

If $e \in \mathrm{Fin}$ then there are only finitely many finished computations of e. (Here, we use the assumption that all machines are deterministic.) In this case φ is true of e in all models in which MAX is greater than the biggest computation of e. On the other hand, if $\models_{\mathrm{sl}} \varphi[e]$, then there are only finitely many finished computations of e. Thus, the domain of e is also finite.

Thus, Fin is weakly FM–representable. □

We have the following lemma.

Lemma 12. *The family of weakly* FM*–representable relations is closed on many–one reductions.*

Proof. For simplicity we consider only sets $A, B \subseteq \omega$. Let $f : \omega \longrightarrow \omega$ be a reduction from A to B that is for all z,

$$z \in A \text{ if and only if } f(z) \in B$$

and let $\varphi_B(x)$ weakly FM–represent B. Additionally, let $\psi_f(x, y)$ FM–represent the graph of f. Now, the following formula $\varphi_A(x)$ FM–represents A,

$$\exists y\,(\psi_f(x, y) \wedge \forall y' < y \neg\psi_f(x, y') \wedge \varphi_B(y)).$$

Here, we need to add the conjunct $\forall y' < y \neg\psi_f(x, y')$ to force the uniqness of y. □

As a corollary from Lemmas 11 and 12 we obtain the following characterization of weak FM–representability.

Theorem 13. *Let $R \subseteq \omega^r$. R is weakly* FM*–representable if and only if A is Σ_2^0.*

Now, we are in a position to solve some questions which were put, explicitly or implicitly, in [1]. Let us recall that $\mathrm{sl}(\mathrm{FM}(\mathbb{N})) = \{\varphi : \models_{\mathrm{sl}} \varphi\}$. So, $\mathrm{sl}(\mathrm{FM}(\mathbb{N}))$ is the theory of almost all finite models from $\mathrm{FM}(\mathbb{N})$. By the definition of $\models_{\mathrm{sl}}$ the above set is in Σ_2^0.

In [1], it was proven by the method of undefinability of truth, that

$$\mathbf{0}' < deg(\mathrm{sl}(\mathrm{FM}(\mathbb{N}))) \leq \mathbf{0}''.$$

Here we strengthen this result by the following,

Theorem 14. sl(FM(ℕ)) *is* Σ^0_2*–complete, so its degree is* $\mathbf{0''}$*.*

Proof. We know that sl(FM(ℕ)) is Σ^0_2. It is Σ^0_2–complete by the procedure from the proof of Lemma 11 which reduces Fin to sl(FM(ℕ)). We put $f(e) = \ulcorner \neg \mathrm{Comp}(\bar{e}, \mathrm{MAX}) \urcorner$. By properties of $\mathrm{Comp}(x, y)$ we obtain:

$$e \in \mathrm{Fin} \text{ if and only if } f(e) \in \mathrm{sl}(\mathrm{FM}(\mathbb{N})).$$

Since Fin is Σ^0_2–complete then sl(FM(ℕ)) is also complete. □

Let us observe that the degree of sl(FM(ℕ)) does not depend on the underlining logic provided it has decidable "*truth in a finite model*" relation and contains first order logic.

Now, let us consider the complexity of the question whether a given formula $\varphi(x_1, \ldots, x_k)$ with free variables $x_1, \ldots, x_k$ FM–represents some relation in FM(ℕ). Let us define the set

$$F_{Det} = \{\varphi(x_1, \ldots, x_k) : \\ \forall n_1 \ldots n_k \in \omega \ \models_{sl} \varphi[n_1, \ldots, n_k] \text{ or } \models_{sl} \neg\varphi[n_1, \ldots, n_k]\}.$$

F_{Det} is the set of formulae which are determined for all substitutions of constant numerical terms for their free variables. In other words, this is the set of formulae which FM–represent some concepts.

We have the following theorem characterizing the degree of F_{Det}.

Theorem 15. F_{Det} *is* Π^0_3*–complete.*

Proof. F_{Det} has a Π^0_3 definition so it is a Π^0_3 relation. Now, let $A \subseteq \omega^k$ be a Π^0_3–relation. We show a many–one reduction from A to F_{Det}.

There is a recursive relation R such that for all $n_1, \ldots, n_k \in \omega$,

$$(n_1, \ldots, n_k) \in A \text{ if and only if } \forall x \exists y \forall z R(n_1, \ldots, n_k, x, y, z).$$

Since Fin is Σ^0_2–complete, we have a total recursive function $g : \omega^{k+1} \longrightarrow \omega$ such that for all $n_1, \ldots, n_k \in \omega$,

$$\forall x \exists y \forall z R(n_1, \ldots, n_k, x, y, z) \text{ if and only if } \forall x\, g(n_1, \ldots, n_k, x) \in \mathrm{Fin}.$$

Now, let $\psi_g(x_1, \ldots, x_k, x, y)$ FM–represent the graph of g and let $\varphi(x_1, \ldots, x_k, x)$ be the following formula

$$\exists y(\psi_g(x_1, \ldots, x_k, x, y) \wedge \forall z < y \neg\psi_g(x_1, \ldots, x_k, x, z) \wedge \neg\mathrm{Comp}(y, \mathrm{MAX})),$$

where $\mathrm{Comp}(x, y)$ is the formula from Lemma 9. Because we consider only deterministic Turing machines $\mathrm{Comp}(y, \mathrm{MAX})$ can be determined only negatively. Thus, for all $n_1, \ldots, n_k \in \omega$,

$$(n_1, \ldots, n_k) \in A \text{ if and only if } \forall m \in \omega \ \models_{\mathrm{sl}} \varphi(\bar{n}_1, \ldots, \bar{n}_k, \bar{m}) \\ \text{if and only if } \varphi(\bar{n}_1, \ldots, \bar{n}_k, x) \in F_{Det}.$$

Thus, we obtained a reduction from A to F_{Det}. □

4 Statistical Representability

In this section we present another weakening of the original concept of FM–representabiliy.[4] Now, we do not require that for all $a_1, \ldots, a_k$ a given formula correctly describes a given relation R on $a_1, \ldots, a_k$. We only want that the description is more likely to be correct than incorrect.

Definition 16. *Let $\varphi(x_1, \ldots, x_k)$ be a formula and $\mathbf{a}$ be a valuation in $\mathbb{N}$. By $\mu_n(\varphi, \mathbf{a})$ we denote*

$$\mu_n(\varphi, \mathbf{a}) = \frac{\mathrm{card}\{\mathcal{A} \in \mathrm{FM}(\mathbb{N}) : \max_{1 \leq i \leq k}\{\mathbf{a}(x_i)\} \leq \mathrm{card}(\mathcal{A}) \leq n \wedge \mathcal{A} \models \varphi[\mathbf{a}]\}}{n}.$$

By $\mu(\varphi, \mathbf{a})$ we denote the limit value of μ_n for $n \to \infty$, if it exists.

$$\mu(\varphi, \mathbf{a}) = \lim_{n \to \infty} \mu_n(\varphi, \mathbf{a}).$$

Since, $\mu(\varphi(x_1, \ldots, x_k), \mathbf{a})$ is determined by values $\mathbf{a}$ on the free variables of φ we write also $\mu(\varphi, a_1, \ldots, a_k)$ with the obvious meaning. If φ is a sentence then the value of $\mu(\varphi, \mathbf{a})$ does not depend on $\mathbf{a}$. In this case we write $\mu(\varphi)$.

Definition 17. *The relation $R \subseteq \omega^r$ is statistically representable if there is a formula $\varphi(x_1, \ldots, x_r)$ with all free variables among $x_1, \ldots, x_r$ such that for all $a_1, \ldots, a_r \in \omega$,*

- *$\mu(\varphi, a_1, \ldots, a_r)$ exists,*
- *if $(a_1, \ldots, a_r) \in R$ then $\mu(\varphi, a_1, \ldots, a_r) > 1/2$*
- *if $(a_1, \ldots, a_r) \notin R$ then $\mu(\varphi, a_1, \ldots, a_r) < 1/2$.*

Theorem 18. *Let $R \subseteq \omega^r$. Then, R is statistically representable if and only if R is* FM*–representable.*

Proof. The implication from right to left is obvious. To prove the converse let us assume that $R \subseteq \omega^r$ is statistically represented by $\varphi(x_1, \ldots, x_r)$. We will give a Σ_2^0 definition of R. Then, since the set of statistically representable relations is obviously closed on the complement, we get that R has to be Δ_2^0. We have the following: for all $a_1, \ldots, a_r \in \omega$,

$$(a_1, \ldots, a_r) \in R \iff \exists N \forall n \geq N \mu_n(\varphi, a_1, \ldots, a_r) > \frac{1}{2}. \qquad (*)$$

The formula on the right side of (*) is Σ_2^0 so it remains to show that it gives a good description of R.

If the right side of (*) holds then of course $\mu(\varphi, a_1, \ldots, a_r)$ is greater or equal $\frac{1}{2}$. But, by the definition of statistical representability, $\mu(\varphi, a_1, \ldots, a_r)$ cannot be equal to $\frac{1}{2}$. Thus,

$$\mu(\varphi, a_1, \ldots, a_r) > \frac{1}{2} \quad \text{and} \quad (a_1, \ldots, a_r) \in R.$$

[4] The results contained in this section are based on [13].

On the other hand, if $(a_1, \ldots, a_r) \in R$ then $\mu(\varphi, a_1, \ldots, a_r) = \frac{1}{2} + \varepsilon$, for some $\varepsilon > 0$. Now, if we choose N in such a way that for all $n \geq N$,

$$|\mu(\varphi, a_1, \ldots, a_r) - \mu_n(\varphi, a_1, \ldots, a_r)| < \frac{\varepsilon}{2}$$

then the right side of (*) holds. □

Definition 19. *The relation $R \subseteq \omega^r$ is weakly statistically representable if there is a formula $\varphi(x_1, \ldots, x_r)$ such that for all $a_1, \ldots, a_r \in \omega$,*

$(a_1, \ldots, a_r) \in R$ *if and only if the value $\mu(\varphi, a_1, \ldots, a_r)$ exists and equals* 1.

Since the statistical representability coincides with FM–representability one could expect that relations which are weakly statisticaly representable are exactly relations which are weakly FM–representable. On the other hand, the quantifier prefix in the expression $\mu(\varphi) = 1$ suggests that these relations are exactly relations which are Π^0_3 in the arithmetical hierarchy. The second guess is correct.

Before we present the theorem we define some auxiliary notions. We write $\sqrt{\mathrm{MAX}} < x$ for the formula $\forall z(xx \neq z)$. We write $\mathrm{Input}(c) = n$ for $\exists e < c\, \mathrm{T}(e, n, c)$ and $x \in W_e$ for $\exists c\, T(e, x, c)$.

Theorem 20. *The family of relations which are weakly statistically representable is exactly the family of Π^0_3 relations in the arithmetical hierarchy.*

Proof. By the method from the proof of Lemma 12, It may be easily shown that the family of weakly statistically representable relations is closed on many one reduction. Thus, it suffices to show that a Π^0_3–complete set is in this family. We take the Π^0_3–complete set coInf of Turing machines with coinfinite domain:

$$\mathrm{coInf} = \{e : \omega \setminus W_e \text{ is infinite}\}.$$

Now, we write the formula $\varphi(z) :=$

$$\forall n \forall c[\{\sqrt{\mathrm{MAX}} < c \wedge n = \mathrm{Input}(c) \wedge \forall c_1(\sqrt{\mathrm{MAX}} < c_1 \Rightarrow n \leq \mathrm{Input}(c_1)\} \Rightarrow$$
$$\forall x\{([(x \notin W_z \wedge x < n) \vee x = 1] \wedge \forall y((y \notin W_z \wedge y < n) \Rightarrow y \leq x)) \Rightarrow \neg(x|\mathrm{MAX})\}]$$

with the property that for all $e \in \omega$,

$$e \in \mathrm{coInf} \text{ if and only if } \mu(\varphi, e) = 1. \qquad (**)$$

The formula φ in a model on $\{0, \ldots, m-1\}$ looks for a computation c greater than $\sqrt{m-1}$ with the smallest imput n. Then, it takes the greatest $x < n$ which is not an input of any e–computation in the model (or it takes 1 if there is no such a x) and forces its own density close to $1 - 1/x$. If there is no such computation c then φ is simply true. Now, we show (**).

Let us assume that W_e is coinfinite and let $\varepsilon > 1/k$ such that $k \notin W_e$. Let $N = \max\{c^2 : \mathrm{Input}(c) \leq k\} + 1$. We show that for all $m > N$, $|1 - \mu_m(\varphi, e)| < \varepsilon$. In the model $\mathbb{N}_m$ there is no computation c such that $\sqrt{m-1} < c$ and $\mathrm{Input}(c) < k$. Thus, φ forces its density at least to $1 - 1/k$ in models greater than N.

Now, let us assume that W_e is cofinite and let $k = \max(\omega \setminus W_e)$. Let us fix arbitrary large N and $c_0 = \max\{c : \mathrm{Input}(c) \leq N\}$. Starting from $\mathbb{N}_{c_0+1}$ up to $\mathbb{N}_{c_0^2}$, φ forces its density to $1 - 1/k$. It follows that $|1 - \mu_{c_0^2}(\varphi, e)| \geq 1/2k$. □

5 Conclusions

We have investigated some variants and weakenings of the notion of FM–representability. Summarizing we observe that:

1. The notion of FM–representability has a natural characterization in terms of arithmetical definability.
2. It captures in a natural way the idea of a relation which can be in a meaningfull way described in finite but potentially infinite domains.
3. FM–representing formulae can be considered as computing devices finitely deciding some relations. So the notion of FM–representability behaves similarly to recursive decidability. The main difference is that in the former case the halting condition – being still finite – cannot be determined in a single finite model. Let us observe that weak FM–representability corresponds – in this sense – to recursive enumerability.

References

1. Mostowski, M.: On representing concepts in finite models. Mathematical Logic Quarterly **47** (2001) 513–523
2. Aristotle: Physics. The Internet Classics Archive (written 350 B.C.) translated by R. P. Hardie and R. K. Gaye. available at: http://classics.mit.edu/Aristotle/physics.html.
3. Mycielski, J.: Analysis without actual infinity. Journal of Symbolic Logic **46** (1981) 625–633
4. Mostowski, M.: On representing semantics in finite models. In Rojszczak†, A., Cachro, J., Kurczewski, G., eds.: Philosophical Dimensions of Logic and Science, Kluwer Academic Publishers (2003) 15–28
5. Mostowski, M.: Potential infinity and the Church Thesis. in manuscript, see also an extended abstract in the electronic proceedings of Denis Richard 60-th Birthday Conference, Clermont–Rerrand, 2000
6. Mostowski, M.: Truth definitions in finite models. in manuscript (1993)
7. Kołodziejczyk, L.: Truth definitions in finite models. The Journal of Symbolic Logic **69** (2004) 183–200
8. Kołodziejczyk, L.: A finite model-theoretical proof of a property of bounded query classes within PH. The Journal of Symbolic Logic **69** (2004) 1105–1116
9. Ebbinghaus, H.D., Flum, J.: Finite model theory. Springer–Verlag (1995)
10. Soare, R.I.: Recursively enumerable sets and degrees. Perspectives in Mathematical Logic. Springer (1987)
11. Krynicki, M., Zdanowski, K.: Theories of arithmetics in finite models. Journal of Symbolic Logic **70(1)** (2005) 1–28
12. Mostowski, M., Wasilewska, A.: Arithmetic of divisibility in finite models. Mathematical Logic Quarterly **50(2)** (2004) 169–174
13. Zdanowski, K.: Arithmetics in finite but potentially infinite worlds. PhD thesis, Warsaw University (2005) in preparation.

Formalising Exact Arithmetic in Type Theory

Milad Niqui

Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands
milad@cs.ru.nl

Abstract. In this work we focus on a formalisation of the algorithms of lazy exact arithmetic à la Potts and Edalat [1]. We choose the constructive type theory as our formal verification tool. We discuss an extension of the constructive type theory with coinductive types that enables one to formalise and reason about the infinite objects. We show examples of how infinite objects such as streams and expression trees can be formalised as coinductive types. We study the type theoretic notion of *productivity* which ensures the infiniteness of the outcome of the algorithms on infinite objects. Syntactical methods are not always strong enough to ensure the productivity. However, if some information about the complexity of a function is provided, one may be able to show the productivity of that function. In the case of the normalisation algorithm we show that such information can be obtained from the choice of real number representation that is used to represent the input and the output.

1 Introduction

In the exact arithmetic approach to numerical computations the emphasis lies on the precision of the outcome of the computation. Due to this the algorithms of exact arithmetic are suitable objects for applying formal methods. Both of the fields of exact arithmetic and formal methods benefit from the verification of exact arithmetic by means of formal tools such as theorem provers or proof assistants. On one hand such formalisations often expose some rarely considered subtleties in the object of the formalisation. On the other hand formalising exact arithmetic provides a serious test to assess the expressiveness of the formal framework that is chosen for the formalisation. This is especially the case if one uses the approaches to the exact arithmetic whose formalisation require a complex type system, such as the lazy exact arithmetic à la Potts and Edalat [1, 2].

Formalising Potts and Edalat's algorithms requires a type theory that is enriched with infinite objects. This is because the Normalisation Algorithm (**NAlg**) — which is the core of Potts and Edalat's lazy exact arithmetic— has infinite objects as its input and output, namely streams and expression trees. Streams and expression trees are both relatively simple infinite objects[1] and therefore

[1] This is because the corresponding categorical objects — final coalgebras— are generated by 'simple' functors, namely polynomial functors [3–§ 4.2].

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 368–377, 2005.

they can be formalised as function types in simple type theories. However, such formalisations require modifying **NAlg** in order to fit in the said type system. Usually this means that the formalised version of **NAlg** is not a *lazy* algorithm anymore; rather, it performs the computation in a sequential manner.

In our work we intend to formalise the **NAlg** while keeping its lazy nature, because in our view laziness is an important characteristic of **NAlg**. Thus in order to formalise these algorithms we choose a type theory extended with coinductive types. More specifically we choose the Calculus of Inductive Constructions (**CIC**) extended with coinductive types [4]. There are two reasons for this choice:

1. **CIC** distinguishes between propositions (computationally irrelevant proofs) and sets (computational content of proofs);
2. **CIC** is implemented as the *Coq* proof assistant [5].

Thus, once formalising the algorithms in *Coq* one can extract programs from the formalisation. This is due to the distinction between propositions and computational contents: one can put all complicated proof objects in the universe of propositions with the knowledge that they will be discarded once the computational content of the formalisation is extracted. Ideally the extracted computational content is identical to the original algorithms. In practice and during the formalisation, in many cases one has to modify the computational content of the original algorithm. From our point of view these changes are acceptable as long as they do not alter the computational complexity of the original algorithm. In other words in order to be able to claim that a *given* algorithm is formalised, the computational complexity of the algorithm and its formalisation should not differ.This is the path that we follow in our formalisation and verification of the **NAlg** algorithm.

2 Coinductive Types

Coinductive types were added to the type theory in order to make it capable of dealing with infinite objects. This extension was done by Hagino [6] using the categorical semantics. The idea is to consider an ambient category for the type theory, and interpret the final coalgebras of this category as coinductive types. The standard way to consider a categorical model for type theory is to interpret types as objects and typing rules as morphisms [7]. But one can also base the presentation of coinductive types on Martin-Löf's constructive type theory. That means that in order to present the rules for a type, one should present the formation, introduction, elimination and equality rules [8]. Here we use this approach to define coinductive types for polynomial functors. Such coinductive types are all that is needed for formalising **NAlg**.

In the Martin-Löf's setting, the formation rule is originally a set formation rule. Since we do not put any constraints on the ambient category $\mathcal{C}$, the formation rule will be modified to the formation of the objects of $\mathcal{C}$. For a symbol t to be a type, it should be an object of $\mathcal{C}$. We fix a distinguished symbol s. Then 't is a type' can be written as $t\colon s$. Furthermore to present the method of

construction, we should assume that we have a description of final coalgebras of endofunctors on $\mathcal{C}$. This means that there exists a partial decision procedure $\mathsf{is}\text{-}\nu\mathrm{F}$, which determines whether F has a final coalgebra. We assume

$$\mathsf{is}\text{-}\nu\mathrm{F} := \mathrm{F} \text{ is polynomial } .$$

There are slightly larger classes of functors for which a total decision procedure exists. Among them are *continuous* functors and functors that correspond to *strictly positive type operators*. Strictly positive type operators and their functorial extensions are used in defining inductive types in **CIC**; their definition can be found e.g. in [9].

Below we shall present the rules of coinductive types. The method of construction will be given as a type constructor symbol ν, with a total decision procedure $\mathsf{is}\text{-}\nu\mathrm{F}$. This method of construction is characterised by the following rules and produces the *coinductive type* νF.

νF-Formation
$$\frac{\mathsf{is}\text{-}\nu\mathrm{F}}{\nu\mathrm{F}\colon s}$$

νF-Introduction
$$\frac{U\colon s \qquad x\colon U \vdash u(x)\colon \mathrm{F}(U)}{y\colon U \vdash (\nu\text{-}\mathsf{it} \quad u \quad y)\colon \nu\mathrm{F}}$$

νF-Elimination
$$\frac{x\colon \nu\mathrm{F}}{(\nu\text{-}\mathsf{out} \quad x)\colon \mathrm{F}(\nu\mathrm{F})}$$

νF-Equality
$$\frac{U\colon s \qquad x\colon U \vdash u(x)\colon \mathrm{F}(U)}{y\colon U \vdash (\nu\text{-}\mathsf{out} \quad (\nu\text{-}\mathsf{it} \quad u \quad y)) = (\mathrm{F}(\nu\text{-}\mathsf{it} \quad u) \quad u(y))}$$

The rule νF-Equality speaks about the *intensional* equalities between terms. This means that it can be considered as a reduction or conversion rule. This way we can present the following rule.

νF-Reduction
$$(\nu\text{-}\mathsf{out} \quad (\nu\text{-}\mathsf{it} \quad u \quad y)) \rightsquigarrow (\mathrm{F}(\nu\text{-}\mathsf{it} \quad u) \quad u(y))$$

The rule νF-Introduction is inspired by the *coiteration scheme* of the final coalgebra of F. There are alternative schemes that can be used for defining functions into a final coalgebra, such as the corecursion scheme or the dual of course-of-value iteration scheme [3–§ 4.4].

At this point we are able to use ν-Formation to introduce the coinductive types. Evidently we only need to know whether the given functor is polynomial. This can be done if we present the functor by means of its *constructors* (for a definition see [3–p. 99]). In fact all we need for defining a coinductive type is the type of its constructors and not the constructors themselves. This approach is taken in *Coq* for defining coinductive types [4]. Consequently, one can use the remaining rules for defining functions on infinite objects and proving the properties of these function.

3 Streams and Expression Trees

We introduce the coinductive types that are used in **NAlg**. Let A, B be two sets and define the following functors.

$$\mathrm{F}_1(X) = A \times X \ \ , \mathrm{F}_2(X) = A \times X + B \times X^2 \ \ .$$

The above functors have final coalgebras. The final coalgebra of F_1 is just the set A^ω of *stream* of elements of A. It is well-known that the set of streams with as destructor the map $\langle \mathsf{hd}, \mathsf{tl} \rangle \colon A^\omega \to A \times A^\omega$ is a final coalgebra.

The final coalgebra of F_2 is the set of $\mathbb{E}(A, B)$ of *expression trees with A-elements and B-operations*. These are infinite binary trees in which every node has one or two children. A unary node is an element of A and a binary node is an element of B. The proof of finality of $\mathbb{E}(A, B)$ can be found in [3–Example 4.2.4]. The destructor is the following map.

$$\mathsf{psupp}_{\mathsf{A},\mathsf{B}}(\theta) := \begin{cases} \mathsf{inl}(\langle a,\, \theta' \rangle) & \textbf{if } \theta \text{ has root } a \text{ and child } \theta' \ , \\ \mathsf{inr}(\langle b,\, \theta',\, \theta'' \rangle) & \textbf{if } \theta \text{ has root } b \text{ and children } \theta', \theta'' \ . \end{cases}$$

Using this, one can define the constructors of this final coalgebra, but here we give (and need) only the type of the constructors:

$$\begin{aligned} &\mathsf{ucons} \colon A \times \mathbb{E}(A, B) \to \mathbb{E}(A, B), \\ &\mathsf{bcons} \colon B \times \mathbb{E}(A, B)^2 \to \mathbb{E}(A, B) \ \ . \end{aligned}$$

We write $\langle\!\langle a;\, \theta \rangle\!\rangle$ and $\langle\!\langle b;\, \theta,\, \theta' \rangle\!\rangle$ respectively for $\mathsf{ucons}(a, \theta)$ and $\mathsf{bcons}(b, \theta, \theta')$. Similarly the constructor of the final coalgebras of streams of A is $\mathsf{cons} \colon A \to A^\omega$. We write $a \colon \alpha$ for $\mathsf{cons}(a, \alpha)$.

Thus, by presenting the above constructors we can form coinductive types of streams and expression trees using $\nu\mathrm{F}_1$-Formation and $\nu\mathrm{F}_2$-Formation rules.

4 The Normalisation Algorithm

We can present **NAlg** [10] using the constructors for expression trees and streams. For this let $\mathbb{R}^+ = (0, +\infty)$ and let $\mathbb{M}$ (resp. $\mathbb{T}$) be the set of Möbius maps (resp. $2 \times 2 \times 2$ tensors over $\mathbb{Z}$) which are refining on $\mathbb{R}^\star := \mathbb{R}^+ \cup \{\, 0, +\infty \,\}$[2]. Let Φ be a *digit set*, i.e., a finite set of refining Möbius maps such that there is a total surjective map (a *representation*) ρ from Φ^ω to $\mathbb{R}^\star$ and that for all $f_0 f_1 \cdots \in \Phi^\omega$ we have

$$\{\, \rho(f_0 f_1 \ldots) \,\} = \bigcap_{i=0}^{\infty} f_0 \circ \ldots \circ f_i(\mathbb{R}^\star) \ \ .$$

[2] Instead of $\mathbb{R}^\star$, we could pick any proper closed subinterval of $\mathbb{R} \cup \{\, -\infty, +\infty \,\}$ [3–Chap. 5].

By $\xi\ \langle x, y\rangle$ we denote the application of the tensor ξ considered as a quadratic map to the points $x, y \in \mathbb{R}^\star$. Let $\phi \in \Phi$. For each $\mu \in \mathbb{M}$ (resp. $\xi \in \mathbb{T}$) we define the *emission condition* as the predicate

$$\begin{aligned}\mathbf{Incl}(\mu, \phi) &:= \mu(\mathbb{R}^+) \subseteq \phi(\mathbb{R}^+)\ ,\\ \mathbf{Incl}(\xi, \phi) &:= \xi\langle\mathbb{R}^+, \mathbb{R}^+\rangle \subseteq \phi(\mathbb{R}^+)\ .\end{aligned}$$

Furthermore we define *left product* (denoted by $\bullet_1$) and *right product* (denoted by $\bullet_2$) of a tensor and a Möbius map as follows.

$$\begin{bmatrix} a & b & c & d\\ e & f & g & h\end{bmatrix} \bullet_1 \begin{bmatrix} A & B\\ C & D\end{bmatrix} = \begin{bmatrix} aA + cC & bA + dC & aB + cD & bB + dD\\ eA + gC & fA + hC & eB + gD & fB + hD\end{bmatrix}\ ,$$

$$\begin{bmatrix} a & b & c & d\\ e & f & g & h\end{bmatrix} \bullet_2 \begin{bmatrix} A & B\\ C & D\end{bmatrix} = \begin{bmatrix} aA + bC & aB + bD & cA + dC & cB + dD\\ eA + fC & eB + fD & gA + hC & gB + hD\end{bmatrix}\ .$$

Finally we are able to present the **NAlg** in a functional from:

$$\mathsf{nf}_\Phi\langle\!\langle \mu;\, \theta\rangle\!\rangle := \begin{cases} \phi\colon \mathsf{nf}_\Phi\langle\!\langle \phi^{-1} \circ \mu;\, \theta\rangle\!\rangle & \textbf{if } \exists \phi \in \Phi,\ \mathbf{Incl}(\mu, \phi)\ ,\\ \mathsf{nf}_\Phi\langle\!\langle \mu \circ \mathsf{hd}\,(\mathsf{nf}_\Phi(\theta));\, \mathsf{tl}\,(\mathsf{nf}_\Phi(\theta))\rangle\!\rangle & \textbf{otherwise}.\end{cases}$$

$$\mathsf{nf}_\Phi\langle\!\langle \xi;\, \theta_1,\, \theta_2\rangle\!\rangle := \begin{cases} \phi\colon \mathsf{nf}_\Phi\langle\!\langle \phi^{-1} \circ \xi;\, \theta_1,\, \theta_2\rangle\!\rangle & \textbf{if } \exists \phi \in \Phi,\ \mathbf{Incl}(\xi, \phi)\ ,\\ \mathsf{nf}_\Phi\langle\!\langle \xi \bullet_1 \mathsf{hd}\,(\mathsf{nf}_\Phi(\theta_1)) \bullet_2 \mathsf{uhd}(\mathsf{nf}_\Phi(\theta_2)) & \\ \qquad ;\, \mathsf{tl}\,(\mathsf{nf}_\Phi(\theta_1)),\, \mathsf{tl}\,(\mathsf{nf}_\Phi(\theta_2))\rangle\!\rangle & \textbf{otherwise}.\end{cases}$$

One can observe that the **NAlg** as presented above is a composition of continuous functions on suitably chosen dcpos. Therefore, being a continuous functional, it has a fixpoint [3–§ 4.5]. Thus the above definition is meaningful and indeed it can be written as such in a functional programming language. But the functions in the type theory should be introduced using the introduction rules of the type theory. Thus **NAlg** can not be formalised as above in the type theory of **CIC** extended with coinductive types.

First, there is a type checking problem in the above presentation of **NAlg**. According to the type of input and output, the type of **NAlg** should be $\mathbb{E} \to \Phi^\omega$. This means that we should define the matrix multiplication (and left and right products) between a digit (which is a refining Möbius map) and a refining Möbius map which is a unary node of an expression tree. Note that in **CIC** there is no subtyping by inheritance. Hence we should coercive subtyping, i.e., we should define a map between Φ and $\mathbb{M}$ and declare it to be a *coercion* [11]. This also solves the problem that in the nested branches the argument of **NAlg** is an expression tree with streams as children.

The second problem with formalising the above presentation of **NAlg** is more difficult to tackle and corresponds to the productivity of **NAlg**, which we discuss in the next section.

5 Productivity

If we formalise **NAlg** in our type theory, we should only use the rules of **CIC** plus those presented in Sect. 2. More specifically, the rule ν-Introduction should be used to define infinite objects. This is the rule which builds the stream that is the outcome of **NAlg**. This rule is based on the coiteration scheme and hence it can only be used to define infinite objects whose construction does not involve an unbounded search. This also would be the case had we chosen more complex schemes of corecursion or the dual of course-of-value recursion to propose an alternative introduction rule. But one can see that the nested branches of **NAlg** require an unbounded search: they absorb from the input expression tree and modify the input until the condition $\mathbf{Incl}(-,-)$ holds.

This means that **NAlg** might not produce output for some inputs, i.e., it might be a partial function. If a function on infinite objects, produces an output for a given input, we say that this function is *productive* for that input. Productivity is dual to the notion of *termination* which is used to ensure the totality of recursive functions.

One can give a more formal definition of productivity using the theory of partially ordered sets. Let $[A]$ be the pointed dcpo of partial lists and streams over set A as defined in [3–p. 111]. By *partial lists* we refer to the finite lists of elements of $A \cup \{\perp_A\}$, where $\perp_A$ is a fresh element added to A. Let $[A;B]$ be the pointed dcpo of partial expression trees with A-elements and B-operations, as defined in [3–p. 114].

Definition 1. *For nonempty sets A and B, in the dcpos $[A]$ and $[A;B]$ we shall call each maximal element a* productive *element.*

For finite partial lists we use the dual term terminative*. A finite partial list σ in $[A]$ is terminative if and only if*

$$\forall 0 \leq i < \mathsf{length}\,(\sigma),\ \sigma(i) \text{ is maximal.}$$

Note that the empty partial list is vacuously terminative.

We call a function between two pointed dcpos productive*, if it takes productive elements to productive elements.*

It is immediate from this definition that a maximal element of the function space of standard dcpos is productive. On the other hand, in the function space, productivity is a weaker notion than maximality [3–Example 4.6.3]. Furthermore it is immediate from the above definition that composition of two productive functions is productive.

It is not always easy to directly verify the productivity of functions. For the case of productive streams there is a criterion which is usually easier to apply than the direct application of the above definition [12–Theorem 32]. Our goal is to present a similar criterion for functions on streams. As we are mainly interested in productivity of fixpoints, we will use the following equation which holds in every pointed dcpo (μf denotes the fixpoint of f).

$$\mu f = \bigsqcup_{n\in\mathbb{N}}^{\uparrow} f^n(\perp)$$

In what follows we assume D and A to be two pointed dcpos with F a continuous functional from $D \longrightarrow [A]$ to $D \longrightarrow [A]$. We denote $F^j(\bot_{D \longrightarrow [A]})$ by f^j which is a function from D to $[A]$. Note that because of monotonicity of F if $i \leq j$ then $f^i \sqsubseteq f^j$. By $(\alpha)_j$ we denote the list consisting of the first j elements of α. We introduce a predicate which we use as a criterion for productivity of fixpoints.

Definition 2. *For $d \in D$ we define the predicate* $\mathsf{cont}(F, d)$ *as*

$$\mathsf{cont}(F, d) := \forall j \geq 0,\ \exists k \geq 0,\ \ (f^k(d))_j \text{ is terminative.}$$

If $\mathsf{cont}(F, d)$*, then we call F to be* terminative at d*. We define the predicate* $\mathsf{Cont}(F)$ *as the global version of* **cont** *on all maximal elements, i.e.,*

$$\mathsf{Cont}(F) := (\forall d \in D,\ d \text{ is maximal } \implies \mathsf{cont}(F, d))\ .$$

If $\mathsf{Cont}(F)$ *holds, then we shall say that F is* terminative*.*

Proposition 3. *If F is terminative then μF is productive.*

The converse of the above proposition does not hold in general. However, the converse result holds if we restrict ourselves to dcpos in which the maximal elements are accessible.

Definition 4. *We call an element d of a pointed dcpo* accessible*, if there are finitely many elements strictly between d and $\bot$.*

Proposition 5. *Let D be a dcpo in which maximal elements are accessible and assume that μF is productive. Then F is terminative.*

6 Productivity of NAlg

We will show that the **NAlg** is productive on some families of expression trees. First note that the behaviour of the **NAlg** is dependent of the digit set that is chosen for representing the output. This is because each non-nested branch of **NAlg** emits an output when the property $\mathbf{Incl}(-, \phi)$ holds, which is basically a topological property of the chosen digit set. We base our work on representations that are *admissible* (cf. [13]), because it is known tat such representations have enough redundancy to be used for computing real computable functions such as elementary functions [13].

Let $\mathbf{S}(x) = (x-1)/(x+1)$ and given an interval $[x, y] \subseteq \mathbb{R}^\star$, let $\mathbf{diam}([x, y]) = |\mathbf{S}(x) - \mathbf{S}(y)|$.

Definition 6. *Let Φ be a finite set of refining increasing Möbius maps. We call Φ an* admissible digit set for $\mathbb{R}^\star$ *if both following conditions hold.*

1. $\lim_{j \to \infty} \max\{\mathbf{diam}(\phi_0 \circ \phi_1 \circ \ldots \circ \phi_{k-1}(\mathbb{R}^\star)) \mid \phi_0, \ldots, \phi_{k-1} \in \Phi\} = 0$;
2. $\bigcup_{\phi_i \in \Phi} \phi_i(\mathbb{R}^+) = \mathbb{R}^+$.

It is immediate that an admissible digit set is a digit set. Therefore there exists a $\mathbf{Rep}_{\Phi}\colon \Phi^{\omega} \longrightarrow \mathbb{R}^{\star}$ such that $\bigcap_{i=0}^{\infty} \phi_0 \circ \ldots \circ \phi_i(\mathbb{R}^{\star}) = \{\, \mathbf{Rep}_{\Phi}(\phi_0\phi_1 \ldots) \,\}$. The following proposition justifies the above definition. The proof can be found in [3–Theorem 5.4.9])

Proposition 7. *Let Φ be an admissible digit set. Then $\mathbf{Rep}_{\Phi}$ is an admissible representation.*

We study productivity on the families defined below.

Definition 8. *Let $\theta \in [\mathbb{M};\mathbb{T}]$. Then θ is* open*, if for every $j > 0$ the initial segment of length j of each branch of θ is a terminative list.*

Let $\alpha \in \Phi^{\omega}$. For an open expression tree θ we define $\theta{+\!\!+}\alpha$ to be the productive expression tree which is obtained by concatenating finite branches of θ with α.

Definition 9. *Let $\dot{\theta}$ be an open expression tree. Then we define the* family of expression trees specified by $\dot{\theta}$ *to be the set $\Theta(\dot{\theta}) = \{\dot{\theta}{+\!\!+}\alpha \mid \alpha \in \Phi^{\omega}\}$*

The next thing we need is a basic quantity which characterises the redundancy of an admissible digit set.

Definition 10. *Let Φ be an admissible digit set. We define the* redundancy *of Φ as*

$$\mathsf{red}(\Phi) = \min\{|\mathbf{S}(\phi_i(0)) - \mathbf{S}(\phi_j(+\infty))| \mid \phi_i, \phi_j \in \Phi,\ \phi_i(0) \neq \phi_j(+\infty)\}.$$

Using the the following important property of redundancy we can show that the emission condition holds after finitely many absorption steps (nested branches).

Proposition 11. *Let $\mu \in \mathbb{M}$ be such that $\mathbf{diam}(\mu(\mathbb{R}^{\star})) < \mathsf{red}(\Phi)$. Then there exists $\phi_i \in \Phi$ such that $\mathbf{Incl}(\mu, \phi_{\mathrm{i}})$.*

To state a similar result for the tensor maps note that if I_1, I_2 are two closed intervals and $\xi \in \mathbb{T}$, then $\xi\langle I_1,\, I_2\rangle$ is a closed interval.

Proposition 12. *Let $\xi \in \mathbb{T}$ such that $\mathbf{diam}(\xi\langle \mathbb{R}^{\star}, \mathbb{R}^{\star}\rangle) < \mathsf{red}(\Phi)$. Then there exists $\phi_i \in \Phi$ such that $\mathbf{Incl}(\xi, \phi_{\mathrm{i}})$.*

The Propositions 11 and 12 give us enough information about the complexity of **NAlg**. Applying these and Proposition 3 we obtain the following.

Proposition 13. *Let Φ be an admissible digits sets. Then for any accessible open tree $\dot{\theta}$,* **NAlg** *is productive on $\Theta(\dot{\theta})$.*

7 Formalisation in *Coq*

The Proposition 13 ensures that **NAlg** should be formalisable as a partial function in our type theory. From the proof of this proposition one can extract a

function $\varkappa\colon \mathbb{E} \longrightarrow \Phi \times \mathbb{E}$ that given an expression tree θ, outputs a digit ϕ_i and an expression tree θ'. The digit ϕ_i is the next digit that would be emitted by **NAlg** with the input θ, and the θ' would be the rest of the input (or the new input) after emitting ϕ_i. Subsequently, the function $\varkappa$ can be used to rewrite **NAlg** as

$$\mathsf{nf}_\Phi(\theta) := \psi\colon \mathsf{nf}_\Phi(\theta') \qquad \textbf{where } \langle \psi, \theta' \rangle = \varkappa(\theta) \ .$$

This shape of **NAlg** can be generated using the ν-Introduction rule. That is to say, for $\theta \in \mathbb{E}$ we define $\mathbf{NAlg}(\theta) := (\nu\text{-}\mathsf{it} \quad \varkappa \quad \theta)$. Because of the way the function $\varkappa$ is defined, this change will not alter the complexity of **NAlg**.

Practically, when we are using the *Coq* proof assistant the modification of **NAlg** using the function $\varkappa$ will also help in satisfying the *guardedness condition*. This condition is a syntactic criterion added to the *Coq* type checker in order to ensure the productivity of infinite objects [4]. The guardedness condition is not very powerful and can only be used for formalising infinite objects that have a canonical shape: the recursive occurrence of the infinite object is the immediate (i.e., unguarded) argument of the constructor of the coinductive type of the codomain. Fortunately, the function $\varkappa$ enables us to satisfy this condition.

Note that in the present paper we have only dealt with *formalising* the **NAlg** in type theory. That means we tried to write it in the language of our type theory. After formalising the algorithm we need to verify its correctness. This involves proving that the formalised algorithm satisfies its specifications. In the case of **NAlg** we should prove that for various choices of families of expression trees as given in [10], this algorithm computes the respective elementary function. This part of work, although mathematical in nature, involves some type theoretic issues and deserves a separate treatment.

8 Further Work

Several extensions to the present work are imaginable. First of all one should try to extend the Proposition 13 to include the families of expression trees as large as possible. This is closely related to assessing the strength of the notion of $\mathbb{E}$-computability which was presented in [3–§ 5.6].

Another ongoing project is to formalise **NAlg** in an alternative way. The idea is to add a proof obligation to the type theoretic version of **NAlg** as an extra argument. This argument would have as type a coinductively defined predicate that states that the input expression tree will emit infinitely many times. Consequently, this can be used in combination with a non-structurally recursive function that behaves as the function $\varkappa$ but is defined on an inductively defined domain. This latter inductively defined domain is also a predicate and it has a constructor for each branch of **NAlg**. The advantage of this method is that it uses the separation between computationally important objects and non-informative objects (predicates) of **CIC**.

References

1. Potts, P.J., Edalat, A.: Exact real computer arithmetic. Technical Report DOC 97/9, Department of Computing, Imperial College (1997)
2. Edalat, A., Potts, P.J., Sünderhauf, P.: Lazy computation with exact real numbers. In Berman, M., Berman, S., eds.: Proceedings of the third ACM SIGPLAN International Conference on Functional Programming (ICFP-98), Baltimore, Maryland, USA, September 27-29, 1998. Volume 34(1) of ACM SIGPLAN Notices., New York, ACM Press (1999) 185–194
3. Niqui, M.: Formalising Exact Arithmetic: Representations, Algorithms and Proofs. PhD thesis, Radboud Universiteit Nijmegen (2004)
4. Giménez, E.: Un Calcul de Constructions Infinies et son Application a la Verification des Systemes Communicants. PhD thesis PhD 96-11, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon (1996)
5. The Coq Development Team: The Coq Proof Assistant Reference Manual, Version 8.0. INRIA. (2004) `http://coq.inria.fr/doc/main.html`, [cited 31st January 2005].
6. Hagino, T.: A Categorical Programming Language. PhD thesis CST-47-87, Laboratory for Foundations of Computer Science, Dept. of Computer Science, Univ. of Edinburgh (1987)
7. Jacobs, B.: Categorical Logic and Type Theory. Volume 141 of Studies in Logic and the Foundations of Mathematics. Elsevier Science Publishers, Amsterdam (1998)
8. Martin-Löf, P.: Intuitionistic Type Theory. Biblioplois, Napoli (1984) Notes of Giovanni Sambin on a series of lectures given in Padova.
9. Coquand, T., Paulin, C.: Inductively defined types (preliminary version). In Martin-Löf, P., Mints, G., eds.: COLOG-88, International Conference on Computer Logic, Tallinn, USSR, December 1988, Proceedings. Volume 417 of Lecture Notes in Comput. Sci. Springer-Verlag (1990) 50–66
10. Potts, P.J.: Exact Real Arithmetic using Möbius Transformations. PhD thesis, University of London, Imperial College (1998)
11. Saïbi, A.: Typing algorithm in type theory with inheritance. In: POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM SIGACT and SIGPLAN, ACM Press (1997) 292–301
12. Sijtsma, B.A.: On the productivity of recursive list definitions. ACM Trans. Program. Lang. Syst. (TOPLAS) **11** (1989) 633–649
13. Weihrauch, K.: Computable Analysis. An introduction. Springer-Verlag, Berlin Heidelberg (2000) 285 pp.

Complexity in Predicative Arithmetic

Geoffrey E. Ostrin[1] and Stan S. Wainer[2]

[1] Institut für Informatik und angewandte Mathematik,
Neubrückstrasse 10,
CH-3012 Bern, Switzerland
geoff@iam.unibe.ch
[2] Department of Pure Mathematics,
University of Leeds, Leeds, LS2 9JT, UK
s.s.wainer@leeds.ac.uk

Abstract. Complexity classes between Grzegorczyk's E^2 and E^3 are characterized in terms of provable recursion in a theory EA(I;O) formalising basic principles of Nelson's Predicative Arithmetic. Extensions by inductive definitions enable full arithmetic PA and higher systems to be recaptured in a setting where the natural bounding functions are "slow" rather than "fast" growing.

Keywords: provable recursion, ordinal analysis, slow growing hierarchy.

1 Introduction

By incorporating the normal/safe variable separation of Bellantoni and Cook [1] into a proof–theoretic setting, results of Leivant [3] are reworked and extended. The resulting theory EA(I;O) formalizes basic principles of Nelson's Predicative Arithmetic [4], and provides a natural "slow growing" analogue of full arithmetic, with the strength of $I\Delta_0(\exp)$. The predicative induction principle is

$$A(0) \wedge \forall a(A(a) \to A(a+1)) \to A(x)$$

where the (normal) "input" x is a numerical constant, and the (safe) "output" variables a range over values defined, or computed, from the given inputs. This induction is weak since one cannot universally quantify over x once it has been introduced. Complexity classes between Grzegorczyk's E^2 and E^3 are characterized by the logical complexity of induction formulas A (for further detail see Ostrin and Wainer [5]). Recent work of the second author with Williams [6] shows how full arithmetic PA is recaptured by adding inductive definitions to EA(I;O).

2 The Theory EA(I;O)

There will be two kinds of individuals: "input" parameters (constants) denoted $x, y, z, \ldots$, and "output" variables denoted $a, b, c, \ldots$, both intended as ranging

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 378–384, 2005.

over natural numbers. The *basic terms* are: variables, input parameters, the constant 0, or the result of repeated application of the successor S or predecessor P. General *terms* are built up in the usual way from the constants and variables, by application of S, P and arbitrary function symbols $f, g, h, \ldots$ denoting partial recursive functions given by sets E of Herbrand-Gödel-Kleene-style defining equations. $t\downarrow$ is shorthand for $\exists a(t \simeq a)$.

Atomic formulas will be equations $t_1 \simeq t_2$ between arbitrary terms, and formulas $A, B, \ldots$ are built from these by applying propositional connectives and quantifiers $\exists a$, $\forall a$ over output variables.

It will be convenient, for later proof theoretic analysis, to work with minimal logic in a sequent-style formalism. This is computationally more natural, and it is not a restriction for us, since a classical proof of $f(x)\downarrow$ can be transformed, by the double-negation interpretation, into a proof in minimal logic.

The derivation–rules are quite standard in appearance. However there is a crucial restriction on the right-$\exists$ and left-$\forall$ rules:

$$\frac{\Gamma \vdash A(t)}{\Gamma \vdash \exists a A(a)} \qquad \frac{\Gamma,\ \forall a A(a),\ A(t) \vdash B}{\Gamma,\ \forall a A(a) \vdash B}$$

namely, the term t must only be a *basic term*.

The logical axioms are, with A atomic, $\Gamma,\ A \vdash A$ and the equality axioms are $\Gamma \vdash t \simeq t$ and, again with A atomic, $\Gamma,\ t_1 \simeq t_2,\ A(t_1) \vdash A(t_2)$. The logic allows these to be generalised straightforwardly to an arbitrary formula A and the quantifier rules allow one to derive immediately

$$\Gamma,\ t\downarrow,\ A(t) \vdash \exists a A(a) \quad \text{and} \quad \Gamma,\ t\downarrow,\ \forall a A(a) \vdash A(t)\ .$$

Thus witnessing terms must be provably defined.

Two further principles are needed, describing the data-type N, namely induction and cases (a number is either zero or a successor). We present these as rules rather than their equivalent axioms, since this will afford a closer match between proofs and computations. The predicative induction rule (with "induction formula" $A(.)$) is

$$\frac{\Gamma \vdash A(0) \quad \Gamma,\ A(a) \vdash A(Sa)}{\Gamma \vdash A(x)}$$

where the output variable a is not free in Γ and where, in the conclusion, x is an input, or a basic term on an input.

The cases rule is

$$\frac{\Gamma \vdash A(0) \quad \Gamma \vdash A(Sa)}{\Gamma \vdash A(t)}$$

where t is any basic term. Note that with this rule it is easy to derive $\forall a(a \simeq 0 \vee a \simeq S(Pa))$ from the definition: $P(0) \simeq 0$ and $P(Sa) \simeq a$.

Definition. Σ_1 formulas are those of the form $\exists \boldsymbol{a} A(\boldsymbol{a})$ where A is a conjunction of atomic formulas and $\boldsymbol{a} = a_1, \ldots, a_k$. A typical example is $f(\boldsymbol{x})\downarrow$.

Definition. A k-ary function f is *provably recursive* in EA(I;O) if it can be defined by a system E of equations such that, with input parameters $x_1, \ldots, x_k$,

$$\bar{E} \vdash f(x_1, \ldots, x_k) \downarrow$$

where $\bar{E}$ denotes the universal closures of the defining equations in E.

2.1 Elementary Functions Are Provably Recursive

Let E be a system of defining equations containing the usual primitive recursions for addition and multiplication, and equations defining polynomials p. Extend E further by adding definitions of iterated exponentials $2_k(p(\boldsymbol{x}))$ where $2_0(p) = p$ and $2_{i+1}(p) = 2^{2_i(p)}$.

Definition. The *progressiveness* of a formula $A(a)$ with distinguished free variable a, is expressed by the formula

$$\mathrm{Prog}_a A \equiv A(0) \wedge \forall a(A(a) \to A(Sa))$$

thus the induction principle of EA(I;O) is equivalent to $\mathrm{Prog}_a A \vdash A(x)$.

The following result gives extensions of this principle to any finitely iterated exponential. A consequence is that every elementary function (i.e. computable in a number of steps bounded by some iterated exponential) is provably recursive in EA(I;O). In the next section we shall see that this is the most that EA(I;O) can do.

Theorem 1. *In EA(I;O) we can prove, for each k and any formula $A(a)$,*

$$\bar{E}, \mathrm{Prog}_a A \vdash A(2_k(p(\boldsymbol{x}))).$$

3 Provably Recursive Functions Are Elementary

For each fixed number k, we inductively generate an infinitary system of sequents

$$E,\ n : N,\ \Gamma \vdash^{\alpha} A$$

where (i) E is a (consistent) set of Herbrand-Gödel-Kleene defining equations for partial recursive functions $f, g, h, \ldots$; (ii) n is a numeral; (iii) A is a closed formula, and Γ a finite multiset of closed formulas, built up from atomic equations between arbitrary terms t involving the function symbols of E; and (iv) α, β denote ordinal bounds which we shall be more specific about later (for the time being think of β as being smaller than α "with respect to projection at k").

Note that we do not explicitly display the parameter k in the sequents below, but if we later need to do this we shall insert an additional declaration $k : I$ in the antecedent thus:

$$E,\ k : I,\ n : N,\ \Gamma \vdash^{\alpha} A\,.$$

Intuitively, k will be a bound on the heights of "unravelled inductions".

The first two rules are just the input and substitution axioms of the equation calculus, the next two are computation rules for N, and the rest are essentially just formalised versions of the truth definition, with Cut added.

E1 $E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ e(\boldsymbol{n})$ where e is either one of the defining equations of E or an identity $t \simeq t$, and $e(\boldsymbol{n})$ denotes the result of substituting, for its variables, numerals for numbers $\boldsymbol{n} = n_1, \ldots, n_r \leq n$.

E2 $E,\ n : N,\ \Gamma,\ t_1 \simeq t_2,\ e(t_1)\ \vdash^{\alpha}\ e(t_2)$ where $e(t_1)$ is an equation between terms in the language of E, with t_1 occurring as a subterm, and $e(t_2)$ is the result of replacing an occurrence of t_1 by t_2.

N1

$$E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ m : N \ \text{ provided } m \leq n + 1$$

N2

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ n' : N \qquad E,\ n' : N,\ \Gamma\ \vdash^{\beta'}\ A}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ A}$$

Cut

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ C \qquad E,\ n : N,\ \Gamma,\ C\ \vdash^{\beta'}\ A}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ A}$$

∃L

$$\frac{E,\ \max(n, i) : N,\ \Gamma,\ B(i)\ \vdash^{\beta_i}\ A \ \text{ for every } i \in N}{E,\ n : N,\ \Gamma,\ \exists b B(b)\ \vdash^{\alpha}\ A}$$

∃R

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ m : N \qquad E,\ n : N,\ \Gamma\ \vdash^{\beta'}\ A(m)}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ \exists a A(a)}$$

∀L

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ m : N \qquad E,\ n : N,\ \Gamma,\ \forall b B(b),\ B(m)\ \vdash^{\beta'}\ A}{E,\ n : N,\ \Gamma,\ \forall b B(b)\ \vdash^{\alpha}\ A}$$

∀R

$$\frac{E,\ \max(n, i) : N,\ \Gamma\ \vdash^{\beta_i}\ A(i) \ \text{ for every } i \in N}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ \forall a A(a)}$$

In addition, there are of course two rules for each propositional symbol, but it is not necessary to list them since they are quite standard. However it should be noted that the rules essentially mimic the truth definition for arithmetic. They provide a system within which inductive proofs in EA(I;O) can be unravelled in a uniform way.

Ordinal Assignment à la Buchholz [2]

The ordinal bounds on sequents above are intensional, "tree ordinals", generated inductively by: 0 is a tree ordinal; if α is a tree ordinal so is $\alpha + 1$; and if $\lambda_0, \lambda_1, \lambda_2, \ldots$ is an ω-sequence of tree ordinals then the function $i \mapsto \lambda_i$ denoted $\lambda = \sup \lambda_i$, is itself also a tree ordinal. Thus tree ordinals carry a specific choice of fundamental sequence to each "limit" encountered in their build-up, and because

of this the usual definitions of primitive recursive functions lift easily to tree ordinals. For example exponentiation is defined by:

$$2^0 = 1, \quad 2^{\beta+1} = 2^\beta + 2^\beta, \quad 2^\lambda = \sup 2^{\lambda_i} \ .$$

For ω we choose the specific fundamental sequence $\omega = \sup(i+1)$. For ε_0 we choose the fundamental sequence $\varepsilon_0 = \sup 2_i(\omega^2)$.

Definitions. For each integer i there is a predecessor function given by:

$$P_i(0) = 0, \quad P_i(\alpha+1) = \alpha, \quad P_i(\lambda) = P_i(\lambda_i)$$

and by iterating P_i we obtain, for each non-zero tree ordinal α, the finite set $\alpha[i]$ of all its "i-predecessors" thus:

$$\alpha[i] = \{P_i(\alpha), P_i^2(\alpha), P_i^3(\alpha), \ldots, 0\}.$$

The *Slow Growing Hierarchy* is given by: $G(\alpha, n) =$ the cardinality of $\alpha[n]$.

Call a tree ordinal α "structured" if every sub-tree ordinal of the form $\lambda = \sup \lambda_i$ (occurring in the build-up of α) has the property that $\lambda_i \in \lambda[i+1]$ for all i. Then if α is structured, $\alpha[i] \subset \alpha[i+1]$ for all i, and each of its sub-tree ordinals β appears in one, and all succeeding, $\alpha[i]$. Thus we can think of a structured α as the directed union of its finite sub-orderings $\alpha[i]$. The basic example is $\omega[i] = \{0, 1, \ldots, i\}$. All tree ordinals used here will be structured ones.

Ordinal Bounds. The condition on ordinal bounds in the above sequents is to be as follows:

- In rules E1, E2, N1, the bound α is arbitrary.
- In all other rules, the ordinal bounds on the premises are governed by $\beta, \beta', \beta_i \in \alpha[k]$ where k is the fixed parameter.

Lemma 2. *(Embedding) If $\bar{E} \vdash f(\boldsymbol{x}) \downarrow$ in EA(I;O) there is a fixed number d determined by this derivation, such that: for all inputs $\boldsymbol{n}$ of binary length $\leq k$, we can derive*

$$E,\ k : I,\ 0 : N \ \vdash^{\omega.d} \ f(\boldsymbol{n}) \downarrow$$

in the infinitary system. Furthermore the non-atomic cut-formulas in this derivation are the induction-formulas occurring in the EA(I;O) proof.

3.1 Complexity Bounds

Notation. We signify that an infinitary derivation involves only Σ_1 cut formulas C, by attaching a subscript 1 to the proof-gate thus: $n : N,\ \Gamma \ \vdash_1^\alpha \ A$. If all cut formulas are atomic equations (or possibly conjunctions of them) we attach a subscript 0 instead.

Lemma 3. *(Bounding) Let Γ, A consist of (conjunctions of) atomic formulas only.*

1. *If $E,\ k:I,\ n:N,\ \Gamma \vdash_1^\alpha\ m:N$ then $m \le n + G(2^\alpha, k)$.*
2. *If $E,\ k:I,\ n:N,\ \Gamma \vdash_1^\alpha\ \exists \boldsymbol{a} A(\boldsymbol{a})$ then there are numbers $\boldsymbol{m} \le n + G(2^\alpha, k)$ such that $E,\ k:I,\ n:N,\ \Gamma \vdash_0^{2\cdot\alpha}\ A(\boldsymbol{m})$.*

Theorem 4. *(Complexity) Suppose f is defined by a system of equations E and $\bar{E} \vdash f(\boldsymbol{x}) \downarrow$ in EA(I;O) with induction formulas of size at most r. Then there is an $\alpha = 2_{r-1}(\omega.d)$ such that: for all inputs $\boldsymbol{n}$ of binary length at most k we have*

$$E,\ k:I,\ 0:N\ \vdash_0^{2\cdot\alpha}\ f(\boldsymbol{n}) \simeq m \quad \text{where}\ m \le 2_r((k+1).d)\ .$$

This is a computation of $f(\boldsymbol{n})$ from E, and the number of computation steps (nodes in the binary branching tree) is less than $4^{2_{r-1}((k+1).d)}$.

Proof. First apply the Embedding Lemma to obtain a number d such that for all inputs $\boldsymbol{n}$ of binary length $\le k$,

$$E,\ k:I,\ 0:N\ \vdash^{\omega.d}\ f(\boldsymbol{n}) \downarrow$$

with cut rank r. Then apply Cut Reduction $r-1$ times, to bring the cut rank down to the Σ_1 level. This gives $\alpha = 2_{r-1}(\omega.d)$ such that

$$E,\ k:I,\ 0:N\ \vdash_1^\alpha\ f(\boldsymbol{n}) \downarrow\ .$$

Then apply the Bounding Lemma to obtain $m \le G(2^\alpha, k) = 2_r((k+1).d)$ such that

$$E,\ k:I,\ 0:N\ \vdash_0^{2\cdot\alpha}\ f(\boldsymbol{n}) \simeq m\ .$$

This derivation uses only the E axioms, the N rules and equational cuts, so it is a computation in the equation calculus. Since all the ordinal bounds belong to $2.\alpha[k]$, the height of the derivation tree is no greater than $G(2.\alpha, k)$ and the number of nodes (or computation steps) is therefore $\le G(4^\alpha, k) = 4^{G(\alpha,k)} = 4^{2_{r-1}((k+1).d)}$.

Theorem 5. *The functions provably recursive in EA(I;O) are exactly the elementary (Grzegorczyk E^3) functions. The functions provably recursive in the Σ_1 inductive fragment of EA(I;O) are exactly the Linear Space or E^2 functions. The functions provably recursive in the "level-2" inductive fragment of EA(I;O) are exactly those computable in a number of steps bounded by an exponential function of their inputs.*

Proof. By the above, if f is provably recursive in EA(I;O) then it is computable in a number of steps bounded by a finitely iterated exponential function of its inputs. This means it is elementary.

If f is provably recursive in the Σ_1 inductive fragment then we can take $r = 1$ in the above. So f is computable in a number of steps bounded by $4^{(k+1).d}$. But k

is the maximum binary length of the inputs $\boldsymbol{n}$, so this bound is just a polynomial in $\max \boldsymbol{n}$. Therefore f is in E^2.

If f is provably recursive using "level-2" inductions then, taking $r = 2$, we obtain the bound $4^{2_1((k+1).d)}$ which is $\leq 2^{p(\boldsymbol{n})}$ for some polynomial p.

Remark. If the theory EA(I;O) were formulated as a theory of *binary* (rather than unary) arithmetic then a similar analysis would characterize PTIME as the functions provably recursive in the Σ_1 inductive fragment.

4 Recapturing Peano Arithmetic

EA(I;O) is a predicative analogue of full PA – its proof theoretic ordinal is still ε_0 but the bounding functions are slow–growing rather than fast–growing (see also Wirz [8]). By extending EA(I;O) with inductive definitions one obtains a theory ID_1(I;O) with the same strength as PA. Its ordinal is now the Bachmann–Howard ordinal, but since the bounding functions are slow growing this means that the provably recursive functions are just those of PA, i.e. fast growing below ε_0 (see Wainer and Williams [6]). Williams' Ph.D. thesis [7] treats further extensions to theories of finitely–iterated inductive definitions ID_n(I;O) having the same strength as classical ID_{n-1}.

References

1. S. Bellantoni and S. Cook, *A new recursion theoretic characterization of the polytime functions*, Computational Complexity Vol. 2 (1992) 97 - 110.
2. W. Buchholz, *An independence result for* Π^1_1–CA + BI, Annals of Pure and Applied Logic Vol. 23 (1987) 131 - 155.
3. D. Leivant, *Intrinsic theories and computational complexity* , in D. Leivant (Ed) Logic and Computational Complexity, Lecture Notes in Computer Science Vol. 960, Springer-Verlag (1995) 177 - 194.
4. E. Nelson, *Predicative arithmetic*, Princeton (1986).
5. G.E. Ostrin and S.S. Wainer, *Proof theoretic complexity*, in H.Schwichtenberg and R. Steinbrüggen, Proof and System Reliability, Kluwer (2002) 369 - 397.
6. S.S. Wainer and R.S. Williams, *Inductive definitions over a predicative arithmetic*, Annals of Pure and Applied Logic, to appear.
7. R.S. Williams, *Finitely iterated inductive definitions over a predicative arithmetic*, Ph.D. thesis, Leeds (2004).
8. M. Wirz, *Wellordering two sorts*, preliminary report, Bern (2004).

Domain-Theoretic Formulation of Linear Boundary Value Problems

Dirk Pattinson

Department of Computing, Imperial College London, UK*

Abstract. We present a domain theoretic framework for obtaining exact solutions of linear boundary value problems. Based on the domain of compact real intervals, we show how to approximate both a fundamental system and a particular solution up to an arbitrary degree of accuracy. The boundary conditions are then satisfied by solving a system of imprecisely given linear equations at every step of the approximation. By restricting the construction to effective bases of the involved domains, we not only obtain results on the computability of boundary value problems, but also directly implementable algorithms, based on proper data types, that approximate solutions up to an arbitrary degree of accuracy. As these data types are based on rational numbers, no numerical errors are incurred in the computation process.

1 Introduction

We consider the linear non-homogeneous system of differential equations

$$\dot{y}(t) = A(t)y(t) + g(t) \qquad 0 \leq t \leq 1 \tag{1}$$

where $g : [0,1] \to [0,1]^n$ is a continuous, time dependent vector function and $A : [0,1] \to [-a,a]^{n\times n}$ is a continuous, time dependent $n \times n$ matrix.

As A is continuous on $[0,1]$, every entry a_{ij} of A will attain its supremum, and we can assume without loss of generality that A takes values in $[-a,a]^{n\times n}$ for $a \in \mathbb{R}$ large enough. We consider the differential equation (1) together with n linear boundary conditions of the form

$$d_i^T y(0) - c_i^T y(1) = p_i \qquad (i = 1, \ldots, n) \tag{2}$$

where $d_1, \ldots, d_n, c_1, \ldots, c_n \in \mathbb{R}^n$ are (column) vectors and $p_1, \ldots, p_n \in \mathbb{R}$.

For any solution y of (1),(2) and $c > 0$, we have that $z = cy$ solves the equation $\dot{z} = Az + cg$, together with the boundary conditions $d_i^T z(0) - c_i^T z(1) = cp_i$ for $i = 1, \ldots, n$. By rescaling the original equation, we can therefore assume $\|g\| \leq 1$ without loss of generality.

* On leave from LMU München, Germany.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 385–395, 2005.

Standard software packages numerically compute solutions of boundary value problems, but due to the floating point representation of the real numbers involved, there is no guarantee on the correctness of the computed results. Indeed, the accumulation of round-off errors can lead to grossly incorrect values, see e.g. [13].

Correctness guarantees for numerical computations can be given in the framework of interval analysis [14]. There, real numbers are represented as intervals, and one applies outward rounding, if the result of an arithmetical operation is not machine representable. While this yields provably correct estimates of the solution, one has no control over the outward rounding, which can produce unduly large intervals. For an implementation of the interval analysis approach one can therefore not give any guarantees on convergence speed.

The approach of this paper is to integrate techniques from domain theory [1, 11] with methods of mathematical analysis. While standard numerical analysis generally pre-supposes exact real numbers and functions as a basic data type, the domain theoretic approach is based on finitely representable data types, which are faithful towards the computational process on a digital computer. In this model, real numbers and real functions arise as limits of finite approximations. In the computation process, a sequence of finitely representable approximations of the input data is transformed into a sequence of finite approximations of the output.

As we can compute without loss of arithmetical precision on finite approximations of numerical data, we can guarantee of the convergence speed of a process also for an implementation. Moreover, if we equip the involved domains with an effective structure, we obtain results about the computability of numerical constructions.

The integration of domain theory and mathematical analysis has already proven a healthy marriage in many application areas. We mention the survey paper [3] and refer to [9, 2, 4, 6] for applications in exact real arithmetic, integration theory and computing with differentiable functions.

Recently, the domain theoretic approach was applied to the solution of initial value problems [5, 8, 7]. In the present paper, this approach is adapted accordingly to deal with linear boundary value problems. We compute approximations to both a fundamental system of solutions and a particular solution by solving $n+1$ initial value problems and then solve a system of approximately given linear equations to obtain a linear combination of the particular solution and the fundamental system that satisfies the boundary conditions. As the solutions of initial value problems in general only exist locally, we cannot use the methods of [5, 8, 7] directly. We therefore need to develop a new technique which is specific to linear differential equations, and produces approximations to the solution on the whole of the unit interval. Using an interval-version of Cramer's rule, the solutions of the initial value problems are then combined to satisfy the boundary conditions. The main contribution of the present paper is twofold: First, we present a domain theoretic method for obtaining global solutions of linear non-homogeneous initial value problems. In a second step, the solution of the

initial value problems are then combined to a solution which satisfies the boundary conditions. The resulting algorithm then produces a sequence of functions, which converge to the solution iff the solution is unique, and to the everywhere undefined function $\perp$ otherwise.

Related Work. We are not aware of any work regarding the computability of boundary value problems. For treatments in the framework of interval analysis, see [15, 12]. Compared with these methods, we believe that the main novelty of our approach is the fact that the computations can be carried out on the basis of proper data types and the resulting guarantee on the convergence speed for implementations.

2 Preliminaries and Notation

We use standard notions of domain theory, see for example [16, 1, 11]. Our approach is based on the *interval domain* $(\mathbb{IR}, \sqsubseteq)$ where

$$\mathbb{IR} = \{[\underline{a}, \overline{a}] : \underline{a}, \overline{a} \in \mathbb{R}, \underline{a} \leq \overline{a}\} \cup \{\mathbb{R}\} \text{ and } a \sqsubseteq b \text{ iff } b \subseteq a$$

is the set of compact real intervals augmented with $\mathbb{R}$, ordered by reverse inclusion. For an interval $[\underline{a}, \overline{a}]$, we write $\mathrm{I}[\underline{a}, \overline{a}]$ for the sub-domain $\mathbb{IR}$ of all intervals contained in $[\underline{a}, \overline{a}]$ and $\mathbb{IR}^n$ (resp. $\mathrm{I}[\underline{a}, \overline{a}]^n$) for the n-fold product of the $\mathbb{IR}$ (resp. $\mathrm{I}[\underline{a}, \overline{a}]$) with itself, equipped with component-wise order; $\perp$ denotes the least element of a partial order. We use the canonical extension of arithmetic operations to intervals without mention, that is, for $a, b \in \mathbb{IR}$ we let $a \operatorname{op} b = \{x \operatorname{op} y : x \in a, y \in b\}$ for $\operatorname{op} \in \{+, -, \cdot, /\}$ where $a/b = \mathbb{R}$ if $0 \in b$. For example, this gives a function $\det : \mathbb{IR}^{n \times n} \to \mathbb{IR}$ computing interval determinants.

The width of a compact interval $[\underline{a}, \overline{a}]$ is $w([\underline{a}, \overline{a}]) = \overline{a} - \underline{a}$ and $w(\mathbb{R}) = \infty$. We let $w(a_1, \ldots, a_k) = \max\{w(a_i) : 1 \leq i \leq k\}$ for $(a_1, \ldots, a_k) \in \mathbb{IR}^k$; note that this includes the case of interval matrices $G \in \mathbb{IR}^{k \times k}$. If $f : [0, 1] \to \mathbb{IR}^k$ is a function, we put $w(f) = \sup\{w(f(t)) : t \in [0, 1]\}$.

Our constructions will live in the following function spaces, which capture approximation of the matrix A, the non-homogeneous part g of the equation and of the constructed solution, respectively. We let

$$\mathcal{M} = [0, 1] \Rightarrow \mathrm{I}[-a, a]^{n \times n} \quad \mathcal{G} = [0, 1] \Rightarrow \mathrm{I}[-1, 1]^n \quad \mathcal{S} = [0, 1] \Rightarrow \mathbb{IR}^n$$

equipped with the pointwise order, where $[0, 1] \Rightarrow D$ is the space of functions that are continuous w.r.t. the euclidean topology on $[0, 1]$ and the Scott topology on D for a directed-complete partial order D.

We identify a real number x with the degenerate interval $[x, x]$; in particular this allows us to view any real valued function $f : \mathrm{dom}(f) \to \mathbb{R}^n$ as taking values in $\mathbb{IR}^n$.

For an interval $a = [\underline{a}, \overline{a}]$ and $r \in \mathbb{R}$, we let $a \oplus r = [\underline{a} - r, \underline{a} + r]$; moreover $(a_1, \ldots, a_n) \oplus r = (a_1 \oplus r, \ldots, a_n \oplus r)$ for $(a_1, \ldots, a_n) \in \mathbb{IR}^n$.

Given $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, we write the sup-norm of x as $\|x\| = \max\{|a_i| : i = 1, \ldots, n\}$. For interval vectors $a \in \mathbb{IR}^n$, we put $\|a\| = \sup\{\|x\| : \mathbb{R}^n \ni x \sqsubseteq a\}$.

By a partition of an interval $[a, b]$ we mean a sequence $Q = (q_0, \ldots, q_k)$ with $a = q_0 < \cdots < q_k = b$; we denote the norm of Q by $|Q| = \max\{q_{i+1} - q_i : 0 \leq i < k\}$ and write $\mathcal{P}[a, b]$ for the set of partitions of $[a, b]$. A partition $Q = (q_0, \ldots, q_k)$ refines a partition $P = (p_0, \ldots, p_l)$, denoted by $P \sqsubseteq Q$, if $\{p_1, \ldots, p_l\} \subseteq \{q_1, \ldots, q_k\}$.

If $f : [a, b] \to \mathbb{IR}$ is a function, we write $f = [\underline{f}, \overline{f}]$ in case $f(t) = [\underline{f}(t), \overline{f}(t)]$ for all $t \in [a, b]$ and let $\int_s^t f(x)dx = [\int_s^t \underline{f}(x)dx, \int_s^t \overline{f}(x)dx]$ if $s \leq t$. This is extended component-wise to functions $f : [a, b] \to \mathbb{IR}^n$.

3 Construction of Fundamental Matrices and Particular Solutions

It is well known that the set of solutions of equation (1) carries the structure of an n-dimensional affine space, which is the translation of the vector space of solutions of the homogeneous problem

$$\dot{y}(t) = A(t)y(t) \qquad 0 \leq t \leq 1 \tag{3}$$

by any solution of the non-homogeneous problem (1). We recall the following classical terminology.

Definition 1. *A* fundamental matrix *of the homogeneous problem (3) is a time-depended $n \times n$ matrix $Y(t) = (y_1(t), \ldots, y_n(t))$ where $y_1, \ldots, y_n$ are linearly independent solutions of (3). A solution of the differential equation (1) is called a* particular solution.

Given a fundamental matrix $Y = (y_1, \ldots, y_n)$ for (3) and a particular solution y_p of the inhomogeneous equation (1), all solutions of (1) are of the form $y_p + \sum_{i=1}^n \alpha_i y_i$ for a sequence $\alpha_1, \ldots, \alpha_n$ of scalars. One then tries to satisfy the boundary conditions by an appropriate choice of $\alpha_1, \ldots, \alpha_n$.

In this section, we describe a method for obtaining a fundamental matrix and a particular solution of the equation (3). This is achieved by solving $n + 1$ initial value problems with linearly independent initial conditions. The following classical lemma ensures, that this gives rise to a fundamental matrix.

Lemma 2. *Suppose $y_1, \ldots, y_n$ are solutions of (3) and $t \in [0, 1]$. Then $y_1, \ldots, y_n$ are linearly independent iff $y_1(t), \ldots, y_n(t)$ are linearly independent.*

In particular, this entails that $y_1(s), \ldots, y_n(s)$ are linearly independent for all $s \in [0, 1]$ provided that there is some $t \in [0, 1]$ such that $y_1(t), \ldots, y_n(t)$ are linearly independent. For the remainder of this section, we therefore focus on solving the differential equation (1), together with the initial condition

$$y(0) = y^0, \text{ assuming } \|y^0\| + \|g\| \leq 1. \tag{4}$$

This allows to compute both a fundamental system of (3) and a particular solution of (1): to obtain a fundamental system, we let $g = 0$ and it suffices to

consider n linearly independent initial conditions (in fact, we will be using n unit vectors $e^1, \ldots, e^n$). For a particular solution, we let $y^0 = 0$; recall our convention $\|g\| \leq 1$.

We cannot directly apply the methods outlined in [8, 7], since there it is presupposed that the function $f(t, y)$ defining the differential equation $\dot{y} = f(t, y)$ is defined in a rectangle $[0, \delta] \times [-K, K]^n \to [-M, M]^n$ with $\delta M \leq K$. In the case of Equation (3), this condition only allows us to compute solutions on a subinterval $[0, \delta]$ of $[0, 1]$, where δ depends on K and M. Note that in general, we cannot expect to obtain a solution of an initial value problem $\dot{y} = f(t, y)$, $y(0) = y^0$ for $f : [0, 1] \times \mathbb{R}^n \to \mathbb{R}^n$ to exist on the whole of $[0, 1]$.

Example 3. The initial value problem $\dot{y} = y^2, y(0) = 1$ has no solution defined on the whole of $[0, 1]$. To see this, note that $y(t) = \frac{1}{1-t}$ is the unique solution on $[0, 1)$ and a solution defined on the whole of $[0, 1]$ would need to agree with y on $[0, 1)$ as well as being continuous.

However, the situation is different for linear systems. Instead of using a glueing method to extend a domain theoretic solution to the whole of $[0, 1]$, we present a variant of Euler's technique that directly allows us to obtain solutions of (1),(4) on the whole of $[0, 1]$.

The idea of the method is the observation that any solution of (1),(4) is bounded in norm on $[0, 1]$. An a priori estimate of the bound on every subinterval of $[0, 1]$ provides us with the necessary information to compute enclosures of the real solution based on a partition of $[0, 1]$. As we cannot assume that the data defining the initial value problem is exactly given, our general treatment assumes that we are dealing with approximations of this data throughout. Assuming that these approximations converge to the data defining the problem, we obtain a solution of the original problem in the limit. Technically, we therefore work with interval matrices, an interval initial condition and an interval valued function $g : [0, 1] \to \mathrm{I}[0, 1]^n$ that defines the non-homogeneous part of the equation.

We now fix the terminology we are going to use in the remainder of the paper.

Terminology 4. *We collect approximations of the data that defines problem (1),(4) in the domain*

$$\mathcal{D} = \{(\mathbf{A}, \mathbf{g}, \mathbf{y}^0) \in \mathcal{M} \times \mathcal{G} \times \mathrm{I}[0, 1]^n : \|\mathbf{g}\| + \|\mathbf{y}^0\| \leq 1\}$$

with partial order inherited from $\mathcal{M} \times \mathcal{G} \times \mathrm{I}[0, 1]^n$. For a partition $Q = (q_0, \ldots, q_k)$ of $[0, 1]$ we define the following constants, which we will meet throughout the exposition:

$$\Delta_i^{(Q)} = q_i - q_{i-1} \qquad K_0^{(Q)} = 1 \qquad K_i^{(Q)} = \frac{K_{i-1}^{(Q)}}{1 - \Delta_i^{(Q)} M} \qquad L_i^{(Q)} = M K_i^{(Q)} + \|g\|$$

where $1 \leq i \leq k$ and $M = an$ (recall our assumption that the matrix A defining the problem takes values in $[-a, a]^n$). We drop the superscript (Q) if the partition is clear from the context and only consider partitions Q satisfying $|Q| \leq \frac{1}{2M}$.

As we will see later, the constant K_i is an upper bound for approximate solutions on the interval $[0, q_i]$ and L_i gives a bound on the growth in the interval $[0, q_i]$. Using the terminology introduced above, our construction takes the following form.

Definition 5. *Suppose* $D = (\mathbf{A}, \mathbf{g}, \mathbf{y}^0) \in \mathcal{D}$ *and* $Q \in \mathcal{P}[0,1]$. *We define* $y_D^Q : [0,1] \to \mathbb{IR}$ *by* $y_D^Q(0) = \mathbf{y}^0$ *and*

$$y_D^Q(t) = y_D^Q(q_i) + \int_{q_i}^{t} \mathbf{A}(t)\,(y_D^Q(q_i) \oplus L_{i+1}^{(Q)} \Delta_{i+1}^{(Q)}) + \mathbf{g}(x)dx$$

for all $t \in (q_i, q_{i+1}]$.

The idea behind this definition is that the term $L_{i+1}\Delta_{i+1}$ acts as a bound on the growth of any solution y of the original problem, and extending the approximate solution y_D^Q with this bound therefore gives rise to an enclosure. Technically, this guarantees the soundness of our construction, which needs the following additional lemma.

Lemma 6. *Suppose* y *is a solution of the IVP (1),(4) and* $Q = (q_0, \ldots, q_k) \in \mathcal{P}[0,1]$. *Then*

$$\|y^0 + \int_0^t A(x)y(x) + g(x)dx\| \leq K_i^{(Q)}$$

for all $t \in [0, q_i]$.

This statement gives the promised bound on the growth of the (unique) solution of the IVP in the subintervals $[q_i, q_{i+1}]$, and is the essential step in the proof of the soundness of our construction.

Proposition 7 (Soundness). *Suppose* y *is the unique solution of the initial value problem (1),(4),* $D \in \mathcal{D}$ *with* $D \sqsubseteq (A, g, y^0)$ *and* $Q \in \mathcal{P}[0,1]$. *Then* $y_D^Q \sqsubseteq y$.

In order to approximate the solution of the problem (1),(4), we will refine the partitions and approximate data that defines the problem simultaneously. Our next goal is therefore to show, that this gives rise to an increasing sequence of approximate solutions. Monotonicity in D is straightforward:

Lemma 8. *Suppose* $D \sqsubseteq E \in \mathcal{D}$ *and* $Q \in \mathcal{P}[0,1]$. *Then* $y_D^Q \sqsubseteq y_E^Q$.

Montonicity in Q is suprisingly difficult to show; we include a proof sketch.

Proposition 9 (Monotonicity in Q). *Suppose* $D \in \mathcal{D}$ *and* $P \sqsubseteq Q$. *Then* $y_D^P \sqsubseteq y_D^Q$.

Proof. We assume that $D = (\mathbf{A}, \mathbf{g}, \mathbf{y}^0)$, $Q = (q_0, \ldots, q_k)$ and $P = (p_0, \ldots, p_l)$. We show, by induction on i, that

$$y_D^P \restriction [0, q_i] \sqsubseteq q_D^Q \restriction [0, q_i],$$

where the case $i = 0$ is trivial. To get the statement for $i + 1$, let $t \in [q_i, q_{i+1}]$ and put $j = \max\{j : p_k \leq q_i\}$. Then, by additivity of integrals,

$$\begin{aligned} y_D^P(t) &= y_D^P(q_i) + \int_{q_i}^{t} \mathbf{A}(x)(y_D^P(p_j) \oplus L_{j+1}^{(P)} \Delta_{j+1}^{(P)}) + \mathbf{g}(x)dx \\ &\sqsubseteq y_D^Q(q_i) + \int_{q_i}^{t} \mathbf{A}(x)(y_D^P(q_i) \oplus L_{j+1}^{(P)} \Delta_{i+1}^{(Q)}) + \mathbf{g}(x)dx \\ &\sqsubseteq y_D^Q(q_i) + \int_{q_i}^{t} A(x)(y_D^Q(q_i) \oplus L_{i+1}^{(Q)} \Delta_{i+1}^{(Q)}) + \mathbf{g}(x)dx = q_D^Q(t) \end{aligned}$$

by induction hypothesis.

The last proposition shows, that we can construct an increasing sequence of functions $y_{D_k}^{Q_k}$ from an increasing sequence $(D_k)_{k\in\omega}$ in $\mathcal{D}$ and an increasing sequence of partitions $(Q_k)_{k\in\omega}$. Our next concern is to show, that this sequence actually converges to a solution of the IVP (1),(4).

Proposition 10 (Convergence Speed). *Suppose $D_k = (\mathbf{A}_k, \mathbf{g}_k, \mathbf{y}_k^0)$ is an increasing sequence in $\mathcal{D}$ with $\bigsqcup_k D_k = (A, g, y^0)$ and (Q_k) is an increasing sequence in $\mathcal{P}[0,1]$ such that $w(\mathbf{A}_k), w(\mathbf{g}_k), w(\mathbf{y}_k^0), |Q_k| \in \mathcal{O}(2^{-k})$. Then $w(y_{D_k}^{Q_k}) \in \mathcal{O}(2^{-k})$.*

Proof. Similar to the corresponding statement in [7].

As a corollary, we obtain completeness, that is, our iterates converge to the (unique) solution of the problem.

Corollary 11. *Under the hypothesis of the previous proposition, $y = \bigsqcup_{k\in\omega} y_{D_k}^{Q_k}$ where y is the unique solution of the problem (1),(4).*

4 Computability of Fundamental Matrices and Particular Solutions

In the previous section, we have used arbitrary interval valued functions to construct approximations to fundamental matrices and particular solutions. In this section, we restrict our attention to the bases of the effectively given domains involved. This lead to computability assertions for both a fundamental matrix and a particular solution. Our construction is parametric in an effective, recursively enumerable, dense subring $R \subseteq \mathbb{R}$, such as the rational or dyadic numbers. We use the following terminology.

Definition 12. *We denote by $\mathbb{IR}_R = \{[\underline{a}, \overline{a}] \in \mathbb{IR} : \underline{a}, \overline{a} \in R\}$ the set of intervals with endpoints in R and $\mathcal{P}[0,1]_R$ the partitions whose points lie in R. We put $\mathbb{IR}_R^n = (\mathbb{IR}_R)^n$. A function $f = [\underline{f}, \overline{f}] : [0,1] \to \mathbb{IR}^k$ is called*

1. piecewise R-constant, *if there exists a partition* $Q = (q_0, \ldots, q_k) \in \mathcal{P}[0,1]_D$ *s.t.* $f \upharpoonright (q_{i-1}, q_i)$ *is constant with value* $\alpha_i \in \mathbb{IR}_R$ *for* $i = 1, \ldots, k$ *and* $f(q_i) = \alpha_i \sqcap \alpha_{i+1}$ *for* $i = 1, \ldots, k-1$ *where* $\sqcap$ *denotes least upper bound.*
2. piecewise R-linear, *if there exists a partition* $Q = (q_0, \ldots, q_k) \in \mathcal{P}[0,1]_D$ *such that* $\underline{f} \upharpoonright [q_{i-1}, q_i]$ *and* $\overline{f} \upharpoonright [q_{i-1}, q_i]$ *are linear for* $i = 1, \ldots, k$ *and* $f(q_i) \in \mathbb{IR}_R$ *for* $i = 0, \ldots, k$.

With this terminology, we consider the following bases of the domains $\mathcal{M}, \mathcal{G}, \mathcal{S}$.

- $\mathcal{M}_R = \{\mathbf{A} \in \mathcal{M} : \mathbf{A}$ *piecewise* R *constant*$\}$
- $\mathcal{S}_R = \{y \in \mathcal{S} : y$ *piecewise* R*-linear* $\}$
- $\mathcal{G}_R = \{h \in \mathcal{G} : h$ *piecewise* R*-constant* $\}$

and we let $\mathcal{D}_R = \{(\mathbf{A}, \mathbf{g}, \mathbf{y}^0) \in \mathcal{D} : \mathbf{A} \in \mathcal{M}_R, \mathbf{g} \in \mathcal{G}_R, \mathbf{y}^0 \in \mathrm{I}[0,1]_R\}$.

It is known these bases provide an effective structure for the domains under consideration. We refer to [16] for the notion of effectively given domains; for ease of presentation we suppress the explicit enumeration of the base.

Proposition 13. *The set* $\mathcal{X}_\mathcal{D}$ *is a base of* $\mathcal{X}$ *for* $\mathcal{X} \in \{\mathcal{M}, \mathcal{G}, \mathcal{S}, \mathbb{IR}\}$ *which provides* $\mathcal{X}$ *with an effective structure.*

Proof. It has been shown in [10] that $\mathcal{X}_\mathcal{D}$ is a base of $\mathcal{X}$; the effectiveness requirement is a straightforward verification.

It can now easily be seen that our constructions from the previous section restrict to the bases just introduced.

Lemma 14. *Suppose* $D \in \mathcal{D}_R$ *and* $Q \in \mathcal{P}[0,1]_R$. *Then* $y^Q_D \in \mathcal{S}_R$, *and* y^Q_D *can be effectively constructed.*

Proof. Given $D = (\mathbf{A}, \mathbf{g}, \mathbf{y}^0)$ and $Q = (q_0, \ldots, q_k)$, the function $\lambda t.\mathbf{A}(t) y^Q_D(q_i) \oplus L_{i+1}\Delta_{i+1} + \mathbf{g}(t)$ is piecewise constant on $[q_i, q_{i+1}]$, hence its integral is piecewise linear and can be computed without divisions.

Recall that an element $e \in E$ of a domain E with effective base E_0 is computable, if the set of basis basis elements $\{e_0 \in E_0 : e_0 \ll e\}$ is recursively enumerable, where $\ll$ is the approximation order of E (see [16, 1] for details). As the data $(A, g, y^0) \in \mathcal{M} \times \mathcal{G} \times \mathrm{I}[0,1])$ consists of (maximal) elements of effectively given domains, we can therefore speak of a *computable* initial value problem. This immediately gives the following corollary.

Corollary 15. *Suppose that* A, g, y^0 *are computable. Then the unique solution of the problem (1),(4) is computable. In particular, both a fundamental matrix of (3) and a particular solution of (1) are computable.*

Proof. As A, g, y^0 are computable, we can obtain recursive increasing sequences $(\mathbf{A}_k)$, $(\mathbf{g}_k)$, $(\mathbf{y}^0_k)$ in $\mathcal{M}_R, \mathcal{G}_R, \mathrm{I}[0,1]_R$, respectively. Choosing a recursive increasing sequence (Q_k) in $\mathcal{P}[0,1]_R$ with $\lim_{k\to\infty} |Q_k| = 0$, we obtain an recursive increasing sequence $y^{Q_k}_{D_k} \in \mathcal{S}_R$ that converges to the solution y of (1),(4), showing that y is computable.

5 Satisfaction of Boundary Conditions

For a fundamental system $(y_1, \ldots, y_n)$ of (3) and a particular solution y_p of (1), we have already seen that all solutions y of the problem (1) are of the form $y = y_p + \sum_{i=1}^{n} \alpha_i y_i$. We now address the problem of finding the correct scalars $\alpha_1, \ldots, \alpha_n$ such that y satisfies the boundary conditions (2). In order to make notation manageable, we introduce the matrices

$$B_0 = \begin{pmatrix} d_1^T \\ \vdots \\ d_n^T \end{pmatrix} \qquad B_1 = \begin{pmatrix} c_1^T \\ \vdots \\ c_n^T \end{pmatrix} \qquad Y = (y_1, \ldots, y_n)$$

where Y is a fundamental system of the linear equation (3). Classically, we have the following result:

Proposition 16. *The boundary value problem (1),(2) has a unique solution iff* $\det(B_0 Y(0) - B_1 Y(1)) \neq 0$ *and this condition is independent of the fundamental system.*

Knowing only approximations of the fundamental system, it can be only semi-decidable whether the boundary value problem has a unique solution. In order to make this precise, we need the following definition.

Definition 17. *An* effectively given *boundary value problem of the form (1),(2) is a recursive and monotone sequence of four-tuples* $(\mathbf{A}_k, \mathbf{g}_k, \mathbf{B}_k^0, \mathbf{B}_k^1)_{k\in\omega}$ *in* $\mathcal{M}_R \times \mathcal{G}_R \times \mathbb{IR}_R^{n\times n} \times \mathbb{IR}_R^{n\times n}$ *such that* $w(\mathbf{A}_k), w(\mathbf{g}_k), w(\mathbf{B}_k^0), w(\mathbf{B}_k^1) \to 0$ *as* $k \to \infty$.

A solution *of an effectively given boundary value problem is a solution of (1),(2) for* $A = \bigsqcup_k \mathbf{A}_k$, $g = \bigsqcup_k \mathbf{g}_k$ *and* $B^i = \bigsqcup_k \mathbf{B}_k^i$ *for* $i = 0, 1$.

Together with the computability results of the previous section, we arrive at our first genuine statement about boundary value problems.

Proposition 18. *It is semi-decidable whether an effectively given boundary value problem has a unique solution.*

Proof. We have shown in Corollary 15 that a fundamental system Y of the homogeneous problem (3) can be constructed effectively. Suppose $(Y_k)_{k\in\omega}$ is a sequence approximating a fundamental system of (3). By Scott continuity of the determinant, we have

$$\det B^0 Y(0) = B^1 Y(1) = \bigsqcup_k \det B_k^1 Y_k(0) - B_k^1 Y_k(1)$$

and therefore $\det B^0 Y(0) = B^1 Y(1) \neq 0$ iff $0 \notin \det(B_k^1 Y_k(0) - B_k^1 Y_k(1))$ for some $k \in \omega$.

The following example shows that unique solvability of boundary value problems is not decidable in general.

Example 19. Consider a recursive increasing sequence (a_k) in $\mathbb{IR}_D$ with $\lim_{k\to\infty} w(a_k) = 0$ and the effectively given boundary value problem $(a_k, 0, 0, 0)_{k\in\omega}$, representing the equation $\dot{y} = ay, y(0) = y(1) = 0$ for $a = \bigsqcup_k a_k$. This problem has a unique solution iff $a = 0$ thus if solvability of boundary value problems were decidable, we could decide whether $\bigsqcup_{k\in\omega} a_k = 0$ for a recursive increasing sequence (a_k).

Assuming that $\det(B^0Y(0) - B^1Y(1)) \neq 0$, our next task is to determine the scalar values for the combination of the solution constituting the fundamental system. If Y is a fundamental system for the linear equation (3) and $p = (p_1, \ldots, p_n)^T$, this boils down to solving the linear system of equations

$$(B^0Y(0) - B^1Y(1)) \cdot (\alpha_1, \ldots, \alpha_n)^T = B^1 y_p(1)$$

assuming that the particular solution y_p satisfies the initial condition $y_P(0) = 0$. For simplicity, we use Cramer's rule, as it can be easily seen to be Scott continuous with an exponential speed of convergence. In practice, one will probably want to use more sophisticated techniques like a Scott-continuous version of Gauss elimination.

Definition 20. *Suppose* $G = (g_1, \ldots, g_n) \in \mathbb{IR}^{n\times n}$ *and* $b \in \mathbb{IR}^n$. *We define* $\mathcal{C}(G, b) = (x_1, \ldots, x_n)^T$ *where* $x_i = \det(g_1, \ldots, g_{i-1}, b, g_{i+1}, \ldots, g_n)/\det(G)$ *and call* y *the result of* applying Cramer's Rule *to* G *and* b.

Note that $\mathcal{C}(G, b) = \bot$ if $0 \in \det(G)$. Clearly for $G \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, $\mathcal{C}(G, b)$ gives the unique solution of the linear equation $Gx = b$. Our concern is continuity and speed of convergence, if G and b are approximated by interval matrices and vectors, respectively.

Lemma 21. *Suppose* $(G_k)_{k\in\omega}$ *and* $(b_k)_{k\in\omega}$ *are monotone sequences of interval matrices and vectors, respectively. Then* $\mathcal{C}(\bigsqcup_k G_k, \bigsqcup_k b_k) = \bigsqcup_{k\in\omega} \mathcal{C}(G_k, b_k)$. *Moreover,* $0 \notin \det(G_0)$ *and* $w(G_k), w(b_k) \in \mathcal{O}(2^{-k})$, *then* $w(\mathcal{C}(A_k, b_k)) \in \mathcal{O}(2^{-k})$.

This lemma puts us in the position to calculate the coefficients for obtaining the solution of the boundary value problem (1),(2), as applying Cramer's Rule restricts to a computable map $\mathcal{C} : \mathbb{IR}_R^{n\times n} \times \mathbb{IR}_R^n \to \mathbb{IR}_R^n$.

Theorem 22. *Suppose* $(\mathbf{A}_k, \mathbf{g}_k, \mathbf{B}_k^0, \mathbf{B}_k^1)$ *is an effectively given boundary value problem. Then we can effectively construct an increasing recursive sequence* $(y_k)_{k\in\omega}$ *in* $\mathcal{S}_R$ *such that* $\bigsqcup_k y_k = y$ *if the problem has a unique solution* y, *and* $\bigsqcup_k y_k = \bot$, *otherwise.*

Moreover, if $w(\mathbf{A}_k), w(\mathbf{g}_k), w(\mathbf{B}^0), w(\mathbf{B}^1) \in \mathcal{O}(2^{-k})$ *and* $y_{k_0} \neq \bot$ *for some* k_0, *we have* $w(y_k) \in \mathcal{O}(2^{-k})$ *for* $k \geq k_0$.

As all these operations can be carried out on the basis of the domains involved, we have the following corollary:

Corollary 23. *Suppose* A, g, B^0, B^1 *are computable. If problem (1),(2) has a unique solution, this solution is computable.*

References

1. S. Abramsky and A. Jung. Domain Theory. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3. Clarendon Press, 1994.
2. A.Edalat. Domain theory and integration. *Theor. Comp. Sci.*, 151:163–193, 1995.
3. A. Edalat. Domains for computation in mathematics, physics and exact real arithmetic. *Bulletin of Symbolic Logic*, 3(4):401–452, 1997.
4. A. Edalat and M. Krznaric. Numerical integration with exact arithmetic. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Automata, Languages and Programming, Proceedings of ICALP 1999*, volume 1644 of *LNCS*. Springer.
5. A. Edalat, M Krznarić, and A. Lieutier. Domain-theoretic solution of differential equations (scalar fields). In *Proceedings of MFPS XIX*, volume 83 of *Elect. Notes in Theoret. Comput. Sci.*, 2004.
6. A. Edalat, A. Lieutier, and D. Pattinson. A computational model for differentiable functions. In V. Sassone, editor, *Proc. FoSSaCS 2005*, Lect. Notes in Comp. Sci., 2005. to appear.
7. A. Edalat and D. Pattinson. A domain theoretic account of euler's method for solving initial value problems. sumitted, available at `http://www.ifi.lmu.de/~pattinso/Publications/`.
8. A. Edalat and D. Pattinson. A domain theoretic account of picard's theorem. In *Proc. ICALP 2004*, Lect. Notes in Comp. Sci., 2004.
9. A. Edalat and P. Sünderhauf. A domain theoretic approach to computability on the real line. *Theoretical Computer Science*, 210:73–98, 1998.
10. T. Erker, M. Esacrdò, and K. Keimel. The way-below relation of function spaces over semantic domains. *Topology and its Applications*, 89(1–2):61–74, 1998.
11. G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. Scott. *Continuous Lattices and Domains*. Cambridge University Press, 2003.
12. E. Hansen. On solving two-point boundary-value problems using interval arithmetic. In E. Hansen, editor, *Topics In Interval Analysis*, pages 74–90. Oxford University Press, 1969.
13. A. Iserles. *Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 1996.
14. R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.
15. F. A. Oliveira. Interval analysis and two-point boundary value problems. *SIAM J. Numer. Anal.*, 11:382–391, 1974.
16. V. Stoltenberg-Hansen, I. Lindström, and E. Griffor. *Mathematical Theory of Domains*. Number 22 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1994.

Membrane Computing: Power, Efficiency, Applications

Gheorghe Păun

Institute of Mathematics of the Romanian Academy,
PO Box 1-764, 7014700 Bucureşti, Romania
and
Research Group on Natural Computing,
Department of Computer Science and Artificial Intelligence,
University of Sevilla,
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
george.paun@imar.ro, gpaun@us.es

Abstract. Membrane computing is an young but already well developed branch of natural computing, having as its goal to abstract computing models from the structure and the functioning of the living cell.

The present paper is an informal introduction to membrane computing, presenting the basic ideas, some central (mathematical) results, and the main areas of application.

1 Starting from Cells

The traditional branches of natural computing, genetic algorithms (more generally, evolutionary computing), neural computing, DNA (molecular) computing, start from/use/imitate either processes taking place at the molecular level in the cell, or processes developing in populations of cells with various levels of organization, never considering the cell itself as a structured body, with inner compartments. Membrane computing, the youngest branch of natural computing, fills in this gap.

It starts from the observation that the living cell is one of the most marvellous machineries evolved by nature. The cell is the smallest living unit, a microscopic "factory", with a complex structure, an intricate inner activity, and an exquisite relationship with its environment. Both substances, from ions to large macromolecules, and information are processed in a cell, according to involved reactions, organized in a robust and at the same time sensitive manner, having as the goal the processes themselves, the life itself of the cell and of the structures where the cells are included – organs, organisms, populations.

Obviously, any cell means membranes. The cell itself is defined – separated from its environment – by a membrane, the external one. Inside the cell, several membranes enclose "protected reactors", compartments where specific biochemical processes take place.

The membranes also allow a selective passage of substances among the compartments delimited by them. This can be a simple selection by size, in the case

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 396–407, 2005.

of small molecules, or a much more intricate selection, through protein channels, which not only select, but can also move molecules from a low concentration to a higher concentration, perhaps coupling molecules, through so-called symport and antiport processes.

Much more: the membranes of a cell do not only delimit compartments where specific reactions take place in solution, hence *inside* the compartments, but many reactions in a cell develop *on the membranes*, catalyzed by the many proteins bound on them.

The biology of the cell contains many fascinating facts from a computer science point of view, and the reader is encouraged to check the validity of this assertion browsing, e.g., through [2], [16]. Sometimes, explicit statements about the cell as a computing unit [15], about the computational-like informational processes taking place in a cell [7], about the role of membranes in making possible the life itself (*life means surfaces inside surfaces* is stated in [14], while [17] puts it in computational terms: *Life = DNA software + membrane hardware*).

From a computer science point of view, these statements, about the computations taking place in a cell, raise the question whether they are mere metaphors or they correspond to computations in the standard (mathematical) understanding of this term; on the other hand, a more general temptation appears here: having in mind the encouraging experience of other branches of natural computing, to get inspired from the structure and the functioning of the living cell and define new computing models, possibly of interest for computer science, for computability in general.

Membrane computing emerged [22] as an answer to this double challenge, proposing a series of models (actually, a general framework for devising models) inspired from the cell structure (a compartmentalized space, defined by a hierarchical arrangement of membranes) and functioning (biochemical processes taking place in the compartments of the membrane hierarchy and, rather important, the way the compartments cooperate/communicate by passing chemicals and information across membranes), as well as from the cell organization in tissue. These models, called P systems, were investigated as mathematical objects, with the main goals being of a (theoretical) computer science type: computation power (in comparison with Turing machines and their restrictions), and usefulness in solving computationally hard problems. The field simply flourished at this level. Comprehensive information can be found in the web page (organized under the auspices of the European Molecular Computing Consortium, EMCC) [31]; a presentation at the level of the spring of year 2002 can be found in [23], while several applications are presented in [9].

In this paper we discuss only the cell-like P systems, whose study is much more developed than that of tissue-like P systems or of neural-like P systems. In short, such a system consists of a hierarchical arrangement of *membranes*, which delimit *compartments*, where abstract *objects* are placed. These objects correspond to the chemicals from the compartments of a cell, and they can be

either unstructured, a case when they can be represented by symbols from a given alphabet, or structured. In the latter case, a possible representation of objects is by strings over a given alphabet (but also more complex structures were considered, such as two-dimensional arrays, trees, etc). Here we discuss only the case of symbol-objects. Corresponding to the situation from reality, where the number of molecules from a given compartment matters, also in the case of objects from the regions of a P system we have to take into consideration their multiplicity, that is why we consider *multisets* of objects assigned to the regions of P systems. These objects evolve according to *rules*, which are also associated with the regions. The rules say both how the objects are changed and how they can be moved (we say *communicated*) across membranes. By using these rules, we can change the *configuration* of a system (the multisets from its compartments); we say that we get a *transition* among system configurations. The way the rules are applied imitates again the biochemistry (but goes one further step towards computability): the reactions are done in *parallel*, and the objects to evolve and the rules by which they evolve are chosen in a *non-deterministic* manner, in such a way that the application of rules is maximal. A sequence of transitions forms a *computation*, and with computations which *halt* (reach a configuration where no rule is applicable) we associate a *result*, for instance, in the form of the multiset of objects present in the halting configuration in a specified membrane.

All these basic ingredients of a membrane computing system (a P system) will be discussed further below. This brief description is meant, on the one hand, to show the passage from the "real cell" to the "mathematical cell", as considered in membrane computing, and, on the other hand, to give a preliminary idea about the computing model we are investigating.

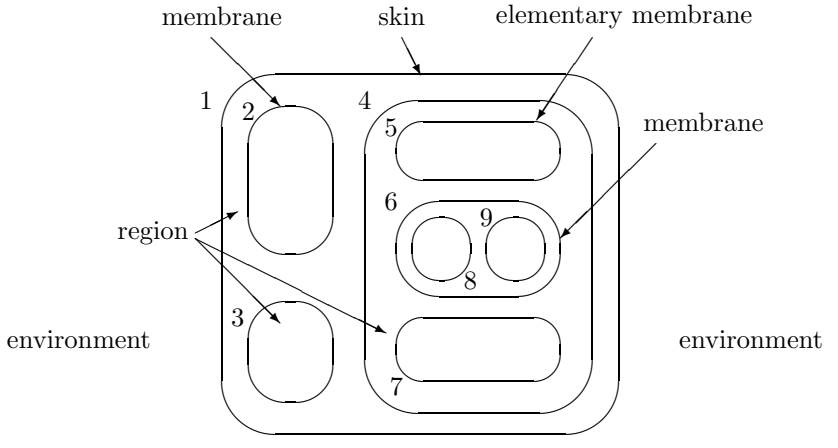
It is important to note at this stage the generality of the approach. We start from the cell, but the abstract model deals with very general notions: membranes interpreted as separators of regions, objects and rules assigned to regions; the basic data structure is the multiset (a set with multiplicities associated with its elements); the rules are used in the non-deterministic maximally parallel manner, and in this way we get sequences of transitions, hence computations. In such terms, membrane computing can be interpreted as a *bio-inspired framework for distributed parallel processing of multisets.*

We close this introductory discussion by stressing the basic similarities and differences between membrane computing and other areas of natural computing. All these areas start from biological facts and abstract computing models. Neural and evolutionary computing are already implemented (rather successfully, especially in the case of evolutionary computing) on the usual computer. DNA computing has a bigger ambition, that of providing a new hardware, leading to bio-chips, to "wet computers". For membrane computing it seems that the most realistic attempt for implementation is *in silico* (some successes are already reported) rather than *in vitro* (no attempt was made yet).

2 The Basic Classes of P Systems

We introduce now the fundamental ideas of membrane computing in a more precise way. What we look for is a computing device, and to this aim we need *data structures, operations* with these data structures, an *architecture* of our "computer", a systematic manner to define *computations* and *results* of computations.

Thus, inspired from the cell structure and functioning, the basic elements of a *membrane system* (*P system*) are (1) the *membrane structure* and the sets of (2) *evolution rules* which process (3) *multisets* of (4) *objects* placed in the compartments of the membrane structure.



A membrane structure is a hierarchically arranged set of membranes. A suggestive representation is as in the figure above. We distinguish the external membrane (corresponding to the plasma membrane and usually called the *skin* membrane) and several internal membranes (corresponding to the membranes present in a cell, around the nucleus, in Golgi apparatus, vesicles, etc); a membrane without any other membrane inside it is said to be *elementary*. Each membrane determines a compartment, also called *region*, the space delimited from above by it and from below by the membranes placed directly inside, if any exists. The correspondence membrane–region is one-to-one, so that we identify by the same label a membrane and its associated region.

In the basic class of P systems, each region contains a multiset of symbol-objects, described by symbols from a given alphabet.

The objects evolve by means of evolution rules, which are also localized, associated with the regions of the membrane structure. The typical form of such a rule is $cd \to (a, here)(b, out)(b, in)$, with the following meaning: one copy of object c and one copy of object d react and the reaction produces one copy of a and two copies of b; the newly produced copy of a remains in the same region (indication *here*), one of the copies of b exits the compartment, going to the surrounding region (indication *out*) and the other enters one of the directly

inner membranes (indication *in*). We say that the objects a, b, b are *communicated* as indicated by the commands associated with them in the right hand member of the rule. When an object exits a membrane, it will go to the surrounding compartment; in the case of the skin membrane this is the environment, hence the object is "lost", it never comes back into the system. If no inner membrane exists (that is, the rule is associated with an elementary membrane), then the indication *in* cannot be followed, and the rule cannot be applied.

A rule as above, with several objects in its left hand member, is said to be *cooperative*; a particular case is that of *catalytic* rules, of the form $ca \to cx$, where a is an object and c is a catalyst, appearing only in such rules, never changing. A rule of the form $a \to x$, where a is an object, is called *non-cooperative*.

The rules associated with a compartment are applied to the objects from that compartment, in a *maximally parallel way*: all objects which can evolve by means of local rules should do it (we assign objects to rules, until no further assignment is possible). The used objects are "consumed", the newly produced objects are placed in the compartments of the membrane structure according to the communication commands assigned to them. The rules to be used and the objects to evolve are chosen in a non-deterministic manner. In turn, all compartments of the system evolve at the same time, synchronously (a common clock is assumed for all membranes). Thus, we have two layers of parallelism, one at the level of compartments and one at the level of the whole "cell".

Note that evolution rules are stated in terms of *names of objects*, they are "multiset rewriting rules", while their application/execution is done using *copies of objects*.

A membrane structure and the multisets of objects from its compartments identify a *configuration* of a P system. By a non-deterministic maximally parallel use of rules as suggested above we pass to another configuration; such a step is called a *transition*. A sequence of transitions constitutes a *computation*. A computation is successful if it halts, it reaches a configuration where no rule can be applied to the existing objects. With a halting computation we can associate a *result* in various ways. The simplest possibility is to count the objects present in the halting configuration in a specified elementary membrane; this is called *internal output*. We can also count the objects which leave the system during the computation, and this is called *external output*. In both cases the result is a number. If we distinguish among different objects, then we can have as the result a vector of natural numbers. The objects which leave the system can also be arranged in a sequence according to the moments when they exit the skin membrane, and in this case the result is a string.

This last possibility is worth emphasizing, because of the qualitative difference between the data structure used inside the system (multisets of objects, hence numbers) and the data structure of the result, which is a string, it contains a positional information, a syntax.

Because of the non-determinism of the application of rules, starting from an initial configuration, we can get several successful computations, hence several

results. Thus, a P system *computes* (one also uses to say *generates*) a set of numbers, or a set of vectors of numbers, or a language.

Of course, the previous way of using the rules from the regions of a P system reminds the non-determinism and the (partial) parallelism from cell compartments, with the mentioning that the maximality of parallelism is mathematically oriented (rather useful in proofs); when using P systems as models of biological systems/processes, the parallelism should be replaced with more realistic features (e.g., reaction rates, probabilities, partial parallelism).

We do not give here a formal definition of a P system. The reader interested in mathematical and bibliographical details can consult the mentioned monograph [23], as well as the relevant papers from the bibliography from [31]. Of course, when presenting a P system we have to specify: the alphabet of objects (a usual finite non-empty alphabet of abstract symbols identifying the objects), the membrane structure (usually represented by a string of labelled matching parentheses), the multisets of objects present in each region of the system (represented by strings of symbol-objects, with the number of occurrences of a symbol in a string being the multiplicity of the object identified by that symbol in the multiset represented by the considered string), the sets of evolution rules associated with each region, as well as the indication about the way the output is defined.

Many modifications/extensions of the very basic model sketched above are discussed in the literature, but we do not mention them here.

3 Computing by Communication

In the systems described above, the symbol-objects were processed by multiset rewriting-like rules (some objects are transformed into other objects, which have associated communication targets). Coming closer to the trans-membrane transfer of molecules, we can consider purely communicative systems, based on the three classes of such transfer known in the biology of membranes: *uniport, symport*, and *antiport* (see [2], [5] for details). Symport refers to the transport where two (or more) molecules pass together through a membrane in the same direction, antiport refers to the transport where two (or more) molecules pass through a membrane simultaneously, but in opposite directions, while the case when a molecule does not need a "partner" for a passage is referred to as uniport.

In terms of P systems, we can consider object processing rules of the following forms: a symport rule (associated with a membrane i) is of the form (ab, in) or (ab, out), stating that the objects a and b enter/exit together membrane i, while an antiport rule is of the form $(a, out; b, in)$, stating that, simultaneously, a exits and b enters membrane i; uniport corresponds to a particular case of symport rules, of the form $(a, in), (a, out)$. An obvious generalization is to consider symport rules (x, in), (x, out) and antiport rules $(x, out; y, in)$ with x, y arbitrary multisets of objects.

A P system with symport/antiport rules has the same architecture as a system with multiset rewriting rules: alphabet of objects, membrane structure, ini-

tial multisets in the regions of the membrane structure, sets of rules associated with the membranes, possibly an output membrane – with one additional component, the set of objects present in the environment. If an object is present in the environment at the beginning of a computation, then it is considered available in arbitrarily many copies (the environment is inexhaustible). This is an important detail: because by communication we do not create new objects, we need a supply of objects, in the environment, otherwise we are only able to handle a finite population of objects, those provided in the initial multiset.

The functioning of a P system with symport/antiport rules is the same as for systems with multiset rewriting rules: the transition from a configuration to another configuration is done by applying the rules in a non-deterministic maximally parallel manner, to the objects available in the regions of the system and in the environment, as requested by the used rules. When a halting configuration is reached, we get a result, in a specified output membrane. (Note that the environment takes an active part in the computation, which is one of the attractive features of this class of P systems, together with the conservation of objects, the mathematical elegance, the computational power, the direct biological inspiration.)

4 Computational Completeness

As we have mentioned before, many classes of P systems, combining various ingredients (as described above or similar) are able of simulating Turing machines, hence they are *computationally complete*. Always, the proofs of results of this type are constructive, and this have an important consequence from the computability point of view: there are *universal* (hence *programmable*) P systems. In short, starting from a universal Turing machine (or an equivalent universal device), we get an equivalent universal P system. Among others, this implies that in the case of Turing complete classes of P systems, the hierarchy on the number of membranes always collapses (at most at the level of the universal P systems). Actually, the number of membranes sufficient in order to characterize the power of Turing machines by means of P systems is always rather small.

We only mention here three of the most interesting universality results:

1. P systems with symbol-objects with catalytic rules, using only two catalysts and two membranes, are computationally universal, [10].
2. P systems with symport/antiport rules of a rather restricted size (example: three membranes, symport rules of weight 2, and no antiport rules, or three membranes and minimal symport and antiport rules) are universal, [4].
3. P systems with symport/antiport rules (of arbitrary size), using only four membranes and only three objects, are universal, [24].

We can conclude that the compartmental computation in a cell-like membrane structure (using various ways of communicating among compartments) is rather powerful. The "computing cell" is a powerful "computer".

Universality results were obtained also in the case of P systems working in the accepting mode (either we introduce a number in the initial configuration and

we say that it is accepted if the computation halts, or we simply consider the sequence of objects taken from the environment during a computation as the string recognized by the computation). An interesting problem appears in this case, because we can consider deterministic systems. Most universality results were obtained in the deterministic case, but there also are situations where the deterministic systems are strictly less powerful than the non-deterministic ones. This is proven in [13], where a class of P systems is also produced for which the deterministic versus non-deterministic problem is equivalent with the LBA problem.

The hierarchy on the number of membranes collapses in many cases also for non-universal classes of P systems, [23], but there also are cases when "the number of membrane matters", to cite the title of [12], where two classes of P systems were defined for which the hierarchies on the number of membranes are infinite.

5 Computational Efficiency

The computational power (the "competence") is only one of the important questions to be dealt with when defining a new (bio-inspired) computing model. The other fundamental question concerns the computing *efficiency*. Because P systems are parallel computing devices, it is expected that they can solve hard problems in an efficient manner – and this expectation is confirmed for systems provided with ways for producing an exponential workspace in a linear time. Three main such biologically inspired possibilities have been considered so far in the literature, and *all of them were proven to lead to polynomial solutions to* **NP***-complete problems*.

These three ideas are *membrane division*, *membrane creation*, and *string replication*. The standard problems addressed in this framework were decidability problems, starting with `SAT`, the Hamiltonian Path problem, the Node Covering problem, but also other types of problems were considered, such as the problem of inverting one-way functions, or the Subset-sum and the Knapsack problems (note that the last two are numerical problems, where the answer is not of the yes/no type, as in decidability problems). Details can be found in [23], [27], as well as in the web page from [31].

Roughly speaking, the framework for dealing with complexity matters is that of *accepting P systems with input*: a family of P systems of a given type is constructed starting from a given problem, and an instance of the problem is introduced as an input in such systems; working in a deterministic mode (or a *confluent* mode: some non-determinism is allowed, provided that the branching converges after a while to a unique configuration, or, in the weak confluent case, all computations halt and all of them provide the same result), in a given time one of the answers yes/no is obtained, in the form of specific objects sent to the environment. The family of systems should be constructed in a uniform mode by a Turing machine, working a polynomial time.

This direction of research is very active at the present moment. More and more problems are considered, the membrane computing complexity classes are refined, characterizations of the **P**≠**NP** conjecture were obtained in this frame-

work, improvements are looked for. An important recent result concerns the fact that **PSPACE** was shown to be included in $\mathbf{PMC}_D$, the family of problems which can be solved in polynomial time by P systems with the possibility of dividing both elementary and non-elementary membranes. The **PSPACE**-complete problem used in this proof was `QSAT` (see [29] for details).

There are in this area a series of open problems, mainly related to the borderline between "efficiency" (the possibility to solve computationally hard problems in polynomial time) and "non-efficiency". From [30] we know that membrane division is necessary for efficiency. However, all constructions from the papers mentioned above about P systems with membrane division use membranes which are "polarized", marked with one of the three "electrical charges" $+, -, 0$. It was recently shown [3] that the number of polarizations can be decreased to two, but it is an intriguing open problem whether or not the polarizations can be completely removed. A similar borderline question concerns the fact that the systems constructed in [29] use division of non-elementary membranes (membranes with other membranes inside can be divided, and on this occasion the contents – the inner membranes included – is replicated), which is a rather powerful operation; can this be avoided, thus solving `QSAT` in polynomial time by using systems with division of only elementary membranes?

6 Applications

Finally, let us shortly discuss some applications of membrane computing – starting however with a general discussion about the features of this area of research which make it attractive for applications in several disciplines, especially for biology.

First, there are several keywords which are genuinely proper to membrane computing and which are of interest for many applications: *distribution* (with the important system-part interaction, emergent behavior, non-linearly resulting from the composition of local behaviors), *algorithmicity* (hence easy programmability), *scalability/extensibility* (this is one of the main difficulties of using differential equations in biology), *transparency* (multiset rewriting rules are nothing else than reaction equations as customarily used in chemistry and bio-chemistry), *parallelism* (a dream of computer science, a common sense in biology), *non-determinism*, *communication* (with the marvellous and still not completely understood way the life is coordinating the many processes taking place in a cell, in contrast with the costly way of coordinating/synchronizing computations in parallel electronic computing architectures, where the communication time become prohibitive with the increase of the number of processors), and so on and so forth.

Then, for biology, besides the easy understanding of the formalism and the transparency of the (graphical and symbolic) representations, encouraging should be also the simple observation that membrane computing emerged as a bio-inspired research area, explicitly looking to the cell for finding computability models (though, not looking initially for models of relevance for the biological

research), hence it is just natural to try to use these models in the study of the very originating ground. This should be put in contrast with the attempt to "force" models and tools developed in other scientific areas, e.g., in physics, to cover biological facts, presumably of a genuinely different nature as that of the area for which these models and tools were created and proven to be adequate/useful.

Now, in what concerns the applications themselves reported up to now, they are developed at various levels. In many cases, what is actually used is the *language* of membrane computing, having in mind three dimensions of this aspect: (i) the long list of concepts either newly introduced, or related in a new manner in this area, (ii) the mathematical formalism of membrane computing, and (iii) the graphical language, the way to represent cell-like structures or tissue-like structures, together with the contents of the compartments and the associated evolution rules (the "evolution engine").

However, this level of application/usefulness is only a preliminary, superficial one. The next level is to use tools, techniques, results of membrane computing, and here there appears an important question: to which aim? Solving problems already stated, e.g., by biologists, in other terms and another framework, could be an impressive achievement, and this is the most natural way to proceed – but not necessarily the most efficient one, at least at the beginning. New tools can suggest new problems, which either cannot be formulated in a previous framework (in plain language, as it is the case in biology, whatever specialized the specific jargon is, or using other tools, such as differential equations) or have no chance to be solved in the previous framework.

Applications of all these types were reported in the literature of membrane computing. As expected and as natural, most applications were carried out in biology, but also applications in computer graphics (where the compartmentalization seems to add a significant efficiency to well-known techniques based on L systems), linguistics (both as a representation language for various concepts related to language evolution, dialogue, semantics, and making use of the parallelism, in solving parsing problems in an efficient way), management (again, mainly at the level of the formalism and the graphical language), in devising sorting and ranking algorithms, cryptography, approximate algorithms for optimization problems, etc. Applications of all these kinds – excepting the case of management, for which we refer to [6] and the references therein – can be found in[9].

These applications are usually based on experiments using programs for simulating/implementing P systems on usual computers, and there are already several such programs, more and more elaborated (e.g., with better and better interfaces, which allow for the friendly interaction with the program). We avoid to plainly say that we have "implementations" of P systems, because of the inherent non-determinism and the massive parallelism of the basic model, features which cannot be implemented, at least in principle, on the usual electronic computer – but which can be implemented on a dedicated, reconfigurable, hardware, as done in [28], or on a local network, as reported, e.g., in [8]. This does not mean that simulations of P systems on usual computers are not useful; actually, such

programs were used in all biological applications mentioned above, and can also have important didactic and research applications. An overview of membrane computing software reported in literature (some programs are available in the web page [31]) can be found in [11].

An application of a completely different type, much related to evolutionary computing, is that recently proposed in [20], [21]: looking for approximate solutions to hard optimization problems by means of *membrane algorithms*. In short, such algorithms consists of a membrane structure in the compartments of which one places local sub-algorithms and candidate solutions to the problem to solve. In each time unit, the local algorithms (they can be genetic algorithms or other type of easy to implement approximate algorithms) improve the solutions from the associated region. After that, the locally best solution is communicated to an inner membrane and the worst solution is moved out. In this way, the good solutions migrate towards the inner-most membranes. Several architectures were considered in [20], [21]: a linear structure of membranes, a tissue-like structure, dynamical structures. Experiments were made with the travelling salesman problem, and, for standard benchmark instances the obtained results were rather encouraging: in several cases the solutions were better than those obtained by simulated annealing; in almost all cases the average and the worst solutions were better than those given by simulated annealing (hence the approach seems to be trustful, we do not need to make too many experiments); in all cases, the convergence is initially very rapid, and a solution which is almost optimal is obtained in a small number of steps, and after that the improvements are very slow. The membrane algorithms are now subject of a particular interest, and several issues are under current research: addressing in this framework other problems, adding further membrane computing ingredients, implementing the algorithms in a distributed manner.

References

1. R. Alberich, J. Casasnovas, M. Llabrés, J. Miró–Juliá, J. Rocha, F. Rosselló, eds.: *Brainstorming Workshop on Uncertainty in Membrane Computing*. Universitat de les Illes Malears, Palma de Mallorca, November 2004.
2. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter: *Molecular Biology of the Cell*, 4th ed. Garland Science, New York, 2002.
3. A. Alhazov, R. Freund, Gh. Păun: P systems with active membranes and two polarizations. In [25], 20–36.
4. A. Alhazov, M. Margenstern, V. Rogozhin, Y. Rogozhin, S. Verlan: Communicative P systems with minimal cooperation. In [19], 162–178.
5. I.I. Ardelean: The relevance of biomembranes for P systems. *Fundamenta Informaticae*, 49, 1–3 (2002), 35–43.
6. J. Bartosik: Paun's systems in modeling of human resource management. *Proc. Second Conf. Tools and Methods of Data Transformation*, WSU Kielce, 2004.
7. D. Bray: Protein molecules as computational elements in living cells. *Nature*, 376 (July 1995), 307–312.
8. G. Ciobanu, G. Wenyuan: A P system running on a cluster of computers. In [18], 123–139.

9. G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds.: *Applications of Membrane Computing*, Springer, Berlin, 2005.
10. R. Freund, L. Kari, M. Oswald, P. Sosik: Computationally universal P systems without priorities: two catalysts are sufficient. *Theoretical Computer Sci.*, 330, 2 (2005), 251–266.
11. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: Available membrane computing software. In [9], 411–438.
12. O.H. Ibarra: The number of membranes matters. In [18], 218–231.
13. O.H. Ibarra: On determinism versus nondeterminism in P systems. Submitted, 2004 (available at `http://psystems.disco.unimib.it`).
14. J. Hoffmeyer: Surfaces inside surfaces. On the origin of agency and life. *Cybernetics and Human Knowing*, 5, 1 (1998), 33–42.
15. S. Ji: The cell as the smallest DNA-based molecular computer. *BioSystems*, 52 (1999), 123–133.
16. W.R. Loewenstein: *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life*. Oxford University Press, 1999.
17. S. Marcus: Bridging P systems and genomics: A preliminary approach. In [26], 371–376.
18. C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Membrane Computing. International Workshop, WMC2003, Tarragona, Spain, Revised Papers. Lecture Notes in Computer Science*, 2933, Springer, Berlin, 2004.
19. G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa, eds.: *Membrane Computing. International Workshop WMC5, Milan, Italy, 2004. Revised Papers, Lecture Notes in Computer Science*, 3365, Springer, Berlin, 2005.
20. T.Y. Nishida: An application of P system: A new algorithm for NP-complete optimization problems. In N. Callaos, *et. al.*, eds., *Proceedings of the 8th World Multi-Conference on Systems, Cybernetics and Informatics*, vol. V, 2004, 109–112.
21. T.Y. Nishida: Membrane algorithms: Approximate algorithms for NP-complete optimization problems. In [9], 301–312.
22. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143 (and Turku Center for Computer Science-TUCS Report 208, November 1998, `http://www.tucs.fi`).
23. Gh. Păun: *Computing with Membranes: An Introduction*. Springer, Berlin, 2002.
24. Gh. Păun, J. Pazos, M.J. Pérez-Jiménez, A. Rodriguez-Paton: Symport/antiport P systems with three objects are universal. *Fundamenta Informaticae*, 64, 1-4 (2005).
25. Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds.: *Proceedings of the Second Brainstorming Week on Membrane Computing, Sevilla, February 2004*. Technical Report 01/04 of Research Group on Natural Computing, Sevilla University, Spain, 2004.
26. Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.: *Membrane Computing. International Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, Revised Papers. Lecture Notes in Computer Science*, 2597, Springer, Berlin, 2003.
27. M.J. Pérez–Jiménez, A. Romero–Jiménez, F. Sancho–Caparrini: Computationally hard problems addressed through P systems. In [9], 313–346.
28. B. Petreska, C. Teuscher: A hardware membrane system. In [18], 269–285.
29. P. Sosik: The computational power of cell division in P dystems: Beating down parallel computers? *Natural Computing*, 2, 3 (2003), 287–298.
30. C. Zandron, C. Ferretti, G. Mauri: Solving NP-complete problems using P systems with active membranes. In I. Antoniou, C.S. Calude, M.J. Dinneen, eds.: *Unconventional Models of Computation*, Springer, London, 2000, 289–301.
31. The Web Page of Membrane Computing: `http://psystems.disco.unimib.it`

The Analogue of Büchi's Problem for Polynomials

Thanases Pheidas[1,⋆] and Xavier Vidaux[2,⋆⋆]

[1] Department of Mathematics, University of Crete,
Knossos Avenue, 714 09 Iraklio, Greece
[2] University of Oxford, Mathematical Institute,
24-29 St Giles, Oxford, OX1 3LB, UK

Abstract. Büchi's problem asked whether a surface of a specific type, defined over the rationals, has integer points other than some known ones. A consequence of a positive answer would be the following strengthening of the negative answer to Hilbert's tenth problem : the positive existential theory of the rational integers in the language of addition and a predicate for the property 'x is a square' would be undecidable. Despite some progress, including a conditional positive answer (pending on conjectures of Lang), Büchi's problem remains open.

In this article we prove an analogue of Büchi's problem in rings of polynomials of characteristic either 0 or $p \geq 13$.
As a consequence we prove the following result in Logic :
Let F be a field of characteristic either 0 or ≥ 17 and let t be a variable. Let R be a subring of $F[t]$, containing the natural image of $\mathbb{Z}[t]$ in $F[t]$. Let L_t be the first order language which contains a symbol for addition in R, a symbol for the property 'x is a square in $F[t]$' and symbols for multiplication by each element of the image of $\mathbb{Z}[t]$ in $F[t]$. Then multiplication is positive-existentially definable over the ring R, in the language L_t. Hence the positive-existential theory of R in L_t is decidable if and only if the positive-existential ring-theory of R in the language of rings, augmented by a constant-symbol for t, is decidable.

1 Introduction

In unpublished work J. Richard Büchi asked the following problem :

Problem 1. Is it true that, for large enough M, the only integer solutions of the recurrence relation

⋆ The first author completed the research towards the present paper while visiting the University of Cyprus. The hospitality of the Cypriot mathematicians is gratefully acknowledged.

⋆⋆ The second author acknowledges the hospitality of the University of Oxford, where the main part of this work was done under a European Marie Curie fellowship MCFI-2002-00722.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 408–417, 2005.

$$x_n^2 + x_{n-2}^2 = 2x_{n-1}^2 + 2 \qquad n = 2, \ldots, M-1 \tag{1}$$

satisfy $\pm x_n = \pm x_{n-1} + 1$?

Büchi's problem remains unsolved. In this paper:

1. We prove a positive answer to the analogous problem for a sequence of non-constant x_n in $F[t]$ if the characteristic of F is either 0 or $p \geq 17$, for $M \geq 14$. This is Theorem 4. The result for $p > 0$ are new while that for characteristic 0 follows (for $M \geq 8$) from results of Vojta. Our proofs are of an elementary nature.
2. As a consequence of (1) we prove a theorem in Logic (Theorem 7).

Büchi's problem and its implications were made public by Leonard Lipshitz in [10]. It was discussed publicly by Joseph Lipman and Barry Mazur (cf. [13]). In [25] Paul Vojta gave two pieces of evidence that Büchi's problem may have a positive answer :

(A) He proved that a conjecture of Serge Lang implies a positive answer to it for $M \geq 8$. In fact Vojta's result gives the same kind of (conditional) answer over the field $\mathbb{Q}$ of rational numbers.

(B) He showed (using Nevanlinna theory) that the analogous problem for holomorphic functions has a positive answer. This may be regarded as evidence in favor of a positive answer to Büchi's problem in the light of the observation that algebraic varieties which posses infinitely many points in some number field are often of a special geometric type (Kobayashi hyperbolic) - conjectures have been made that this correspondence is an equivalence ([24]).

In what follows X_M is the projective subvariety of the projective M-space $\mathbf{P}^M$, over $\mathbb{C}$, cut out by the equations (in projective coordinates $(x, x_0, \ldots, x_{M-1})$)

$$x_n^2 + x_{n-2}^2 = 2x_{n-1}^2 + 2x^2, \qquad n = 2, \ldots, M-1$$

Vojta observed that

Proposition 2. *For $M \geq 6$ the variety X_M is a surface of general type.*

Then he showed

Theorem 3. *(i) ([25], Theorem 3.1) For $M \geq 8$, the only curves on X_M of geometric genus 0 or 1 are the 'trivial' lines $\pm x_n = \pm x_0 - nx$, $n = 0, \ldots, M-1$.*

(ii) ([25], Theorem 6.1) Let $M \geq 8$ be an integer and let $f : \mathbb{C} \to X_M$ be a non-constant holomorphic curve. Then the image of f lies in one of the 'trivial' lines.

Statement (i) of the Theorem has as a consequence that if a conjecture of Lang (or a weaker 'question' of Bombieri) is true then Büchi's problem has a positive answer. Statement (ii) shows that the analogue of Büchi's problem for holomorphic functions has a positive answer.

Our main result in this article is an analogue of Büchi's problem for rings of polynomials.

Theorem 4. *Let F be a field and t be a variable. Assume that $(x_n)_{n=0}^{M-1}$ is a sequence of polynomials $x_n \in F[t]$, not all constant, which satisfy the recurrence relation (1). Assume that the characteristic of F is either 0 or $p \geq 17$ and $M \geq 14$.*

Then there are $\varepsilon_0, \ldots, \varepsilon_{M-1}$ with $\varepsilon_n \in \{-1, 1\}$ such that for each n, $\varepsilon_n x_n = \varepsilon_0 x_0 + n$.

We prove it in Section 2. The case of zero characteristic follows also from any of the two statements of Theorem 3, for $M \geq 8$ (for a proof of both statements see [15]).

Theorem 4, and especially the case of positive characteristic, may be regarded as evidence in favor of a positive answer to Büchi's problem, independent of that provided by Vojta.

Büchi's problem, besides being a testing ground for number theoretical techniques and conjectures, has some interesting applications in Logic. Büchi had in mind to apply the answer, if positive, in order to prove a negative answer to the following question.

Let L be the language (set of symbols) which contains a symbol for addition, a symbol for the property 'x is a square', a symbol for equality and symbols for the elements 0 and 1 (all symbols, operations and relations are interpreted in $\mathbb{Z}$ in the usual manner).

Question 5. Is the positive existential theory of L over $\mathbb{Z}$ decidable?

If answered negatively, Question 5 will be one of the strongest forms of negative answer to Hilbert's tenth problem (cf. [12] and [1]) known today - and optimal in many ways (for example cf. the decidability results in [10] and [20] and the surveys in [21] and [17]). A negative answer to Question 5 would imply that there is no algorithm to answer the solvability of systems $A \cdot X = B$ over $\mathbb{Z}$, where A is an $n \times m$ matrix and B an $m \times 1$ matrix with entries in $\mathbb{Z}$ and X is an $m \times 1$ matrix whose i-th entry is x_i^2 - each x_i is an unknown. (But the solvability of one - only - quadratic equation is decidable, cf. [4] and the more general result in [5]).

We define the languages (sets of symbols) L_t and L_T, in which we will write statements (formulas) which we will interpret in the field $F(t)$ of rational functions. The language L_t extends L by the following symbols

- Constant-symbols for the elements of the natural image of $\mathbb{Z}[t]$ in $F[t]$;
- For each element c of the natural image of $\mathbb{Z}[t]$ in $F[t]$, a unary function-symbol for the function $f_c \colon x \mapsto cx$.

The language L_T extends L by the following symbol

- A one-place predicate symbol T which will be interpreted as '$T(x)$ if and only if $x \notin F$'.

A sentence of L_t (resp. L_T) is *positive-existential* if it is of the form $\exists x \psi(x)$ where $x = (x_1, \ldots, x_n)$ is a tuple of variables and $\psi(x)$ is a disjunction of conjunctions of formulas of the form $g(x) = 0$ and 'x_i is a square' (resp. and $T(x_i)$),

where $g(x) = a_1x_1 + \cdots + a_nx_n - b$, with the $a_i, b \in \mathbb{Z}[t]$ (resp. $\in \mathbb{Z}$). The *positive-existential theory* of a subring R of $F[t]$ in the language L_t (resp. L_T) is the set of positive-existential sentences of L_t (resp. L_T) which are true in R. We ask

Question 6. Let R be a ring of functions of the independent variable t.
(a) Is the positive-existential theory of R in L_t decidable?
(b) Is the positive-existential theory of R in L_T decidable?

Büchi's problem is crucial in answering Question 6 in the following way : Let L_t^{ring} (resp. L_T^{ring}) be the extension of L_t (resp. L_T) by a symbol for multiplication in $F[t]$. Consider a ring R satisfying the hypothesis of Theorem 4. The conclusion of Theorem 4 implies that multiplication in R is positive-existentially definable both in L_t and in L_T (we will show this in the last section). Hence, if the analogue of Hilbert's tenth problem over R, in the language L_t^{ring} (resp. L_T^{ring}) has a negative answer, then the positive-existential theory of R in L_t (resp. L_T) is undecidable.

Theorem 7. *Let F be a field of characteristic either 0 or $p \geq 17$. Let t be transcendental over F (a variable) and let R be a subring of $F[t]$, containing the natural image of $\mathbb{Z}[t]$ in $F[t]$. Then*

(i) Multiplication over R is positive-existential in the language L_t. Consequently (cf. [2] and [3]) the positive-existential theory of R in L_t is undecidable.

(ii) Multiplication over R is positive-existential in the language L_T. Consequently (cf. [16]) the positive-existential theory of R in L_T is undecidable.

Negative answers to the analogue of Hilbert's tenth problem are known for all rings of polynomials over integral domains (for the language L_t^{ring} see [2] and [3] and for the language L_T^{ring} see [16]). For the status of problems regarding decidability of Diophantine problems over rings of functions, see [17] and [21].

The proof of Theorem 7 which is given in Section 3, is an easy adaptation of the analogous argument for $\mathbb{Z}$, which is due to Büchi (made public by Lipshitz). The same, essentially, proof shows a similar result for the ring of holomorphic functions on the complex plane, using Vojta's Theorem 3(ii). We state it :

Theorem 8. *(Vojta) Let R be a subring of the ring of holomorphic functions of the variable t, on the complex plane, containing the ring $\mathbb{Z}[t]$. Then multiplication in R is positive-existentially definable in the languages L_t. Consequently, if the positive-existential theory of R in the language L_t^{ring} is undecidable, then the positive-existential theory of R in L_t is undecidable as well.*

But it is an open problem whether the positive-existential theory of the ring of holomorphic functions in the language L_t^{ring} is decidable or undecidable (but cf. [11], [22] and [19]).

2 Büchi's Problem for Polynomials

In this section we prove Theorem 4. For reasons of space we leave some (relatively easy) proofs to the reader.

We suppose that the characteristic of the field F is either 0 or $p \geq M$. We can suppose without loss of generality that the base field F is algebraically closed. Also observe that if the characteristic of F is $p > 0$ and all the x_n are p-th powers, that is, $x_n \in F[t^p]$, then if the sequence $(x_n)_n$ satisfies (1), so does the sequence

$$(x_n^{1/p})_n$$

hence it suffices to consider only the case in which not all the x_n's are p-th powers. So from now on we assume:

Assumption 9. *(a) The field F is algebraically closed.*
(b) One of the following holds:

1. *The characteristic of F is 0 and at least one of the x_n is not in F.*
2. *The characteristic of F is $p \geq 3$, $p \geq M \geq 3$ and not all x_n are elements of $F[t^p]$.*

Lemma 10. *(i) The recurrence relation (1) is equivalent to:*

$$x_n^2 = (1-n)x_0^2 + nx_1^2 + n(n-1), \qquad n = 0, \ldots, M-1. \tag{2}$$

(ii) For any two distinct non-zero indices n and m we have

$$mx_n^2 = (m-n)x_0^2 + nx_m^2 - mn(m-n) \tag{3}$$

The proof is left to the reader.

From Equation (3) we observe:

Corollary 11. *(i) Assume that the characteristic of F is 0. Then all but possibly one of the x_n are non-constant polynomials.*
(ii) Assume that the characteristic of F is $p > 0$. Then all but possibly one of the x_n are in $F[t] \setminus F[t^p]$.

The next lemma gives us an invariant of the sequence (x_n) which will be used often from now on.

Lemma 12. *For any two integers $n \neq m$ the expression*

$$\frac{x_m^2 - x_n^2}{m-n} - m - n$$

does not depend on n and m.

Proof. The proof follows from Equation (3) and is left to the reader.

Definition 13. *(i) For any $n, m = 0, \ldots, M-1$ with $n \neq m$ we will be writing*

$$\nu = \frac{x_m^2 - x_n^2}{m-n} - m - n \tag{4}$$

(and we will be recalling that it does not depend on n and m).

(ii) For any $k = 0, \ldots, M-1$ *we will be denoting by*

$$\nu_k = \nu + 2k$$

that is,

$$\nu_k = \frac{x_m^2 - x_n^2}{m-n} - m - n + 2k \ .$$

Lemma 14. *Let* m, n, k *be pairwise different integers. Then the greatest common divisor of* $\{x_m, x_n, x_k\}$ *is* (1) *(the unit ideal).*

The proof follows fro Lemma 12 and is left to the reader.

Definition 15. *For any* $x \in F[t] \setminus \{0\}$, $\deg(x)$ *is the degree of* x. *We write* $\deg(0) = -\infty$. *We denote by* d *the maximum of the degrees of the* x_n *for* $n = 0, \ldots, M-1$.

Lemma 16. *One of the following is true:*

1. *All* x_n *have the same degree* d.
2. *There is an index* ℓ *such that for each* $n \neq \ell$ *we have* $\deg(x_n) = d$ *and* $\deg(x_\ell) < d$.

The proof follows from Lemma 12 and is left to the reader.

Definition 17. *Recalling Lemma 16, we let* ℓ *be an index such that for each index* n *we have*

$$\deg(x_\ell) \leq \deg(x_n) = d \ .$$

Definition 18. *For each index* n *we define*

$$\Delta_n = 2\nu_n \nu' x_n' - \nu'^2 x_n - 4 x_n x_n'^2 \ .$$

Lemma 19. *Assume that the characteristic of* F *is either* 0 *or* $p \geq 17$ *and that* $M \geq 14$. *Then for each index* n *we have*

$$\Delta_n = 0 \ . \tag{5}$$

Proof. We list a sequence of claims which lead to the proof. The proof of each claim is left to the reader.

Claim 1 : For each two indices m and n,

$$x_m \Delta_m = x_n \Delta_n \ .$$

Claim 2 : We have $\deg(\nu) \leq 2d$.

Claim 3 : The function Δ_n is a polynomial in $F[t]$, of degree at most $5d-2$.

We want to prove that for each index n, we have $\Delta_n = 0$. For the sake of contradiction we assume that for some index r, $\Delta_r \neq 0$. Throughout the rest of

the proof we fix an arbitrary index r for which $\Delta_r \neq 0$. By the definition of Δ_r it follows that $x_r \neq 0$, so $x_r \Delta_r \neq 0$.

We write β for the least common multiple of the elements of the set $\{x_n \mid n \neq r\}$.

Claim 4 : We have $\deg(\beta) \leq 6d - 2$.
Claim 5 : We have

$$\prod_{n \neq r} x_n | \beta^2 \qquad \text{and} \qquad \deg(\beta) \geq \frac{M-2}{2} d$$

where $|$ means 'divides in $F[t]$'.

The proof of the Lemma follows. By Claims 4 and 5 we obtain

$$\frac{M-2}{2} d \leq \deg(\beta) \leq 6d - 2 \ ,$$

which can not hold if $M \geq 14$.

Lemma 20. *With the assumptions of Lemma 19 we have*

$$\nu' \neq 0 \ .$$

The proof is left to the reader.

Lemma 21. *With the assumptions of Lemma 19, for each index n, one of the following two statements is true :*
(a) There is a $\gamma_n \in F(t) \setminus \{0\}$ such that $\gamma_n' = 0$ and

$$\nu_n = \frac{x_n^2}{\gamma_n} + \gamma_n \tag{6}$$

(b) There is an $\epsilon_n = \pm 1$ and there is a δ_n such that $\delta_n' = 0$ and

$$\nu_n = 2\epsilon_n x_n + \delta_n \ . \tag{7}$$

Outline of proof: Rewrite Equations $\Delta_n = 0$ as

$$2\nu_n \nu_n' x_n' = x_n(\nu_n'^2 + 4x_n'^2)$$

and write $\frac{\nu_n}{x_n}$ by $\frac{\zeta_n}{\rho_n}$ and $\frac{\nu_n'}{x_n'}$ by $\frac{a_n}{b_n}$, for some polynomials a_n, b_n, ζ_n and ρ_n with $(a_n, b_n) = (\zeta_n, \rho_n) = (1)$, hence obtain

$$2\zeta_n a_n b_n = \rho_n(a_n^2 + 4b_n^2) \ ,$$

hence $\zeta_n = \beta_n(a_n^2 + 4b_n^2)$ and $\rho_n = 2\beta_n a_n b_n$ for some non-zero β_n in the base field F. Defining

$$\mu_n = \beta_n \frac{\nu_n}{\zeta_n}$$

express

$$\nu_n = \frac{1}{\beta_n}\zeta_n\mu_n = (a_n^2 + 4b_n^2)\mu_n \tag{8}$$

$$x_n = \frac{\rho_n}{\zeta_n}\nu_n = \frac{2\beta_n a_n b_n}{\beta_n(a_n^2 + 4b_n^2)}(a_n^2 + 4b_n^2)\mu_n = 2a_n b_n \mu_n \ . \tag{9}$$

Compute the derivatives ν' and x_n' in terms of μ_n, a_n, b_n and their derivatives. Conclude with

$$2[4b_n^2 - a_n^2]b_n'\mu_n = [-4b_n^2 + a_n^2]b_n\mu_n' \ .$$

The case $a_n^2 = 4b_n^2$ gives Equation (7). Otherwise obtain

$$\mu_n = \frac{\alpha_n}{b_n^2}$$

for some non-zero α_n which gives Equation (6).

Lemma 22. *With assumptions and notation as in Lemma 21 the following holds: if Equation (7) holds for some index r then we have $\delta_r = 0$.*

The proof is left to the reader.

Corollary 23. *With assumptions and notation as in Lemma 21 if for some r Equation (7) holds then it holds for all indices and the conclusion of Theorem 4 holds.*

The proof is left to the reader.

Lemma 24. *With assumptions and notation as in Lemma 21 we have: Equation (6) can not hold for any index.*

Outline of proof: By Lemma 23 Equation (6) holds for each index. Using Equation (6) and Equation (4) show that there is a $\gamma \in F(t^p)$ such that for each index n we have $\gamma_n = \gamma + n$, hence $x_n^2 = (\gamma + n)(\nu - \gamma + n)$. Show that $\gamma \in F[t]$ and, eventually, $\gamma \in F$.

Let k be an index other than m, h. Then, by Equation (6), the elliptic curve

$$Y^2 = (X + m)(X + h)(X + k)$$

has as solutions

$$(X, Y) = \left(\nu - \gamma, \frac{x_m}{\sqrt{\gamma + m}}\frac{x_h}{\sqrt{\gamma + h}}\frac{x_k}{\sqrt{\gamma + k}}\right)$$

which is impossible since an elliptic curve is of genus 1 and does not admit a non-constant rational parametrization.

Proof of Theorem 4. Clear by Lemmas 21, 22 and 24 and Corollary 23.

3 Consequences for Logic

Proof of Theorem 7 Our proof is an adjustment of the proof of Theorem 1.4 of [15].

We write $P_2(x)$ to mean 'x is a square in $F(t)$'. We consider a ring R as in the hypothesis of the Theorem. Let $\phi(z,w)$ denote the formula

$$\exists w_0,\dots,w_{13}\in R$$

$$[w=w_0\wedge 2z=w_1-w_0-1\bigwedge_{i=2,\dots,13} w_i+w_{i-2}=2w_{i-1}+2\bigwedge_{i=0,\dots,13}P_2(w_i)]\ .$$

Assume that $w=z^2$. Then it is trivial to see that $\phi(z,w)$ holds true by taking $w_i=(z+i)^2$.

Now assume that $\phi(z,w)$ is true. We claim that either $w=z^2$ or $w\in F$. For the sake of contradiction assume that $w\notin F$. Assume that the sequence $(w_n)_{n=0}^{13}=(x_n^2)_{n=0}^{13}$ satisfies the quantifier-free part of $\phi(z,w)$. Then $(w_n)_{n=0}^{13}$ satisfies the hypothesis of Theorem 4 and consequently $w_1=x_1^2=(\pm x_0+1)^2$, hence, since $2z=w_1-w_0-1$, we have

$$2z=x_1^2-x_0^2-1=(\pm x_0+1)^2-x_0^2-1=\pm 2x_0$$

and $w=z^2$.

Hence the following formula $\psi(z,w)$ is equivalent to $w=z^2$:

$$\psi(z,w)\colon \phi(z,w)\wedge\phi(tz,t^2w)\wedge\phi(z+t,w+2tz+t^2)\wedge\phi(t(z+t),t^2(w+2tz+t^2))$$

(the details are left to the reader). Thus squaring and, consequently, multiplication is positive-existentially definable in L_t.

The similar arguments for L_T hold with $\psi(z,w)$ replaced by

$$\bar{\phi}(z,w)\colon\ \exists w_0,\dots,w_{13}\in R$$

$$[w=w_0\wedge 2z=w_1-w_0-1\bigwedge_{i=2,\dots,13} w_i+w_{i-2}=2w_{i-1}+2\bigwedge_{i=0,\dots,13}P_2(w_i)\wedge T(w)]\ .$$

It is easily seen that Theorem 4 implies that $\bar{\phi}(z,w)$ is equivalent to $w=z^2$.

References

1. M. Davis, *Hilbert's tenth problem is unsolvable*, American Mathematical Monthly **80**, 233-269 (1973).
2. J. Denef, *The Diophantine Problem for polynomial rings and fields of rational functions*, Transactions of the American Mathematical Society **242**, 391-399 (1978).
3. —— *The diophantine problem for polynomial rings of positive characteristic*, Logic Colloquium **78**, (M. Boffa, D. van Dalen, K. McAloon editors), North Holland, Amsterdam, 131-145 (1979).
4. F. Grunewald and D. Segal, *How to solve a quadratic equation in integers*, Mathematical Proceedings of the Cambridge Philosophical Society **89**, 1-5 (1981).

5. F. Grunewald and D. Segal, *Some general algorithms I and II*, Ann. Math., 112 (1980), 531-617
6. R. Hartshorne, *Algebraic geometry*, Springer Verlag, Grad. texts in math. (1977).
7. K.H. Kim and F.W. Roush, *Diophantine undecidability of* $\mathbb{C}(t_1, t_2)$, Journal of Algebra **150**, 35-44 (1992).
8. ——*Diophantine unsolvability over p-adic function fields*, Journal of Algebra **176**, no. 1, 83-110. (1995).
9. S. Lang, *Elliptic functions*, Graduate Texts in Mathematics, Springer-Verlag, New York (1987).
10. L. Lipshitz, *Quadratic forms, the five square problem, and diophantine equations*, The collected works of J. Richard Büchi (S. MacLane and Dirk Siefkes, eds.) Springer, 677-680, (1990).
11. L. Lipshitz and T. Pheidas, *An analogue of Hilbert's Tenth Problem for p-adic entire functions*, The Journal of Symbolic Logic, **60-4**, 1301-1309, (1995).
12. Y. Matiyasevic, *Enumerable sets are diophantine*, Dokladii Akademii Nauk SSSR, **191** (1970), 279-282; English translation. Soviet Mathematics Doklady **11**, 354-358, (1970).
13. B. Mazur, *Questions of decidability and undecidability in number theory*, The Journal of Symbolic Logic **59-2**, 353-371, (1994).
14. T. Pheidas, *Hilbert's Tenth Problem for fields of rational functions over finite fields*, Inventiones Matematicae **103**, 1-8, (1991).
15. T. Pheidas and X. Vidaux, *Extensions of Büchi's problem : Questions of decidability for addition and k-th powers*, preprint
16. T. Pheidas and K. Zahidi, *Undecidable existential theories of polynomial rings and function fields*, Communications in Algebra **27(10)**, 4993-5010 (1999).
17. —— *Undecidability of existential theories of rings and fields : A survey*, Contemporary Mathematics **270**, 49-106 (1999).
18. B. Poonen, *Hilbert's Tenth Problem over rings of number-theoretic interest*, obtainable from http://math.berkeley.edu/ poonen/
19. L. Rubel, *An essay on diophantine equations for analytic functions*, Expositiones Mathematicae, **14**, 81-92, (1995).
20. A. Semenov, *Logical theories of one-place functions on the set of natural numbers*, Mathematics of the USSR-Izvestija **22**, 587-618 (1984).
21. —— *Hilbert's tenth problem over number fields, a survey*, Contemporary Mathematics **270**, 107-137 (2000).
22. X. Vidaux, *An analogue of Hilbert's 10th problem for fields of meromorphic functions over non-Archimedean valued fields*,aE Journal of Number Theory, **101**, 48-73, (2003).
23. C.R. Videla, *Hilbert's Tenth Problem for rational function fields in characteristic* 2, Proceedings of the American Mathematical Society **120-1**, 249-253 (1994).
24. P. Vojta, *Diophantine approximations and value distribution theory*, Lecture Notes in Mathematics, Springer-Verlag, **1239** (1987)
25. P. Vojta, *Diagonal quadratic forms and Hilbert's Tenth Problem*, Contemporary Mathematics **270**, 261-274 (2000).

On the Turing Degrees of Divergence Bounded Computable Reals*

Robert Rettinger[1] and Xizhong Zheng[2,3]

[1] Theoretische Informatik II, FernUniversität Hagen,
D-58084 Hagen, Germany
[2] Department of Computer Science, Jiangsu University,
Zhenjiang 212013, China
[3] Theoretische Informatik, BTU Cottbus,
D-03044 Cottbus, Germany
zheng@informatik.tu-cottbus.de

Abstract. The d-c.e. (difference of c.e.) and dbc (divergence bounded computable) reals are two important subclasses of Δ_2^0-reals which have very interesting computability-theoretical as well as very nice analytical properties. Recently, Downey, Wu and Zheng [2] have shown by a double witness technique that not every Δ_2^0-Turing degree contains a d-c.e. real. In this paper we show that the classes of Turing degrees of d-c.e., dbc and Δ_2^0 reals are all different.

1 Introduction

According to Turing [10], a real number x is *computable* if there is a computable function $f : \mathbb{N} \to \{0, 1, \cdots, 9\}$ such that $x = \sum_{n\in\mathbb{N}} f(n) \cdot 10^{-n}$. Equivalently, by Robinson [9] and others, a real $x \in [0; 1]$ is computable iff its Dedekind cut $L_x := \{r \in \mathbb{Q} : r < x\}$ is a computable set; iff x has a computable binary expansion in the sense that $x = x_A := \sum_{n\in A} 2^{-(n+1)}$ for a computable set $A \subseteq \mathbb{N}$; and iff there is a computable sequence (x_s) of rational numbers which converges to x *effectively* in the sense that $|x_s - x_{s+1}| \leq 2^{-s}$ for all s, and so on. Analogously, the Turing reducibility between real numbers and the Turing degree of a real can be equivalently defined based on binary expansion, Dedekind cut or Cauchy sequence representations, respectively (see e.g., [5]). For example, x_A is Turing reducible to x_B (denoted by $x_A \leq_T x_B$) if $A \leq_T B$ and $x_A \equiv_T x_B$ if $x_A \leq_T x_B$ & $x_B \leq_T x_A$. The Turing degree of a real x is defined as the class of all reals which are Turing equivalent to x, namely, $\deg_T(x) := \{y \in \mathbb{R} : y \equiv_T x\}$. Thus, the degree of a computable real consists of all computable reals. On the other hand, Ho [6] shows that a real is Turing reducible to $\mathbf{0}'$, the Turing degree of the halting problem, if and only if it is *computably approximable* (c.a.), i.e., it is the limit of a computable sequence of

* This work is supported by DFG (446 CHV 113/240/0-1) and NSFC (10420130638).

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 418–428, 2005.

rational numbers (see [1]). Since the binary expansion of a c.a. real is a Δ_2^0-set, the Turing degree of a c.a. real is also called a Δ_2^0-Turing degree or simply Δ_2^0-degree.

A lot of interesting classes of reals between computable and c.a. reals have been introduced in literature (see, [1, 8, 11]). For instance, a real x is *left* (*right*) *computable* if it is the limit of an increasing (decreasing) computable sequence of rational numbers. The left computable reals are also called computably enumerable (c.e., for short) (see [3, 4]) because their left Dedekind cuts are c.e. sets of rational numbers and they play a similar role in computable analysis as c.e. sets in classical computability theory. A real x is *d-c.e.* (difference of c.e.) if it is the difference of two c.e. reals. The class of d-c.e. reals is the arithmetical closure of c.e. reals and is actually a real closed field (see [1, 7]). More importantly, as shown in [1], a real x is d-c.e. iff there is a computable sequence (x_s) of rational numbers converging to x *weakly effectively* in the sense that the sum $\sum_{s\in\mathbb{N}} |x_s - x_{s+1}| \leq 1$. Therefore, d-c.e. reals are also called *weakly computable* (wc) and the class of all d-c.e. reals is denoted by **WC**.

Another interesting class of reals is introduced in [8] where a real x is called *divergence bounded computable* (dbc, for short) if there are a total computable function h and a computable sequence (x_s) of rational numbers which converges to x *h-bounded effectively*, i.e., for all n, there are at most $h(n)$ non-overlapping index pairs (i, j) such that $|x_i - x_j| \geq 2^{-n}$ (so-called 2^{-n}*-jumps*). The class of all dbc reals is denoted by **DBC**. It is shown in [8] that the class **DBC** is strictly between the classes of d-c.e. and computably approximable reals. Besides, like **WC**, the class **DBC** is also a real closed field. Furthermore, the class **DBC** is closed under total computable real functions while **WC** is not. Actually, **DBC** is the closure of **WC** under computable total real functions.

In the following, we call a Turing degree c.e., d-c.e. or dbc if it contains a c.e., d-c.e. or dbc real, respectively. In [12], it is shown that there is a d-c.e. Turing degree which does not even contain an ω-c.e. set. Here a set $A \subseteq \mathbb{N}$ is ω-c.e. if there are a computable function h and a computable sequence (A_s) of finite sets which converges to A such that $|\{s \in \mathbb{N} : A_s(n) \neq A_{s+1}(n)\}| \leq h(n)$ for all n and a Turing degree is called ω-c.e. if it contains an ω-c.e. set. Recently, Downey, Wu and Zheng showed in [2] that any ω-c.e. Turing degree contains at least a d-c.e. real, but there exists also a Δ_2^0-Turing degree which does not contain any d-c.e. real. That is, the class of d-c.e. degrees is strictly between the classes of ω-c.e. degrees and Δ_2^0-degrees.

In this paper, we extend the result of [2] and show that, even the class of dbc Turing degrees does not exhaust all Δ_2^0-Turing degrees. In [2], the double witness technique is used. Here we use a different construction technique. This technique closely connects the Turing reducibility with the Euclidean topology of $\mathbb{R}$ and has potentially more applications. For example, by this technique we can separate the class of dbc Turing degrees from the class of d-c.e. Turing degrees. Therefore, the classes of d-c.e. degrees, dbc degrees and Δ_2^0-degrees, respectively, are all different.

2 Preliminaries

Let $\Sigma := \{0,1\}$ be a binary alphabet. For convenience we identify a real $x \in [0,1]$ with its binary characteristic sequence $x \in \Sigma^\omega$ and identify a dyadic rational number $r \in [0,1]$ with its binary characteristic string $r \in \Sigma^*$. For any binary string w, an open interval of w is defined by $I(w) := w\Sigma^\omega \setminus \{w1^\omega, w0^\omega\}$. For $I := I(w)$ and $n \in \mathbb{N}$, we define two subintervals $L_n(I) := I(w00^n1)$ and $R_n(I) := I(w10^n1)$. Let I, J be any intervals, their minimal and maximal distances are denoted by $d(I,J) := \min\{|x-y| : x \in I \,\&\, y \in J\}$ and $D(I,J) := \max\{|x-y| : x \in I \,\&\, y \in J\}$, respectively. Given two intervals $I_1 := I(w_1)$ and $I_2 := I(w_2)$ of a distance $d(I_1, I_2) = \delta > 0$ and a number n, let w_0 be the shortest string such that $|w_0| \geq n$ and $d(I_1, I(w_0)) = d(I(w_0), I_2) > 2^{-n}$. Thus the interval $I(w_0)$ has at most a length 2^{-n} and locates exactly in the middle between I_1 and I_2. This interval $I(w_0)$ is denoted by $M_n(I_1, I_2)$. The middle point of an interval I is denoted by $\mathrm{mid}(I)$.

Let (N_e^A) be a computable enumeration of all Turing machines with oracle A and suppose that N_e^A computes the computable functional Φ_e^A. By definition, a real x is Turing reducible to y if there is an $i \in \mathbb{N}$ such that $x = \Phi_i^y$ i.e., $x(n) = \Phi_i^y(n)$ hold for all $n \in \mathbb{N}$. Thus, in order to construct two non-Turing equivalent reals x, y we have to guarantee that $x \neq \Phi_i^y \;\&\; y \neq \Phi_j^x$ for all i, j. To this end, we define a "length function" recording the maximal temporal agreement between (x, y) and (Φ_i^y, Φ_j^x) and try to destroy this agreement if it is possible to keep it finite. This is usually quite complicated if we consider all (i, j) in a priority construction. The following observation will simplify the matter a lot. If $x = \Phi_i^y$, then there is another computable functional Φ_j such that $x \upharpoonright n = \Phi_j^y(n)$ for all n, where $x \upharpoonright n$ is the initial segment of x of length n. (Here we identify a natural number n with the n-th binary word under the length-lexicographical ordering.) Thus, two reals x and y are Turing equivalent iff there are $i, j \in \mathbb{N}$ such that x is (i,j)-*Turing equivalent* to y (denoted by $x \equiv_T^{(i,j)} y$) in the following sense

$$(\forall n \in \mathbb{N}) \left(x \upharpoonright n = \Phi_i^y(n) \;\&\; y \upharpoonright n = \Phi_j^x(n) \right). \tag{1}$$

Although this is only a simple variation of usual Turing equivalence of the form $x = \Phi_i^y \;\&\; y = \Phi_j^x$, it connects the Turing equivalence to the topological structure of $\mathbb{R}$. To see that, we show the following lemma which is essential for the proofs of our main results later.

Lemma 1. *Given an open interval $I_0 \subseteq \Sigma^\omega$, $i, j, t \in \mathbb{N}$, there exist two open intervals $I \subseteq I_0$ and $J \subseteq \Sigma^\omega$ of at most length 2^{-t} such that*

$$(\forall x, y) \left((x \in I \wedge x \equiv_T^{(i,j)} y \Longrightarrow y \in J) \;\&\; (y \in J \wedge x \equiv_T^{(i,j)} y \Longrightarrow x \in I_0) \right). \tag{2}$$

Notice that, if the intervals I, J satisfy (the first part of) condition (2), then any real of I can only be (i,j)-Turing equivalent to a real of J. In other words, if the reals x, y satisfy $x \in I \;\&\; y \notin J$, then they are not (i,j)-Turing equivalent! Thus, to avoid the constructed real x being (i,j)-Turing equivalent to a given

real y, it suffices to fix two interval pairs (I_l, J_l) and (I_r, J_r) of this property, then choose x from I_l whenever y seems not in J_l and change x to I_r if y enters J_l and so on. This is simply the usual "jump trick". For convenience, we call an interval I (i,j)-*reducible* to J (denoted by $I \prec_{(i,j)} J$) if, for all x, y, $x \in I \wedge x \equiv_T^{(i,j)} y \Longrightarrow y \in J$. The second part of (2) guarantees that, if two distinct I_0-intervals are give, then the corresponding J-intervals are also distinct.

The lemma 1 can be strengthened to the following effective version.

Corollary 2. *There exists a partial computable $\theta :\subseteq \mathbb{N}^3 \times \Sigma^* \to (\Sigma^*)^2$ so that if there exist $x \in I(w)$ and $y \in \Sigma^\omega$ with $x \equiv_T^{(i,j)} y$, then $\theta(i,j,t,w) \downarrow = (u,v)$ and the intervals $I := I(u)$ and $J := I(v)$ satisfy Lemma 1 for $I_0 := I(w)$.*

Later on, $\theta(i,j,n,w)$ is also denoted by $\theta(i,j,2^{-n}, I(w))$. Suppose that M_θ is a Turing machine which computes the function θ. Then, for any s, θ_s is the function computed by M_θ up to step s.

3 A Δ_2^0-Degree Containing no DBC Reals

In [2] it is shown that not every Δ_2^0-Turing degree contains a d-c.e real. Since the class of dbc reals is a proper superset of the class of d-c.e. reals, it is natural to ask, if this result can be extended to the case of dbc reals. Namely, whether there exists a Δ_2^0-Turing degree which contains no dbc reals. The next theorem gives a positive answer of this question.

Theorem 3. *There exists a c.a. real which is not Turing equivalent to any dbc real, i.e., not every Δ_2^0-Turing degree contains a dbc real.*

Proof. We first prove the theorem by the double witnesses technique of [2]. That is, we construct a computable sequence (A_s) of finite subsets of natural numbers which converges to A such that A is not Turing equivalent to any divergence bounded computable real. To this end, let $(b_e, h_e, \Phi_e, \Psi_e)$ be an effective enumeration of all tuples of computable functions $b_e :\subseteq \mathbb{N} \to \mathbb{D}$, $h_e :\subseteq \mathbb{N} \to \mathbb{N}$, and computable functionals Φ_e, Ψ_e. For any $e, s \in \mathbb{N}$, if $b_e(s)$ is defined, then let $B_{e,s}$ be a finite set of natural numbers such that $b_e(s) = x_{B_{e,s}}$. Thus, the set A has to satisfy all the following requirements.

$$R_e: \quad \left.\begin{array}{l} b_e \text{ and } h_e \text{ are total functions and } (b_e(s)) \\ \text{converges } h_e\text{-bounded effectively to } x_{B_e} \end{array}\right\} \Longrightarrow A \neq \Phi_e^{B_e} \vee B_e \neq \Psi_e^A.$$

Let A_s be the approximation of A constructed at the end of stage s. We define a length function l as follows:

$$l(e,s) := \max\{x : A_s \upharpoonright x = \Phi_{e,s}^{B_{e,s}} \upharpoonright x \;\&\; B_{e,s} \upharpoonright \varphi_{e,s}(x) = \Psi_{e,s}^{A_s} \upharpoonright \varphi_{e,s}(x)\},$$

where φ_e is the use function of the functional Φ_e.

To satisfy a requirement R_e, it suffices to guarantee that $l(e,s)$ is bounded from above. To this end, we first choose a witness n_e large enough. At the

beginning, let $A(n_e-1)(n_e) = 00$. If there does not exist s such that $l(e,s) > n_e$, then we are done. Otherwise, suppose that $l(e,s_1) > n_e$ for an s_1. Let $m_e := \psi_{e,s_1}(\varphi_{e,s_1}(n_e))$. Assume w.l.o.g. that $n_e < m_e$. If $h_{e,s_1}(m_e)$ is also defined, then let $A_{s_1+1}(n_e-1)(n_e) = 10$. Wait for a new stage $s_2 > s_1$ such that $l(e,s_2) > n_e$ holds again. If no such a stage exists, then we are done again. Otherwise, let $A_{s_2+1}(n_e-1)(n_e) := 11$. If there exists another stage $s_3 > s_2$ such that $l(e,s_3) > n_e$, then let $A_{s_3+1}(n_e-1)(n_e) := 00$. In this case, the set A_{s_3+1} is recovered to that of stage s_1, i.e., $A_{s_3+1} = A_{s_1}$. This closes a cycle in which the values $A(n_e-1)(n_2)$ change in the order of $00 \to 10 \to 11 \to 00$. This process will continue as long as the number of 2^{-m_e}-jumps of the sequence $(x_{B_{e,s}})$ (i.e., the sequence $(b_e(s))$) does not exceed $h_e(m_e)$ yet.

Thus, we achieve a temporary disagreement between A and $\Phi_e^{B_e}$ by changing the values $A(n_e-1)(n_e)$ whenever the length of agreement goes beyond the witness n_e. After that, if the agreement becomes bigger than n_e again, then the corresponding value $\Phi_e^{B_e}(n_e-1)(n_e)$ has to be changed too and this forces the initial segment $B_e \upharpoonright \varphi_e(n_e)$ to be changed, say, $B_{e,s} \upharpoonright \varphi_{e,s}(n_e) \neq B_{e,t} \upharpoonright \varphi_{e,t}(n_e)$. There are two possibilities now.

Case 1. $|x_{B_{e,s}} - x_{B_{e,t}}| \geq 2^{-m_e}$. If the sequence $(b_e(s))$ converges h_e-bounded effectively, then $(b_e(s))$ has at most $h_e(m_e)$ non-overlapping 2^{-m_e}-jumps. Thus, this can happen at most $h_e(m_e)$ times.

Case 2. $|x_{B_{e,s}} - x_{B_{e,t}}| = 2^{-m} < 2^{-m_e}$ for an $m > m_e$. Let n be the least natural number $n < m_e$ such that $B_{e,s}(n) \neq B_{e,t}(n)$. In this case, as binary word, $B_{e,s}$ has either forms $0.w10\cdots0v$ or $0.w01\cdots1v$ for some $w, v \in \{0,1\}^*$ and $B_{e,t}$ takes another one. This implies that, if the sequence $(x_{B_e(s)})$ does not have 2^{-m_e}-jumps after some stage s any more, then the initial segment $B_{e,s} \upharpoonright m_e$ can have only two possible forms: $0.w10\cdots0$ or $0.w01\cdots1$. Correspondingly, the combination $\Phi_{e,s}^{B_{e,s}}(n_e-1)(n_e)$ can have at most two possibilities too. However, in every circle described above, $A(n_e-1)(n_e)$ takes three different forms, i.e., $00, 10$ and 11. In other words, we can always achieve a disagreement $A \neq \Phi_e^B$ at some stage and hence the requirement R_e is satisfied eventually.

To satisfy all requirements simultaneously, we apply a finite injury priority construction. The details are omitted here.

In the following, we give another proof based on the "interval-jumping" technique. This technique will be used again in the next section.

A New Proof. We will construct a computable sequence (x_s) of rational numbers converging to a real x which satisfies, for all $i,j,k \in \mathbb{N}$, the following requirements.

$$R_{\langle i,j,k\rangle}: \quad \left.\begin{array}{l}\varphi_k \text{ and } h_k \text{ are total functions and } (\varphi_k(s)) \\ \text{converges } h_k\text{-bounded effectively to } y_k\end{array}\right\} \Longrightarrow x \not\equiv_T^{(i,j)} y_k,$$

where (φ_k, h_k) is a computable enumeration of all pairs of partial computable functions $\varphi_k :\subseteq \mathbb{N} \to \mathbb{D}$ and $h_k :\subseteq \mathbb{N} \to \mathbb{N}$.

To satisfy a single requirement R_e for $e = \langle i,j,k\rangle$, we fix a base interval I_{e-1} and try to find a *witness interval* $I_e \subseteq I_{e-1}$ such that any real $x \in I_e$ satisfies R_e. As the first candidate, let $I_e := R_1(I_{e-1})$. If no real of this interval is (i,j)-

Turing equivalent to some real, then we are done. Otherwise, by Corollary 2, we can find an interval pair (I_r, J_r) such that $I_r \subseteq I_e$ and $I_r \prec_{(i,j)} J_l$. Then let $I_e := L_1(I_{e-1})$. Analogously, either I_e is already a correct witness interval or we can find another interval pair (I_l, J_l) such that $I_l \subseteq I_e$ and $I_l \prec_{(i,j)} J_l$. Suppose that we have obtained two such interval pairs. We can choose I_r or I_l as witness interval depending on whether the sequence $(\varphi_k(s))$ enters the interval J_l or J_r. This trick works if the intervals J_l and J_r are not connected, i.e., $d(J_l, J_r) \geq \delta_e > 0$. In this case, let $n_e := (\mu n)(2^{-n} \leq \delta_e)$. Then at most $h_k(n_e)$ jumps suffice if the sequence $(\varphi_k(s))$ converges h_k-bounded effectively.

To guarantee $d(J_l, J_r) > 0$ we introduce a third interval pair: Let I_e be a new interval between the intervals I_l and I_r, say $I_e := M_e(I_l, I_r)$. Again, either I_e is a correct witness interval or we can get a third interval pair (I_m, J_m) such that $I_m \subseteq I_e$ and $I_m \prec_{(i,j)} J_m$. Since, by condition (2), three intervals J_l, J_m and J_l cannot be overlapping and hence at least two of them have a positive minimal distance by which we can apply the normal jump trick comfortably. To satisfy all requirements simultaneously, a finite injury priority construction suffices.

Notice that, in the double-witnesses construction, a reasonable upper bound of the number of changes among the candidate intervals is not available and hence the constructed number x is computably approximable. However, in the construction based on the interval-jumping technique, we consider two interval pairs (I_l, J_l) and (I_r, J_r) and choose x from I_l or I_r whenever y enters J_r or J_l, respectively. In this case, it is possible to estimate the number of necessary jumps of x by calculate the allowed number of y jumps. This idea can be used to separate the classes of Turing degrees of dbc reals and d-c.e. reals in the next section.

4 Separating DBC-Degrees from D-C.E. Degrees

In this section we will show that the classes of d-c.e. and dbc degrees are different and hence Theorem 3 extends the result of [2] properly.

Theorem 4. *There exists a Turing degree of divergence bounded computable reals which does not contain any d-c.e. reals.*

Proof. We construct a computable sequence (x_s) of rational numbers which converges h-bounded effectively to x for some computable function h such that the limit x is not Turing equivalent to any d-c.e. reals. Since any d-c.e. real is the limit of a computable sequence (y_s) of rational numbers such that $\sum_{s\in\mathbb{N}} |y_s - y_{s+1}| \leq 1$, it suffices to satisfy, for all $i, j, k \in \mathbb{N}$, the following requirements

$$R_{\langle i,j,k\rangle}: \qquad \sum_{s\in\mathbb{N}} |\varphi_k(s) - \varphi_k(s+1)| \leq 1 \;\&\; \lim_{s\to\infty} \varphi_k(s) = y_k \Longrightarrow x \not\equiv_T^{(i,j)} y_k,$$

where (φ_e) is a computable enumeration of all partial computable functions $\varphi_e :\subseteq \mathbb{N} \to \mathbb{D}$.

To satisfy a single requirement R_e for $e = \langle i, j, k\rangle$, we use a "jump trick" as in the proof of Theorem 3. Namely, choose two interval pairs (I_l, J_l) and (I_r, J_r) with $d(J_l, J_r) := \delta > 0$ according to Corollary 2 such that $I_l \prec_{(i,j)} J_l$ and $I_r \prec_{(i,j)} J_r$. If such interval pairs do not exist, then we can obtain an interval I such that no reals of I is (i, j)-Turing equivalent to any real. In this case, the interval I is a *witness interval* of R_e because any real x from I satisfies R_e and we are done. Otherwise, depending on whether the sequence $(\varphi_k(s))$ enters the interval J_l or J_r, we can choose x_s's from I_r or I_l accordingly. If the sequence $(\varphi_k(s))$ satisfies the condition $\sum_{s\in\mathbb{N}} |\varphi_k(s) - \varphi_k(s+1)| \leq 1$, then the inequality $x := \lim_s x_s \neq y_k := \lim_s \varphi_k(s)$ can be achieved by at most $1/\delta$ jumps of the sequence (x_s) between the intervals I_l and I_r and R_e can be satisfied.

Unfortunately, this strategy does not guarantee that the constructed sequence (x_s) converges h-bounded effectively for some computable function h, because the number of required jumps depends only on the distance $\delta := d(J_l, J_r)$ but not on the size of the jumps themselves which are bounded by $\Delta := D(I_l, I_r)$. To solve this problem, we look for new interval pairs such that their Δ and δ are related reasonably.

Suppose that we have two interval pairs (I_l, J_l) and (I_r, J_r) with $I_l \prec_{(i,j)} J_l$ and $I_r \prec_{(i,j)} J_r$. Let $a := \max\{n : D(I_l, I_r) \leq 2^{-n}\}$ and $b := (\mu n)(d(J_l, J_r) \geq 2^{-n})$. Then the above strategy contributes at most 2^b jumps of a distance up to 2^{-a} to the sequence (x_s). Suppose in addition that $l(I_l), l(I_r) \leq D(I_l, I_r)/8$. Let I_e be an interval of length $l(I_e) \leq d(I_l, I_r)/8$ which locates in the middle between I_l and I_j, then $\max\{D(I_e, I_l), D(I_e, I_r)\} \leq D(I_l, I_r)\cdot 5/8 \leq 2^{-a}\cdot 2^{-1/2} \leq 2^{-(a+1/2)}$. If we can find intervals $I_m \subseteq I_e$ and J_m of length $l(J_m) \leq d(J_l, J_e)/4$ according to Lemma 1 such that $I_m \prec_{(i,j)} J_m$, then we have $\max\{d(J_r, J_m), d(J_l, J_m)\} \geq d(J_l, J_r)/4 \geq 2^{-(b+2)}$. (If such J_m do not exit, then no real of I_m is (i, j)-Turing equivalent to other reals and we are done). Suppose that $d(J_r, J_m) \geq d(J_l, J_m)$ (the other case can be treated similarly), then we have $d(J_m, J_r) \geq 2^{-(b+2)}$ and $D(I_m, I_l) \leq 2^{-(a+1/2)}$.

Now, let (I_m, J_m) and (I_r, J_r) be our new interval pairs which are denoted by (I_l^1, J_l^1) and (I_r^1, J_r^1), respectively. This procedure is called an "interval length reduction". If we repeat this procedure n times, then we possibly arrive at the interval pairs (I_l^n, J_l^n) and (I_r^n, J_r^n) such that

1. $I_l^n \prec_{(i,j)} J_l^n$ and $I_r^n \prec_{(i,j)} J_r^n$, and
2. $D(I_l^n, I_r^n) \leq 2^{-(a+n/2)}$ and $d(J_l^n, J_r^n) \geq 2^{-(b+2n)}$.

If we stop in between, then we have an interval, say I_m^t, such that no reals of this interval is (i, j)-Turing equivalent to any real and hence we are done. Otherwise, fix an n large enough such that $2^{8(a+n/2)} \geq 2^{b+2n}$ and apply the "jump trick" for the interval pairs (I_l^n, J_l^n) and (I_r^n, J_r^n). In this case, each jump of (x_s) (between new I_l and I_r) has at most a distance $2^{-(a+n/2)}$ while each jump of $(\varphi_k(s))$ (between new J_l and J_r) has at least a distance $2^{-(b+2n)}$. Because of the condition $\sum_{s\in\mathbb{N}} |\varphi_k(s) - \varphi_k(s+1)| \leq 1$, we need at most $2^{b+2n} \leq 2^{8(a+n/2)}$ jumps of a distance up to $2^{-(a+n/2)}$. In other words, 2^{8t} jumps of a length up to 2^{-t} suffice for $t = a + 2/n$. Actually, for smaller t the above strategy contributes

no more jumps than 2^{8t} of the distance up to 2^{-t}. If we fix a base interval I in advance for R_e of at most length 2^{-e} and choose the intervals I_l and I_r only from I, then the improved strategy for R_e satisfies the following conditions

(A) It causes only 2^{-n}-jumps for $n \geq e$; and
(B) It contributes 2^{-n}-jumps at most 2^{8n} times totally for any n.

By a priority technique we can combine above strategies for all R_e's together and construct a computable sequence (x_s) of rational numbers whose limit x satisfies all R_e simultaneously. If we can arrange that conditions (A) and (B) are still satisfied, then the sequence (x_s) converges 2^{9n}-bounded effectively. The reason is, for any n, a 2^{-n}-jump can only be caused by a requirement R_e for $e < n$ and the action for R_e contributes at most 2^{8n} 2^{-n}-jumps. Therefore, the total number of 2^{-n}-jumps is not larger than 2^{9n} and hence x is divergence bounded computable.

Let's give a formal construction of the sequence (x_s) now. At any stage s, all requirements are appointed a state from the following set:

$$S := \{\text{waiting, prepare}(t), \text{reduce}(n), \text{jump}(n), \text{satisfied} : t \leq 4 \ \& \ n \in \mathbb{N}\}.$$

At stages s, there is a number c_s such that all R_e for $e > c_s$ are in the state "waiting". For any $e \leq c_s$, R_e is in the other state and is appointed a parameter $\epsilon_e > 0$, a base interval I_{e-1}, four supplementary intervals $I_{e,l}$, $I_{e,r}$, $J_{e,l}$, $J_{e,r}$ and a witness interval I_e which is at the same time the base interval for R_{e+1}. The middle point of I_{c_s} is defined as x_s. Besides, R_e is appointed two parameters n_e and m_e whenever it arrives the state "prepare(3)", and an additional z_e when it obtains the state "jimp(0)". If it is necessary, these parameters can be appended by $[s]$ (for instant $I_{e,l}[s]$) to denote its actual values at stage s. In the construction, all parameters which are not redefined explicitly at stage $s+1$ remain the same as that of stage s.

The formal construction:

Stage $s = 0$: Let $I_{-1} := I(0)$ and define

$$(I_{0,l}, I_{0,r}, J_{0,l}, J_{0,r}, I_0) := (L_1(I_{-1}), R_1(I_{-1}), \emptyset, \emptyset, R_1(I_{-1})).$$

and $\epsilon_0 := l(I_0)/2$. Set R_0 to state "prepare(0)". All other R_e for $e > 0$ are initialized by setting its state to "waiting" and all of its parameters to undefined.

Stage $s+1$: A requirement R_e for $e = \langle i, j, k\rangle$ *requires* attention if either

R1: R_e is not in the state "waiting" or "jump(t)" for any t and $\theta_s(i, j, \epsilon_e, I_e) \downarrow = (u, v)$, or
R2: R_e is in the state "jump(t)" for some $t \in \mathbb{N}$ and there exists a $z > z_e[s]$ such that $\varphi_{k,s}(z) \downarrow = q \in J_e$ where $J_e = J_{e,l}$ if $I_e = I_{e,l}$ or $J_e = J_{e,r}$ otherwise.

If no requirement requires attention, then let R_e be the requirement of highest priority which is in the state waiting. Thus, $I_{e-1} = I(w)$ is defined for some $w \in \Sigma^*$. Let n be the minimal number such that 2^{-n+1} is less than all jumps of the sequence $(x_t)_{t \leq s}$. Then define

$$(I_{e,l}, I_{e,r}, J_{e,l}, J_{e,r}, I_e) := (L_n(I_{e-1}), R_n(I_{e-1}), \emptyset, \emptyset, R_n(I_{e-1})). \tag{3}$$

Let $\epsilon_e := l(I_e)/2$ and set R_e into the state "prepare(0)".

Otherwise, suppose that R_e is the requirement of the highest priority which requires attention. We consider the following cases.

Case 1. R_e is in the state "prepare(0)". Define

$$(I_{e,r}, J_{e,r}, I_e)[s+1] := (I(u), I(v), I_{e,l}[s]).$$

Let $\epsilon_e := l(I_e)/2$ and set R_e to the state "prepare(1)".

Case 2. R_e is in the state "prepare(1)". Redefine $(I_{e,l}, J_{e,l}) := (I(u), I(v))$. Let $t := (\mu t)(d(I_{e,l}, I_{e,r}) \geq 2^{-t})$. Then define further $I_e := M_{t+1}(I_{e,l}, I_{e,r})$, $\epsilon_e := l(I_e)/2$ and set R_e to the state "prepare(2)".

Case 3. R_e is in the state "prepare(2)". Now we have three "(i,j)-reducible" intervals pairs $(I_{e,l}, J_{e,l})$, $(I_{e,r}, J_{e,r})$ and $(I(u), I(v))$. The intervals $J_{e,l}, J_{e,r}$ and $I(v)$ are obviously distinct. This means that at least one of $d(J_{e,l}, J_{e,r})$, $d(J_{e,l}, I(v))$ or $d(J_{e,r}, I(v))$ is positive. Let δ_e be the maximal one of them and $b_e := (\mu n)(\delta_e \geq 2^{-n})$. Redefine

$$\begin{cases} (I_{e,r}[s+1], J_{e,r}[s+1]) := (I_e[s], I(v)) & \text{if } d(J_{e,l}, I(v)) = \delta_e; \\ (I_{e,l}[s+1], J_{e,l}[s+1]) := (I_e[s], I(v)) & \text{if } d(J_{e,r}, I(v)) = \delta_e. \end{cases}$$

(For the case $d(J_{e,l}, J_{e,r}) = \delta_e$, nothing should be changed.) That is, the new intervals $J_{e,l}[s+1]$ and $J_{e,r}[s+1]$ achieve the largest distance δ_e. Choose a maximal $a \in \mathbb{N}$ such that $D(I_{e,l}[s+1], I_{e,r}[s+1]) \leq 2^{-a}$. Now R_e is ready for the procedure of "interval-length reduction". As mentioned before, the number n_e of required iterations is determined by $2^{8(a+n/2)} \geq 2^{b+2n}$ and therefore we define $n_e := \lceil (b-8a)/2 \rceil$. Remember that the reduction produce demands on the length of the I-intervals to be relative small compared to their maximal distance. Furthermore, the procedure inserts a new interval between the I-intervals and possibly also a new interval between the J-intervals. To guarantee this procedure can be iterated n times without confliction, all interval length should be small enough relative to their minimal distance. More precisely, all intervals can have a length at most 2^{-m_e} where m_e is the least natural number satisfying

$$2^{-m_e} \leq \min\{2^{-b_e},\ 2^{-d_e}\} \cdot 4^{-(n_e+2)} \tag{4}$$

where $d_e = (\mu t)(2^{-t} \leq l(I_{e,l}, I_{e,r}))$. Now let $I_e := I_{e,r}$ and $\epsilon_e := 2^{-m_e}$ and set R_e into the state "prepare(3)".

Case 4. R_e is in the state "prepare(3)". Define

$$(I_{e,r}, J_{e,r}, I_e)[s+1] := (I(u), I(v), I_{e,l}[s])$$

and set R_e to the state "prepare(4)".

Case 5. R_e is in the state "prepare(4)". Redefine $(I_{e,l}, J_{e,l}) := (I(u), I(v))$. Then define further $I_e := M_{m_e}(I_{e,l}[s+1], I_{e,r}[s+1])$ and set R_e to the state "reduce(0)".

Case 6. R_e is in the state "reduce(t)". Redefine

$$\begin{cases} (I_{e,l}[s+1], J_{e,l}[s+1]) := (I(u), I(v)) & \text{if } d(J_{e,r}, I(v)) \geq d(J_{e,l}, I(v)); \\ (I_{e,r}[s+1], J_{e,r}[s+1]) := (I(u), I(v)) & \text{otherwise.} \end{cases}$$

Notice that the distances of new and old intervals have the following relation:

$$D(I_{e,l}[s+1], I_{e,r}[s+1]) \leq D(I_{e,l}[s], I_{e,r}[s]) \cdot 2^{-1/2} \tag{5}$$
$$d(J_{e,l}[s+1], J_{e,r}[s+1]) \geq d(J_{e,l}[s], J_{e,r}[s]) \cdot 2^{-2}. \tag{6}$$

If $t < n_e$, then define further $I_e := M_{m_e}(I_{e,l}[s+1], I_{e,r}[s+1])$ and set R_e into the state "reduce$(t+1)$". Otherwise, if $t = n_e$, then let $I_e := I_{e,r}$, $z_e := -1$ and set R_e into the state "jump(0)".

Case 7. R_e is in the state "jump(t)". Redefine

$$I_e := \begin{cases} I_{e,l}, & \text{if } \varphi_{k,s}(z) \in J_{e,r}; \\ I_{e,r}, & \text{if } \varphi_{k,s}(z) \in J_{e,l}. \end{cases}$$

where z is the number satisfying the requiring condition R2. If $t \cdot d(J_{e,l}, J_{e,r}) < 1$, then let $z_e[s+1] := z$ and set R_e into the state "jump$(t+1)$". Otherwise set R_e into the state "satisfied".

In all these cases, we say that R_e *receives attention*. In addition, if R_e receives attention, then all requirements $R_{e'}$ for $e' > e$ are *initialized* by setting it into the state "waiting" and letting all its parameters to be undefined. If $R_{e'}$ is not in the state "waiting" at stage s, then $R_{e'}$ is *injured* at this stage.

This ends the construction. We can show that our construction succeeds. That is, the computable sequence (x_s) converges to a dbc real x which is not Turing equivalent to any d-c.e. real.

References

1. K. Ambos-Spies, K. Weihrauch, and X. Zheng. Weakly computable real numbers. *Journal of Complexity*, 16(4):676–690, 2000.
2. R. Downey, G. Wu, and X. Zheng. Degrees of d.c.e. reals. *Mathematical Logic Quartely*, 50(4/5):345–350, 2004.
3. R. G. Downey. Some computability-theoretical aspects of real and randomness. Preprint, September 2001.
4. R. G. Downey and D. R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Springer-Verlag, 200? monograph to be published.
5. A. J. Dunlop and M. B. Pour-El. The degree of unsolvability of a real number. In *Proceedings of CCA 2000, Swansea, UK, September 2000*, Berlin, 2001. Springer.
6. C.-K. Ho. Relatively recursive reals and real functions. *Theoretical Computer Science*, 210:99–120, 1999.
7. A. Raichev. D.c.e. reals, relative randomness, and real closed fields. In *CCA 2004, August 16-20, 2004, Lutherstadt Wittenberg, Germany*, 2004.
8. R. Rettinger, X. Zheng, R. Gengler, and B. von Braunmühl. Weakly computable real numbers and total computable real functions. In *Proceedings of COCOON 2001, Guilin, China, August 20-23, 2001*, volume 2108 of *LNCS*, pages 586–595. Springer-Verlag, 2001.
9. R. M. Robinson. Review of "Peter, R., Rekursive Funktionen". *The Journal of Symbolic Logic*, 16:280–282, 1951.

10. A. M. Turing. On computable numbers, with an application to the "Entscheidungsproblem". *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.
11. X. Zheng. Recursive approximability of real numbers. *Mathematical Logic Quarterly*, 48(Suppl. 1):131–156, 2002.
12. X. Zheng. On the Turing degrees of weakly computable real numbers. *Journal of Logic and Computation*, 13(2):159–172, 2003.

New Algorithmic Paradigms in Exponential Time Algorithms

Uwe Schöning

Abteilung Theoretische Informatik,
Universität Ulm,
89069 Ulm, Germany
schoenin@informatik.uni-ulm.de

Difficult combinatorial problems typically require exponential time algorithms. Recently there is a new point of view that algorithms operating in (worst-case) exponential time can still be quite useful if the constants in the exponential complexity function are small enough. On the other hand, we do not know many algorithmic design principles for such algorithms up to now. This talk discusses various ways of designing good exponential time algorithms which reduce the value of the respective constants.

It seems that randomization is a particularly successful concept in the context of exponential-time algorithms. It turns out that the kind of randomization needed in exponential algorithms differs from the known examples of randomization in the context of polynomial-time algorithms.

Our running examples in this talk will be NP-complete problems like 3-satisfiability or 3-colorability.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, p. 429, 2005.

Some Reducibilities on Regular Sets*

Victor L. Selivanov

A.P. Ershov Institute of Informatics Systems,
Siberian Division of the Russian Academy of Sciences, Russia
vseliv@nspu.ru

Abstract. We discuss some known and introduce some new reducibilities on regular sets. We establish some facts on the corresponding degree structures and relate some reducibilities to natural hierarchies of regular sets. As an application, we characterize regular languages whose leaf-language classes (in the balanced model) are contained in the polynomial hierarchy. For any reducibility we try to give some motivation and interesting open questions, in a hope to convince the reader that study of these reducibilities is important for automata theory and computational complexity.

1 Introduction

The notion of reducibility appeared in computability theory and plays a central role there . Afterwards, different notions of reducibility were employed in different branches of computation theory and of definability theory (in descriptive set theory this is Wadge reducibility, in complexity theory — polynomial-time m-reducibility, in finite model theory — logical reducibilities [8], and so on). Some of these reducibilities turned out to be also quite important for the corresponding fields.

More recently, people began to consider reducibilities inducing nontrivial degree structures on regular sets (i.e., on languages recognized by finite automata) [3, 19, 17, 5]. In this paper, we continue to discuss some known and introduce some new reducibilities on regular sets. We establish some facts on the corresponding degree structures and relate the reducibilities to natural hierarchies of regular sets. For any reducibility we try to give some motivation and interesting open questions, in a hope to convince the reader that study of these reducibilities is important for automata theory and complexity theory.

We use (mostly without definitions here) some standard terminology and notation from computability theory, automata theory and complexity theory, say

* This paper is a part of [13]. I am grateful to Klaus Wagner for hosting my stay at the University of Würzburg in pre-Christmas days of 2004, and to him and Christian Glaßer for helpful discussions. Thanks are also due to anonymous referees for the careful reading.

The author is supported by the Alexander von Humboldt Foundation.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 430–439, 2005.

terminology on reducibilities and degrees, notation of languages by regular expressions or the concept of a polynomial-time non-deterministic Turing machine. Letters A, B will denote alphabets which are always assumed to contain at least two symbols. By A^+ we denote the set of all non-empty words over A, and by A^* the set of all words (including the empty word ε). Since usually we work with a fixed alphabet A, we normally do not mention the alphabet explicitly. The length of a word w is denoted $|w|$. Since we use the logical approach to regular languages [4, 9, 18], we work mostly with languages of non-empty words $L \subseteq A^+$. Correspondingly, the complement $\overline{L}$ of such a language L is defined by $\overline{L} = A^+ \setminus L$. As usual, $P(A^+)$ denotes the power set of A^+. By $P'(A^+)$ we denote the class of all non-trivial (i.e. distinct from $\emptyset$ and A^+) subsets of A^+. By $\mathcal{R}$ ($\mathcal{R}'$) we denote the class of all regular (resp., regular non-trivial) languages over A.

In Section 2 we discuss two important reducibilities closely related to so called leaf language approach to complexity classes which is described in detail in a recent survey [21]. In Section 3 we remind the reader of some known facts and state some new facts related to the logical approach to automata theory. In Section 4 we generalize most results from [17] and establish some new facts about versions of the quantifier-free reducibility. In Section 5 we present some results on first-order reducibilities. We conclude in Section 6 with mentioning some other reducibilities and open questions.

2 plt- and ptt-Reducibilities

A language $L \subseteq A^*$ is *polylogtime reducible* to $K \subseteq B^*$, for short $L \leq_{\mathrm{plt}} K$ ([3, 19]), iff there exist functions $f : A^* \times \mathbb{N} \to B$ and $g : A^* \to \mathbb{N}$, computable in polylogarithmic time on a deterministic Turing machine with random access to the input bits, such that $x \in L \leftrightarrow f(x,1)f(x,2)\dots f(x,g(x)) \in K$ for every $x \in A^*$. By a *plt-function* we mean any function of the form $x \mapsto f(x,1)f(x,2)\dots f(x,g(x))$ where f and g are computable in polylogarithmic time.

To explain relationship of this reducibility to complexity theory, let us recall some relevant definitions. Consider a polynomial-time nondeterministic Turing machine M working on an input word x over some alphabet B and printing a letter from another alphabet A after finishing any computation path. These values are the leaves of the binary tree defined by the nondeterministic choices of M on input x. An ordering of the tuples in the program of M determines a left-to-right ordering of all the leaves. In this way, M may be considered as a deterministic transducer computing a total function $M : B^* \to A^+$. Now, relate to any language $L \subseteq A^+$ (called in this situation a leaf language) the language $M^{-1}(L) \subseteq B^*$. Denote by $\mathrm{Leaf}_{\mathrm{b}}(L)$ the set of languages $M^{-1}(L)$, for all machines M specified above which have balanced computation trees, and denote by $\mathrm{Leaf}_{\mathrm{u}}(L)$ the set of languages $M^{-1}(L)$, for all machines M specified above (which may have unbalanced computation trees).

Obviously, we have $\mathrm{Leaf}_{\mathrm{b}}(L) \subseteq \mathrm{Leaf}_{\mathrm{u}}(L)$ for every language L, and there exist languages L where $\mathrm{Leaf}_{\mathrm{b}}(L) = \mathrm{Leaf}_{\mathrm{u}}(L)$ is unlikely. It turns out that many inportant complexity classes have natural and useful descriptions in terms of leaf

languages (see [21] and references therein). The following theorem was proved by D.P. Bovet, P. Crescenzi and R. Silvestri [3] and independently by N.K. Vereshchagin [19].

Theorem 1. *For all languages L and K, $L \leq_{\rm plt} K$ iff $\mathrm{Leaf_b}(L)^{\mathcal{O}} \subseteq \mathrm{Leaf_b}(K)^{\mathcal{O}}$ for every oracle $\mathcal{O}$.*

In [21] a notion of reducibility (called *ptt*-reducibility) was introduced which is related to the unbalanced leaf language definability exactly in the same way as *plt*-reducibility is related to the balanced leaf language definability in Theorem 1. For these reasons (and because regular languages are most natural to use as leaf languages) investigation of *plt*- and *ptt*-reducibilities (especially on regular languages) seems important. For some results in this directions see [17, 21, 5]. We start with the following obvious fact in which we, for simplicity of notation, identify preorders with the corresponding quotient partial orders (i.e., degree structures).

Proposition 2. *1. $\{\emptyset\}$ and $\{A^+\}$ are two distinct minimal elements of the degree structures $(P(A^+); \leq_{\rm plt})$ and $(\mathcal{R}; \leq_{\rm plt})$ which are below any other element.*

2. The structures $(P(A^+); \leq_{\rm plt})$ and $(\mathcal{R}; \leq_{\rm plt})$ are upper semilattices under the supremum operation $L \oplus K = aL \cup (A \setminus a)K$, where a is a fixed letter from A.

3. The structure $(P'(A^+); \leq_{\rm plt})$ is a distributive upper semilattice.

We do not know whether the structure $(\mathcal{R}'; \leq_{\rm plt})$ is a distributive upper semilattice.

Are there greatest elements in $(\mathcal{R}; \leq_{\rm plt})$ and $(\mathcal{R}; \leq_{ptt})$ (as usual, the corresponding languages are called complete)? From [7] we get the positive answer and even a clear descripion of complete languages. Namely, for a regular language L the following statements are equivalent: L is *plt*-complete; L is *ptt*-complete; the syntactic monoid $M(L)$ is not solvable (i.e., contains a non-solvable subgroup).

The following characterization of the relation $L \leq_{\rm plt} 0^*1(0 \cup 1)^*$ is a 'balanced' version of the corresponding result from [10, 2] for the 'unbalanced' model. We need to define some patterns (i.e. subgraphs in automata).

Definition 3. *Let $\mathcal{A}$ be a (deterministic) finite automaton (over A), q its initial state, and let $s.w$ denote the state reached by $\mathcal{A}$ when reading the word w starting from the state s.*

1. *A* balanced coNP*-pattern for $\mathcal{A}$ is formed by states s_0, s_1 and words x, y, u, v such that: $q.x = s_0$, $s_i.u = s_i$ for both $i < 2$, $s_0.v = s_1 = s_1.v$, $s_0.y$ is accepting, $s_1.y$ is rejecting, and $|u| = |v|$.*
2. *A* balanced co1NP*-pattern for $\mathcal{A}$ is formed by states $s_0, s_1, \ldots, s_n$ $(n > 1)$ and words x, y, u, v such that: $q.x = s_0$, $s_i.u = s_i$ for all $i \leq n$, $s_i.v = s_{i+1}$ for all $i < n$, $s_n.v = s_n$, the states $s_0.y, s_2.y, \ldots, s_n.y$ are accepting while the state $s_1.y$ is rejecting, and $|u| = |v|$.*

3. A balanced n-counting *pattern* ($n > 1$) *for* $\mathcal{A}$ *is formed by states* $s_0, \ldots, s_{n-1}$ *and words* x, y, u, v *such that:* $q.x = s_0$, $s_i.u = s_i$ *for all* $i < n$, $s_i.v = s_{i+1}$ *for all* $i < n-1$, $s_{n-1}.v = s_0$, $s_0.y$ *is accepting,* $s_1.y$ *is rejecting, and* $|u| = |v|$.

Theorem 4 ([5]). *Let* L *be a regular language and* $\mathcal{A}$ *the minimal automaton recognizing* L. *Then* $L \leq_{\rm plt} 0^*1(0 \cup 1)^*$ *iff* $\mathcal{A}$ *does not have balanced coNP, co1NP and* n*-counting patterns* ($n > 1$).

From results in [5] it also follows that a non-trivial regular language L defines the smallest degree in $(\mathcal{R}'; \leq_{\rm plt})$ iff $L \leq_{\rm plt} 0^*1(0 \cup 1)^*$ and $L \leq_{\rm plt} \overline{0^*1(0 \cup 1)^*}$.

Next we prove a result on initial segment of the structure $(\mathcal{R}'; \leq_{\rm plt})$. This result is implicit in [5] (provided we use Theorem 1 and some known facts on oracle separations). Nevertheless, we present a direct proof because its ideas are also used in some proofs below. Let $(P; \leq)$ be un upper semilattice with a least element 0. Recall that an *atom* of P is a minimal non-zero element of P. The semilattice is called *atomic* if below every non-zero element there is an atom.

Theorem 5. *The semilattice* $(\mathcal{R}'; \leq_{\rm plt})$ *is atomic with infinitely many atoms.*

Proof. Let $E \subseteq A^+$ be the language of words having at least one letter distinct from a fixed letter $a \in A$. Obviously, $E \equiv_{\rm plt} 0^*1(0 \cup 1)^*$. For any prime p, let $M_p \subseteq A^+$ consist of all words such that the number of occurences of letters distinct from a is divided by p. We claim that the languages $E, \overline{E}, M_p$ (p prime) define exactly the atoms of $(\mathcal{R}'; \leq_{\rm plt})$, i.e.:

1) the languages $E, \overline{E}, M_p$ are pairwise *plt*-incomparable,

2) for any $L \in \mathcal{R}'$ of non-smallest *plt*-degree, at least one of $E, \overline{E}, M_p$ is *plt*-reducible to L,

The assertion 1) follows form Theorem 1 and well-known oracle separations (alternatively, it maybe easily observed from definition of *plt*-reducibility).

2) Let $L \in \mathcal{R}'$ be of non-smallest degree. Then $L \not\leq_{\rm plt} E$ or $L \not\leq_{\rm plt} \overline{E}$. We consider only the first case, the second being dual. By Theorem 4, $\mathcal{A}$ contains a balanced coNP-pattern, or co1NP-pattern, or n-counting pattern. In the case of coNP-pattern, consider the *plt*-function $f(w) = xh(w)y$, where $h : A^+ \to A^+$ is the homomorphism satisfying $h(a) = u$ and $h(b) = v$ for any $b \in A \setminus \{a\}$. The function f satisfies $\overline{E} = f^{-1}(L)$, hence $\overline{E} \leq_{\rm plt} L$. In case of co1NP-pattern, we similarly get $0^*10^* = f^{-1}(L)$. Since $E \leq_{\rm plt} 0^*10^*$, we get $E \leq_{\rm plt} L$.

The case of the balanced n-counting pattern is a bit more tedious. Let $K \subseteq \{0,1\}^+$ be the language of all binary words w with $\#_1(w) \equiv i \pmod n$ to some $i < n$ such that $s_i.y$ is an accepting state of $\mathcal{A}$ (so $0 \in K$ and $1 \notin K$). Here $\#_1(w)$ denotes the number of occurances of 1 in w. As above, the *plt*-function $f : \{0,1\}^+ \to A^+$ with properties $f(w) = xh(w)y$, $h(0) = u$ and $h(1) = v$, reduces K to L. By the proof of Lemma 6 from [1], we may assume w.l.o.g. that n is prime. Hence, it suffices to reduce M_n to K. Let $M'_n \subseteq \{0,1\}^+$ be the set of words with $\#_1(w) \equiv 0 \pmod n$. Obviously, $M_n \equiv_{\rm plt} M'_n$, so it sufices to reduce M'_n to K.

Define a function g on $\{0,1\}^+$ as follows. Let $|g(w)| = |w|^{n-1}$ and for every $i \in \{1, \ldots, |w|^{n-1}\}$ the i-th letter in $g(w)$ is 1 iff $w(i_1) = \cdots = w(i_{n-1}) = 1$, where

$(i_1, \ldots, i_{n-1})$ is the i-th tuple in the lexicographic ordering of $\{1, \ldots, |w|\}^{n-1}$. Here $w(i)$ denotes the i-th letter of w. One easily checks that g is a *plt*-function and $\#_1(g(w)) = (\#_1(w))^{n-1}$. By Fermat's theorem, $\#_1(g(w)) \equiv 0 \pmod n$ if $\#_1(w) \equiv 0 \pmod n$ and $\#_1(g(w)) \equiv 1 \pmod n$ otherwise. Hence, g reduces M'_n to K. □

plt and *ptt*-reducibilities are not well enough related to natural hierarchies of regular sets [5, 21]. In Section 4 we consider reducibilities which behave better in this respect.

3 Regular Languages and Logic

Relate to any alphabet $A = \{a, \ldots\}$ the signature $\sigma = \sigma_A = \{\leq, Q_a, \ldots, \bot, \top, p, s\}$, where $\leq$ is a binary relation symbol, Q_a (for any $a \in A$) is a unary relation symbol, $\bot$ and $\top$ are constant symbols, and p, s are unary function symbols. A word $u = u_0 \ldots u_n \in A^+$ may be considered as a structure $\mathbf{u} = (\{0, \ldots, n\}; <, Q_a, \ldots)$ of signature σ, where $<$ has its usual meaning, $Q_a (a \in A)$ are unary predicates on $\{0, \ldots, n\}$ defined by $Q_a(i) \leftrightarrow u_i = a$, the symbols $\bot$ and $\top$ denote the least and the greatest elements, while p and s are respectively the predecessor and successor functions on $\{0, \ldots, n\}$ (with $p(0) = 0$ and $s(n) = n$). For a sentence ϕ of σ, let $L_\phi = \{u \in A^+ | \mathbf{u} \models \phi\}$. In [9] it was shown that the class of all languages of the form L_ϕ, where ϕ ranges through first-order sentences of σ, coincides with the class of star-free languages (known also as aperiodic languages).

We will consider also some enrichments of the signature σ. Namely, for any positive integer d let τ_d be the signature $\sigma \cup \{P_d^0, \ldots, P_d^{d-1}\}$, where P_d^r is the unary predicate true on the positions of a word which are equivalent to r modulo d (we again think that positions are numbered by non-negative integers $\{0, 1, \ldots\}$). Note that signature τ_1 is essentially the same as σ because P_1^0 is the valid predicate. In contrast, for $d > 1$ languages L_ϕ for sentences ϕ of τ_d need not be star-free. E.g., the sentence $P_2^1(\top)$ defines the language consisting of all words of even length.

For $n > 0$, let Σ_n be the class of all languages L_ϕ, where ϕ ranges through the Σ_n-sentences of σ. Let $\Pi_n = \mathrm{co}(\Sigma_n)$ denote the class of complements of Σ_n-languages, and $\Delta_n = \Sigma_n \cap \Pi_n$. In the same way we define classes of languages $\Sigma_n^{\tau_d}$ and so on related to signature τ_d. In [18] it was shown that the sequence $\{\Sigma_n\}_{n>0}$ essentially coincides with dot-depth hierarchy which is a popular object of automata theory.

To our knowledge, the hierarchy $\Sigma_n^{\tau_d}$ for $d > 1$ was not so far considered in the literature. It turns out that methods developed in [18, 14, 15] for the dot-depth and Straubing-Therien hierarchies (based on Ehrenfeucht-Fraisse games) generalize in a straightforward way to hierarchies $\Sigma_n^{\tau_d}$ and to the diference hierarchy over any $\Sigma_n^{\tau_d}$. E.g., any of these hierarchies does not collapse. Most of results from [11, 6] related to the difference hierarchy over Σ_1 also generalize to similar hierarchies over $\Sigma_1^{\tau_d}$. We plan to give more details on this in a forthcoming paper.

We are also interested in the signature $\tau = \bigcup_d \tau_d$. By Σ_n^τ we denote classes of the hierarchy of languages induced by the quantifier-alternation hierarchy of τ-sentencies. From results presented in [16] it follows that the hierarchy Σ_n^τ exhausts the class of so called quasiaperiodic languges. It is easy to show that $\Sigma_n^\tau = \bigcup_d \Sigma_n^{\tau_d}$ for every $n > 0$.

For any set $\mathcal{P}$ of positive integers, let $FO + MOD(\mathcal{P})$ denote the class of languages defined by σ-sentences using counting quantifiers $\exists^{q,r}$ with moduli in $\mathcal{P}$, along with the usual first order quantifiers. It is known (see e.g. [16]) that the class $FO + MOD = FO + MOD(\{1, 2, \ldots\})$ consists exactly of languages with solvable syntactic monoid. More information on the logical approach maybe found in [16, 18, 11].

4 Quantifier-Free Reducibilities

In [17] the reducibility by quantifier-free formulas of signature σ was introduced and studied. Here we generalize notions and results of [17] to signature τ_d for every fixed $d > 0$, and present some new results. A *$qf\tau_d$-interpretation I over alphabets $A = \{a, \ldots\}$ and $B = \{b, \ldots\}$* is given by a tuple

$$(\phi_U(\bar{x}), \phi_<(\bar{x}, \bar{y}), \phi_\perp(\bar{x}), \phi_\top(\bar{x}), \phi_S(\bar{x}, \bar{y}), \phi_b(\bar{x}), \ldots, \phi_d^r(\bar{x}))$$

where $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{y} = (y_1, \ldots, y_n)$ are sequences of different variables of the same length $n > 0$ (n is fixed in advance) and $\phi_U(\bar{x}), \ldots, \phi_d^r(\bar{x})$ are quantifier-free formulas of τ_d^A with the following properties. Let $u = u_0 \cdots u_l$ be any word over A of length $|u| = l + 1$. Then the set $T = \{\bar{x} \in \{0, \ldots, l\}^n | \mathbf{u} \models \phi_U(\bar{x})\}$ should be non-empty and $\phi_<(\bar{x}, \bar{y}), ..., \phi_\perp(\bar{x}), \phi_\top(\bar{x}), \phi_S(\bar{x}, \bar{y})), \phi_b(\bar{x}), \phi_d^r(\bar{x})$ interpreted in $\mathbf{u}$ should define a model the universe T isomorphic to a word over B (the formulas $\phi_\perp(\bar{x}), \phi_\top(\bar{x})$ should be true exactly on the first and the last element, respectively). Any $qf\tau_d$-interpretation I induces a function $u \mapsto u^I$ from A^+ into B^+.

Examples. 1. Let $u \mapsto pu$ be the function on A^+ which adds a fixed prefix $p \in A^*$ to a word u. Is there a $qf\tau_d$-interpretation I over A and A such that $u^I = pu$ for any u? For $p = \varepsilon$ the answer is of course positive, otherwise it is negative (since any $qf\tau_d$-interpretation sends words of length 1 to words of length 1). But it is easy to see that there is a $qf\tau_d$-interpretation I such that $u^I = pu$ for any u of length > 1. The same of course applies to the operation of adding a suffix to a word.

2. Let $h : A^+ \to B^+$ be a semigroup morphism. Such functions are defined by their values $h : A \to B^+$ on the letters of A (i.e., words of length 1) because we have $h(a_0 \cdots a_l) = h(a_0) \cdots h(a_l)$, where $a_i \in A$. It is easy to see that for any such h with the property $\forall a, b \in A(|h(a)| \equiv |h(b)| \pmod d)$ there is a $qf\tau_d$-interpretation I over A and B such that $u^I = h(u)$ for almost all $u \in A^+$ (i.e. for all but finitely many of words).

Definition 6. *1. A function $f : A^+ \to B^+$ is called a* $qf\tau_d$-function *if there is a $qf\tau_d$-interpretation I over A and B such that $u^I = f(u)$ for almost all $u \in A^+$.*

2. *We say that $L \subseteq A^+$ is $qf\tau_d$-*reducible *to $K \subseteq B^+$ (in symbols $L \leq_{\mathrm{qf}\tau_\mathrm{d}} K$) if $L = f^{-1}(K)$ for some $qf\tau_d$-function $f : A^+ \to B^+$.*

The next theorem generalizes corresponding facts from [17] obtained there for the case of $qf\sigma$-reducibility.

Theorem 7. *1. The relation $\leq_{\mathrm{qf}\tau_\mathrm{d}}$ is reflexive and transitive.*
2. *$\{\emptyset\}$ and $\{A^+\}$ are two distinct minimal elements of the degree structure $(P(A^+); \leq_{\mathrm{qf}\tau_\mathrm{d}})$ which are below any other element.*
3. *The structure $(P(A^+); \leq_{\mathrm{qf}\tau_\mathrm{d}})$ is an upper semilattice.*
4. *The structure $(P'(A^+); \leq_{\mathrm{qf}\tau_\mathrm{d}})$ is a distributive upper semilattice with a least element which consists exactly of non-trivial $\Delta_1^{\tau_\mathrm{d}}$-languages.*
5. *The classes $\bigcup_n \Sigma_n^{\tau_\mathrm{d}}$, $\mathcal{R}$, $FO + MOD(\mathcal{P})$ ($\mathcal{P}$ is any set of positive integers containing d) are ideals of $(P(A^+); \leq_{\mathrm{qf}\tau_\mathrm{d}})$.*
6. *The classes $\Sigma_n^{\tau_\mathrm{d}}, \Pi_n^{\tau_\mathrm{d}}$, as well as all levels of the difference hierarchy over $\Sigma_n^{\tau_\mathrm{d}}$, are principal ideals of $(P(A^+); \leq_{\mathrm{qf}\tau_\mathrm{d}})$.*
7. *Let D_n be a $qf\tau_d$-complete set in the n-th level of the difference hierarchy over $\Sigma_1^{\tau_\mathrm{d}}$. Then $D_n \oplus \overline{D}_n$ is the infimum of sets $D_{n+1}, \overline{D}_{n+1}$ under $\leq_{\mathrm{qf}\tau_\mathrm{d}}$.*

In [17] some relationships between *plt*- and $qf\sigma$-reducibilities were established. These results also generalize to signatures τ_d.

Theorem 8. *1. If $L \leq_{\mathrm{qf}\tau_\mathrm{d}} M$ and M is of counting type then $L \leq_{\mathrm{plt}} M$.*
2. *If both L and M are of finite counting type then $L \leq_{\mathrm{qf}\tau_\mathrm{d}} M$ iff $L \leq_{\mathrm{plt}} M$.*
3. *Within the class of differences of $\Sigma_1^{\tau_\mathrm{d}}$ sets there are infinitely many languages modulo $\equiv_{\mathrm{qf}\tau_\mathrm{d}}$.*

Our next result states some relationships of the hierarchy Σ_n^τ to *plt*-reducibility.

Theorem 9. *1. There exists $K \in \Sigma_n^\sigma$ such that $L \leq_{\mathrm{plt}} K$ for every $L \in \Sigma_n^\tau$.*
2. *For any regular language L, $L \leq_{\mathrm{plt}} 0^*1(0 \cup 1)^*$ iff $L \in \Sigma_1^\tau$.*

Proof. (1) Let $K = H_n$ be the 'standard' set witnessing that $\Sigma_n^\sigma \not\subseteq \Pi_n^\sigma$ from the proof of Lemma 3.1 in [17]. Let $L \in \Sigma_n^\tau$, then $L \in \Sigma_n^{\tau_\mathrm{d}}$ for some $d > 0$. Let $f : A^+ \to A_n^+$ be the function constructed in the proof of that lemma (only this time ϕ is a quantifier-free formula of signature τ_d). One easily checks that f is a *plt*-function. Since $L = f^{-1}(K)$, $L \leq_{\mathrm{plt}} K$.

(2) Let $L \in \Sigma_1^\tau$. By the proof of (1), $L \leq_{\mathrm{plt}} K_1 = 0^*1(0 \cup 1)^*$. Conversely, let $L \leq_{\mathrm{plt}} 0^*1(0 \cup 1)^*$. By [5], L is a finite union of languages of the form $w_0(A^d)^*w_1 \cdots (A^d)^*w_n$, where $n \geq 0, d > 0$ and $w_i \in A^+$. One easily writes down an existential sentence ϕ of signature τ_d such that $w_0(A^d)^*w_1 \cdots (A^d)^*w_n = L_\phi$. Therefore, $L \in \Sigma_1^\tau$. □

We do not know whether the converse of (1) holds, i.e. whether Σ_n^τ is a principal ideal of $(\mathcal{R}; \leq_{\mathrm{plt}})$ for any $n > 1$ (for $n = 1$ this follows from (2)).

Next we consider an application to leaf language classes. First we characterize quasiaperiodic languages (which coincide with languages in $\bigcup_n \Sigma_n^\tau$, see [16]) in terms of forbidden patterns. The proof of the next result is obtained rather easily from the proof of Theorem VI.4.1 in [16] (for more details see [13]).

Theorem 10. *1. A regular language L is qausiaperiodic iff every finite automaton recognizing L has no balanced n-counting pattern ($n > 1$).*
2. For any regular language L and any $d > 0$, $L \in \cup_n \Sigma_n^{\tau_{\rm d}}$ iff every finite automaton recognizing L has no d-balanced n-counting pattern ($n > 1$).

Now we are able to characterize (uniformly on oracles) regular languages L with the property $\mathrm{Leaf_b}(L) \subseteq PH$, where PH is the class of languages in the polynomial-time hierarchy. Similar characterization for the unbalanced leaf language definability is well known.

Theorem 11. *A regular language L is quasiaperiodic iff $\mathrm{Leaf_b}(L)^O \subseteq PH^O$ for all oracles O.*

Proof. Let L be quasiaperiodic, then $L \in \Sigma_n^\tau$ for some $n > 0$. By Theorem 9(1), $L \leq_{\rm plt} K$ for some $K \in \Sigma_n^\sigma$. By Theorem 1, $\mathrm{Leaf_b}(L)^O \subseteq \mathrm{Leaf_b}(K)^O$ for all O. It is known [21] that $\mathrm{Leaf_b}(K)^O \subseteq PH^O$ for all O, hence $\mathrm{Leaf_b}(L)^O \subseteq PH^O$ for all O.

Conversely, let L be non-quasiaperiodic. By Theorem 10, there is an automaton which recognizes L and has a balanced n-counting pattern ($n > 1$). By the proof of Theorem 5, $M_p \leq_{\rm plt} L$ for some prime p. By Theorem 1, $\mathrm{Leaf_b}(M_p)^O \subseteq \mathrm{Leaf_b}(L)^O$ for all O. It is known [21] that $\mathrm{Leaf_b}(M_p)^O \not\subseteq PH^O$ for some O. Therefore, $\mathrm{Leaf_b}(L)^O \not\subseteq PH^O$ for some O. □

Corollary 12. *The class of quasiaperiodic languages is an ideal of $(\mathcal{R}; \leq_{\rm plt})$.*

We conclude this section with an analog of Theorem 5. For this we need the following parametrized version of Theorem 4. The notions of d-balanced coNP-, co1NP- and n-counting patterns are obtained from Definition 3 by replacing the equality $|u| = |v|$ on the equivalence $|u| \equiv |v| \equiv 0 \pmod d$.

Theorem 13. *Let L be a regular language and $\mathcal{A}$ the minimal automaton recognizing L. Then $L \leq_{\mathrm{qf}\tau_{\rm d}} 0^*1(0 \cup 1)^*$ iff $\mathcal{A}$ does not have d-balanced coNP, co1NP and n-counting patterns ($n > 1$).*

Note that for $d = 1$ we get the corresponding result from [2] (and that other forbidden-pattern results from [2, 5] are also parametrizable in the same way). The proof of the last theorem is obtained by easy modifications of the proofs of Claim 1 and Corollary 2 in [5].

With Theorem 13 at hand, the following result is proved in the same way as Theorem 5.

Theorem 14. *The semilattice $(\mathcal{R}'; \leq_{\mathrm{qf}\tau_{\rm d}})$ is atomic with infinitely many atoms.*

From distributivity of the semilattice $(\mathcal{R}'; \leq_{\mathrm{qf}\tau_{\rm d}})$ we immediately get the following.

Corollary 15. *The lattice $(Fin; \subseteq)$ of all finite subsets of ω is isomorphic to an ideal of $(\mathcal{R}'; \leq_{\mathrm{qf}\tau_{\rm d}})$.*

We note that the result from [17] on the pincipal ideal generated by the set $U \oplus \overline{U}$, where $U = 0^*1^*$, is also true for the structure $(P(A^+); \leq_{\mathrm{qf}\tau_{\rm d}})$. In particular, this ideal consists exactly of 11 degrees.

5 First Order Reducibilities

Here we consider weaker logical reducibilities, namely reducibilities $\leq_{\mathrm{fo}\tau_{\mathrm{d}}}$ by first order formulas of signature τ_d. Definition of $\leq_{\mathrm{fo}\tau_{\mathrm{d}}}$ is the same as that of $\leq_{\mathrm{qf}\tau_{\mathrm{d}}}$, only now the interpretation I consists of arbitrary first order formulas of τ_d. The following assertion is straightforward.

Theorem 16. *1. The relation $\leq_{\mathrm{fo}\tau_{\mathrm{d}}}$ is reflexive and transitive.*
2. $\{\emptyset\}$ and $\{A^+\}$ are two distinct minimal elements of $(P(A^+); \leq_{\mathrm{fo}\tau_{\mathrm{d}}})$ which are below any other element.
3. The structure $(P(A^+); \leq_{\mathrm{fo}\tau_{\mathrm{d}}})$ is an upper semilattice.
4. The classes $\mathcal{R}$, $FO + MOD(\mathcal{P})$ ($\mathcal{P}$ is any set of positive integers containing d) are ideals of $(P(A^+); \leq_{\mathrm{fo}\tau_{\mathrm{d}}})$.
5. The structures $(P'(A^+); \leq_{\mathrm{fo}\tau_{\mathrm{d}}})$ and $(\mathcal{R}'; \leq_{\mathrm{fo}\tau_{\mathrm{d}}})$ are distributive upper semilattices with a least element consisting exactly of non-trivial languages from $\bigcup_n \Sigma_n^{\tau_{\mathrm{d}}}$.

The next theorem is an analog of Theorem 14 and is proved in the same way (using Theorem 10(2)).

Theorem 17. *The semilattice $(\mathcal{R}'; \leq_{\mathrm{fo}\tau_{\mathrm{d}}})$ is atomic with infinitely many atoms which are defined exactly by the sets M_p (p prime).*

The reducibility $\leq_{\mathrm{fo}\sigma}$ seems to be related to a well-known open problem of automata theory, namely the problem of generalized star-height. We believe that better understanding of the structure $(\mathcal{R}; \leq_{\mathrm{fo}\sigma})$ may shed some light on this problem.

6 Other Reducibilities and Open Questions

There are also other natural reducibilities on regular sets. E.g., let $\leq_{\mathrm{fom}}$ be defined in the same way as $\leq_{\mathrm{fo}\tau_{\mathrm{d}}}$ but this time the interpretation I consists of arbitrary $(FO + MOD)$-formulas. One can easily establish for this reducibility an analog of Theorem 16 and that there are two distinct (modulo $\equiv_{\mathrm{fom}}$) non-trivial regular languages. We do not know whether there exist three non-trivial regular languages which are pairwise distinct modulo $\equiv_{\mathrm{fom}}$.

We do not currently know whether there exist regular languages which are complete under "logical" reducibilities considered above. Analogs of open questions from [17] for $qf\tau_d$-reducibilities seem also interesting, as well as questions on relationships between $qf\tau_d$-reducibilities for different d (it may be shown that these reducibilities are pairwise incomparable).

One could consider also reductions by functions computable by natural classes of finite transducers. Such reducibilities were successfully applied for classification of regulat ω-languages (see [20]).

References

1. B. Borchert. On the acceptance power of regular languages. *Theoret. Comp. Sci*, 148 (1995), 207–225.
2. B. Borchert, D. Kuske and F. Stephan. On existentially first–order definable languages and their relation to NP. *Theor. Informatics Appl.*, 33 (1999), 259–269.
3. D.P. Bovet, P. Crescenzi and R. Silvestri. A uniform approach to define complexity classes. *Theoret. Comp. Sci*, 104 (1992), 263–283.
4. J.R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logic Grundl. Math.*, 6 (1960), 66–92.
5. C. Glaßer. Polylogtime-reductions decrees dot-depth, Proceedings 22nd Symposium on Theoretical Aspects of Computer Science, *Lecture Notes in Computer Science*, v. 3404 (2005), Berlin, Springer.
6. C. Glaßer amd H. Schmitz. The Boolean Structure of Dot-Depth One. *J. of Automata, Languages and Combinatorics*, 6 (2001), 437–452.
7. U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer and K.W. Wagner. On the power of polynomial time bit-reductions, *Proc. 8th Structure in Complexity Theory*, 1993, 200–207.
8. N. Immermann. *Descriptive Complexity*. Berlin, Springer, 1999.
9. R. McNaughton and S. Papert. *Counter–free automata*. MIT Press, Cambridge, Massachussets, 1971.
10. J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30 (1997), 383–422.
11. V.L. Selivanov. A logical approach to decidability of hierarchies of regular star-free languages. *Lecture Notes in Computer Science*, v. 2010. Berlin, Springer, 2001, 539–550
12. V.L. Selivanov. Relating automata-theoretic hierarchies to complexity-theoretic hierarchies. *Theoret. Informatics Appl.*, 36 (2002), 29–42.
13. V.L. Selivanov. Some hierarchies and reducibilities on regular languages. University of Würzburg, Technical Report 349, 2004, 21 pp.
14. A.G. Shukin. Difference hierarchies of regular languages. *Comp. Systems*, Novosibirsk, 161 (1998), 141–155 (in Russian).
15. V.L. Selivanov and A.G. Shukin. On hierarchies of regular star-free languages (in Russian). Preprint 69 of A.P. Ershov Institute of Informatics Systems, 2000, 28 p.
16. H. Straubing. Finite automata, formal logic and circuit complexity. Birkhäuser, Boston, 1994.
17. V.L. Selivanov and K.W. Wagner. A reducibility for the dot-depth hierarchy. Proc. 29th Int. Symp. on Mathematical Foundations of Computer Science, *Lecture Notes in Computer Science*, v. 3153. Berlin, Springer, 2004, 783–793.
18. W. Thomas. Classifying regular events in symbolic logic. *J. Comp. and Syst. Sci.*, 25 (1982), 360–376.
19. N.K. Vereshchagin. Relativizable and non-relativizable theorems in the polynomial theory of algorithms. *Izvestiya Rossiiskoi Akademii Nauk*, 57 (1993), 51–90 (in Russian, there is English translation).
20. K.W. Wagner. On ω-regular sets. *Inform. and Control*, 43 (1979), p. 123—177.
21. K.W. Wagner. Leaf language classes. Proc. of MCU'2004-conference in Saint-Petersburg, Russia, September 21-24, 2004 (revised version to appear in LNCS 3354).

Computability and Discrete Dynamical Systems

Wilfried Sieg

Carnegie Mellon University,
Pittsburgh, PA 15213, U.S.A

Church's and Turing's theses dogmatically assert that an informal notion of computability is captured by a particular mathematical concept. I present an analysis of computability that leads to precise concepts, but dispenses with theses.

To investigate computability is to analyze processes that can in principle be carried out by calculators. Drawing on this lesson we owe to Turing and recasting work of Gandy, I formulate finiteness and locality conditions for two types of calculators, human computing agents and mechanical computing devices; the distinctive feature of the latter is that they can operate in parallel.

The analysis leads to axioms for discrete dynamical systems (representing human and machine computations) and allows the reduction of models of these axioms to Turing machines. Cellular automata and a variety of artificial neural nets can be shown to satisfy the axioms for machine computations.

References

1. Gandy, Robin: Church's Thesis and Principles for Mechanisms. In *The Kleene Symposium*, edited by J. Barwise, H.J. Keisler, and K. Kunen, North-Holland 1980, pp. 123–148.
2. Sieg, Wilfried: Calculations by Man and Machine: Conceptual Analysis. In *Reflections on the Foundations of Mathematics*, edited by W. Sieg, R. Sommer, and C. Talcott, Lecture Notes in Logic 15 (2002), pp. 390–409.
3. Sieg, Wilfried: Calculations by Man and Machine: Mathematical Presentation. In *In the Scope of Logic, Methodology and Philosophy of Science*, Vol. I (2002), edited by P. Gärdenfors, J. Wolenski, and K. Kijania-Placek, Kluwer, pp. 247–262.
4. Turing, Alan: On Computable Numbers with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 2, Vol. 45 (1936), pp. 161–228.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, p. 440, 2005.

Uniform Operators

Ivan N. Soskov

Faculty of Mathematics and Computer Science,
Sofia University, 5 James Bourchier Blvd.
1164 Sofia, Bulgaria
soskov@fmi.uni-sofia.bg

Abstract. We present the definition and a normal form of a class of operators on sets of natural numbers which generalize the enumeration operators.

1 Introduction

In his book [1–p.145] ROGERS gives the following intuitive explanation of the notion of enumeration reducibility:

> Let sets A and B be given. ... To put it as briefly as possible: A is *enumeration reducible* to B if there is an effective procedure for getting an enumeration of A from *any* enumeration of B.

On the next page ROGERS continues with the formal definition of the enumeration reducibility, where W_z denotes the c.e. set with Gödel number z and D_u denotes the finite set having canonical code u.

Definition 1. A is *enumeration reducible* to B (notation: $A \leq_{\mathrm{e}} B$) if

$$(\exists z)(\forall x)[x \in A \iff (\exists u)[\langle x, u\rangle \in W_z \ \&\ D_u \subseteq B]].$$

A is *enumeration reducible* to B *via* z if

$$(\forall x)[x \in A \iff (\exists u)[\langle x, u\rangle \in W_z \ \&\ D_u \subseteq B]].$$

Finally ROGERS defines for every z the enumeration operator $\Phi_z : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$.

Definition 2. $\Phi_z(X) = Y$ if $Y \leq_{\mathrm{e}} X$ via z.

Though the relationship of the intuitive definition with the formal one is well explained in [1] it is tempting to formalize the intuitive definition in a more direct way. Consider again the sets A and B. To get an enumeration of B we need an oracle X and if we have such an enumeration relative to X than B will be c.e. in X, so $B = W_b^X$ for some $b \in \mathbb{N}$, where W_b^X denotes the domain of the b-th Oracle Turing Machine using as oracle the characteristic function of X. From the intuitive remarks it follows that if $A \leq_e B$, and $B = W_b^X$, then there exists an a such that $A = W_a^X$ and we can obtain such an a from b in a way which does not depend on the oracle X. So it seems reasonable to consider the following definition of a class of operators which we call *uniform operators*.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 441–450, 2005.

Definition 3. A mapping $\Gamma : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ is called *uniform operator* if there exists a total function γ on the natural numbers such that for all $b \in \mathbb{N}$ and $X \subseteq \mathbb{N}$ we have that $\Gamma(W_b^X) = W_{\gamma(b)}^X$.

The following result shows that the intuitive remarks quoted at the beginning correspond exactly to the formal definition of the enumeration operators.

Theorem 4. *The uniform operators coincide with the enumeration operators.*

The theorem above can be considered as a uniform version of a result of Selman [2].

Theorem 5 (Selman).

$$A \leq_{\mathrm{e}} B \iff \forall X(B \text{ is c.e. in } X \Rightarrow A \text{ is c.e. in } X).$$

Selman's theorem is generalized by Case [3] and Ash [4]. Following the same fashion we come to the following definition.

Definition 6. Let $n, k \in \mathbb{N}$. A mapping $\Gamma : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ is *uniform operator of type* $(n \to k)$ if there exists a total function γ on the natural numbers such that for all $b \in \mathbb{N}$ and $X \subseteq \mathbb{N}$ we have that $\Gamma(W_b^{X^{(n)}}) = W_{\gamma(b)}^{X^{(k)}}$.

The characterization of the uniform operators of type $(n \to k)$ uses the notion of *enumeration jump* defined in Cooper [5] and further studied by McEvoy [6]. Here we shall use the following definition of the e-jump which is m-equivalent to the original one, see [6]:

Definition 7. Given a set A, let $K_A^0 = \{\langle x, z\rangle : x \in \Phi_z(A)\}$. Define the *e-jump* A'_{e} of A to be the set $K_A^0 \oplus (\mathbb{N} \setminus K_A^0)$.

For any set A by $A_{\mathrm{e}}^{(n)}$ we shall denote the n-th e-jump of A.

Theorem 8. *1. Let $k < n$. Then the uniform operators of type $(n \to k)$ coincide with the constant mappings $\lambda B.S$, where S is some Σ_{k+1}^0 set.*

2. Let $n \leq k$. Then the uniform operators of type $(n \to k)$ are exactly those mappings of $\Gamma : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ for which there exists an enumeration operator Φ such that for all $B \subseteq \mathbb{N}$, $\Gamma(B) = \Phi((B \oplus \emptyset^{(n)})_{\mathrm{e}}^{(k-n)})$.

Finally let us consider the general case.

Definition 9. Let $k_0 < \cdots < k_r$ and k be natural numbers. A mapping $\Gamma : \mathcal{P}(\mathbb{N})^{(r+1)} \to \mathcal{P}(\mathbb{N})$ is a *uniform operator of type* $(k_0, \ldots, k_r \to k)$ if there exists a function $\gamma : \mathbb{N}^{r+1} \to \mathbb{N}$ such that for all $b_0, \ldots, b_r \in \mathbb{N}$ and $X \subseteq \mathbb{N}$,

$$\Gamma(W_{b_0}^{X^{(k_0)}}, \ldots, W_{b_r}^{X^{(k_r)}}) = W_{\gamma(b_0,\ldots,b_r)}^{X^{(k)}}.$$

Let us fix the natural numbers $k_0, \ldots, k_r$. Denote by $\bar{k}$ the sequence $k_0, \ldots, k_r$. Given sets of natural numbers $B_0, \ldots, B_r$, we define the set $\mathcal{P}_{\bar{k}}^{(k)}(B_0, \ldots, B_r)$ by induction on k.

Definition 10. (i) Set

$$\mathcal{P}^{(0)}_{\bar{k}}(B_0,\dots,B_r) = \begin{cases} B_0, \text{ if } k_0 = 0, \\ \emptyset, \quad \text{otherwise.} \end{cases}$$

(ii) Let

$$\mathcal{P}^{(k+1)}_{\bar{k}}(B_0,\dots,B_r) = \begin{cases} (\mathcal{P}^{(k)}_{\bar{k}}(B_0,\dots,B_r))'_{\mathrm{e}}, & \text{if } k+1 \notin \{k_1,\dots,k_r\}, \\ (\mathcal{P}^{(k)}_{\bar{k}}(B_0,\dots,B_r))'_{\mathrm{e}} \oplus B_i, & \text{if } k+1 = k_i. \end{cases}$$

For example, for any two natural numbers n and k and any $B \subseteq \mathbb{N}$ we have that

$$\mathcal{P}^{(k)}_n(B) = \begin{cases} \emptyset^{(k)}, & \text{if } k < n, \\ (\emptyset^{(n)} \oplus B)^{(k-n)}_{\mathrm{e}}, & \text{if } n \le k. \end{cases}$$

The theorem below is our main result.

Theorem 11. *1. The uniform operators of type $(k_0,\dots,k_r \to k)$ are exactly those mappings $\Gamma : \mathcal{P}(\mathbb{N})^{r+1} \to \mathcal{P}(\mathbb{N})$ for which there exists an enumeration operator Φ such that for all subsets $B_0,\dots,B_r$ of $\mathbb{N}$,*

$$\Gamma(B_0,\dots,B_r) = \Phi(\mathcal{P}^{(k)}_{k_0,\dots,k_r}(B_0,\dots,B_r)).$$

2. For every uniform operator Γ of type $(k_0,\dots,k_r \to k)$ there exists a total computable function $\gamma(b_0,\dots,b_r)$ such that for all $b_0\dots,b_r \in \mathbb{N}$ and $X \subseteq \mathbb{N}$,

$$\Gamma(W^{X^{(k_0)}}_{b_0},\dots,W^{X^{(k_r)}}_{b_r}) = W^{X^{(k)}}_{\gamma(b_0,\dots,b_r)}.$$

In the rest of the paper we present a proof of Theorem 11.

2 Regular Enumerations

The proof of Theorem 11 uses the technique of the regular enumerations, presented in [7] and [8].

Let us consider a sequence $\{B_i\}$ of sets of natural numbers.

Roughly speaking a k-regular enumeration f is a kind of generic function such that for all $i \le k$, B_i is computably enumerable in $f^{(i)}$ uniformly in i.

Let f be a total mapping on $\mathbb{N}$. We define for every i, e, x the relation $f \models_i F_e(x)$ by induction on i:

Definition 12. (i) $f \models_0 F_e(x) \iff \exists v(\langle x,v\rangle \in W_e \ \&\ (\forall u \in D_v)(f((u)_0) = (u)_1))$;

(ii)

$$f \models_{i+1} F_e(x) \iff \exists v(\langle x,v\rangle \in W_e \ \&\ (\forall u \in D_v)((u = \langle e_u, x_u, 0\rangle \ \&\ f \models_i F_{e_u}(x_u)) \vee (u = \langle e_u, x_u, 1\rangle \ \&\ f \not\models_i F_{e_u}(x_u)))).$$

Set $f \models_i \neg F_e(x) \iff f \not\models_i F_e(x)$.

The following lemma can be easily proved by induction on i:

Lemma 13. *For every i there exists a total computable function $h_i(a)$ such that for all a,*

$$W_a^{f^{(i)}} = \{x : f \models_i F_{h_i(a)}(x)\}.$$

In what follows we shall use the term *finite part* for finite mappings of $\mathbb{N}$ into $\mathbb{N}$ defined on finite segments $[0, q-1]$ of $\mathbb{N}$. Finite parts will be denoted by the letters τ, δ, ρ. If $\mathrm{dom}(\tau) = [0, q-1]$, then let $\mathrm{lh}(\tau) = q$.

We shall suppose that an effective coding of all finite sequences and hence of all finite parts is fixed. Given two finite parts τ and ρ we shall say that τ is less than or equal to ρ if the code of τ is less than or equal to the code of ρ. By $\tau \subseteq \rho$ we shall denote that the partial mapping ρ extends τ and say that ρ is an extension of τ. For any τ, by $\tau \restriction n$ we shall denote the restriction of τ on $[0, n-1]$.

Set for every i, $\overline{B}_i = \mathbb{N} \oplus B_i$.

Below we define for every i the i-regular finite parts.

The 0-*regular finite parts* are finite parts τ such that $\mathrm{dom}(\tau) = [0, 2q+1]$ and for all odd $z \in \mathrm{dom}(\tau), \tau(z) \in \overline{B}_0$.

If $\mathrm{dom}(\tau) = [0, 2q+1]$, then the 0-rank $|\tau|_0$ of τ is equal to the number $q+1$ of the odd elements of $\mathrm{dom}(\tau)$. Notice that if τ and ρ are 0-regular, $\tau \subseteq \rho$ and $|\tau|_0 = |\rho|_0$, then $\tau = \rho$.

For every 0-regular finite part τ, let B_0^τ be the set of the odd elements of $\mathrm{dom}(\tau)$.

Given a 0-regular finite part τ, let

$$\tau \Vdash_0 F_e(x) \iff \exists v(\langle x, v\rangle \in W_e \ \&\ (\forall u \in D_v)(\tau((u)_0) \simeq (u)_1))$$

$$\tau \Vdash_0 \neg F_e(x) \iff \forall(0\text{-regular } \rho)(\tau \subseteq \rho \Rightarrow \rho \nVdash_0 F_e(x)).$$

Proceeding by induction, suppose that for some i we have defined the i-regular finite parts and for every i-regular τ – the i-rank $|\tau|_i$ of τ, the set B_i^τ and the relations $\tau \Vdash_i F_e(x)$ and $\tau \Vdash_i \neg F_e(x)$. Suppose also that if τ and ρ are i-regular, $\tau \subseteq \rho$ and $|\tau|_i = |\rho|_i$, then $\tau = \rho$.

Set $X_j^i = \{\rho : \rho \text{ is } i\text{-regular } \&\ \rho \Vdash_i F_{(j)_0}((j)_1)\}$.

Given a finite part τ and a set X of i-regular finite parts, let $\mu_i(\tau, X)$ be the least extension of τ belonging to X if any, and $\mu_i(\tau, X)$ be the least i-regular extension of τ otherwise. We shall assume that $\mu_i(\tau, X)$ is undefined if there is no i-regular extension of τ.

A *normal i-regular extension* of an i-regular finite part τ is any i-regular finite part $\rho \supseteq \tau$ such that $|\rho|_i = |\tau|_i + 1$.

Let τ be a finite part defined on $[0, q-1]$ and $r \geq 0$. Then τ is $(i+1)$-*regular* with $(i+1)$-*rank* $r+1$ if there exist natural numbers

$$0 < n_0 < l_0 < m_0 < b_0 < n_1 < l_1 < m_1 < b_1 \cdots < n_r < l_r < m_r < b_r < n_{r+1} = q$$

such that $\tau \restriction n_0$ is an i-regular finite part with i-rank equal to 1 and for all j, $0 \leq j \leq r$, the following conditions are satisfied:

a) $\tau \restriction l_j$ is a normal i-regular extension of $\tau \restriction n_j$;
b)

$$\tau \restriction m_j = \begin{cases} \mu_i(\tau \restriction (l_j+1), X^i_{\langle p,l_j\rangle}), & \text{if } \tau(n_j) \simeq \langle i+1, p\rangle + 1, \\ \text{a normal } i\text{-regular extension of } \tau \restriction l_j, & \text{otherwise;} \end{cases}$$

c)

$$\tau \restriction b_j = \begin{cases} \mu_i(\tau \restriction (m_j+1), X^i_{\langle p,q\rangle}), & \text{if } \tau(m_j) \simeq \langle p, q\rangle + 1, \\ \text{a normal } i\text{-regular extension of } \tau \restriction m_j, & \text{if } \tau(m_j) \simeq 0; \end{cases}$$

d) $\tau(b_j) \in \overline{B}_{i+1}$;
e) $\tau \restriction n_{j+1}$ is a normal i-regular extension of $\tau \restriction b_j$.

The following lemma shows that the $(i+1)$-rank is well defined.

Lemma 14. *Let τ be an $(i+1)$-regular finite part. Then*

1. *Let $n'_0, l'_0, m'_0, b'_0, \ldots, n'_p, l'_p, m'_p, b'_p, n'_{p+1}$ and $n_0, l_0, m_0, b_0, \ldots, n_r, l_r, m_r, b_r, n_{r+1}$ be two sequences of natural numbers satisfying a)–e). Then $r = p, n_{p+1} = n'_{p+1}$ and for all $j \le r, n_j = n'_j, l_j = l'_j, m_j = m'_j$ and $b_j = b'_j$.*
2. *If ρ is $(i+1)$-regular, $\tau \subseteq \rho$ and $|\tau|_{i+1} = |\rho|_{i+1}$, then $\tau = \rho$.*
3. *τ is i-regular and $|\tau|_i > |\tau|_{i+1}$.*

Let τ be $(i+1)$-regular and $n_0, l_0, m_0, b_0, \ldots, n_r, l_r, m_r, b_r, n_{r+1}$ be the sequence satisfying a)–e). Then let $B^\tau_{i+1} = \{b_0, \ldots, b_r\}$ and $M^\tau_{i+1} = \{m_0, \ldots, m_r\}$.

To conclude with the definition of the regular finite parts, let for every $(i+1)$-regular finite part τ

$$\tau \Vdash_{i+1} F_e(x) \iff \exists v(\langle x, v\rangle \in W_e \ \&\ (\forall u \in D_v)((u = \langle e_u, x_u, 0\rangle \ \&\ \tau \Vdash_i F_{e_u}(x_u)) \vee (u = \langle e_u, x_u, 1\rangle \ \&\ \tau \Vdash_i \neg F_{e_u}(x_u)))).$$

$$\tau \Vdash_{i+1} \neg F_e(x) \iff (\forall (i+1)\text{-regular } \rho)(\tau \subseteq \rho \Rightarrow \rho \nVdash_{i+1} F_e(x)).$$

Definition 15. Let f be a total mapping of $\mathbb{N}$ in $\mathbb{N}$. Then f is a *k-regular enumeration* (with respect to $\{B_i\}$) if the following conditions hold:

(i) For every finite part $\delta \subseteq f$, there exists a k-regular extension τ of δ such that $\tau \subseteq f$.
(ii) If $i \le k$ and $z \in \overline{B}_i$, then there exists an i-regular $\tau \subseteq f$ such that $z \in \tau(B^\tau_i)$.
(iv) If $i < k$, then for every pair $\langle p, q\rangle$ of natural numbers, there exists an $i+1$-regular finite part $\tau \subseteq f$ such that for some $m \in M^\tau_{i+1}$, $\tau(m) \simeq \langle p, q\rangle + 1$.

Clearly, if f is a k-regular enumeration and $i \le k$, then for every $\delta \subseteq f$, there exists an i-regular $\tau \subseteq f$ such that $\delta \subseteq \tau$. Moreover there exist i-regular finite parts of f of arbitrary large rank.

Given a regular f, let for $i \le k, B^f_i = \{b : (\exists \tau \subseteq f)(\tau \text{ is } i\text{-regular } \&\ b \in B^\tau_i)\}$. Clearly $f(B^f_i) = B_i$.

Now let us turn to the properties of the regular finite parts and of the regular enumerations.

3 Properties of the Regular Enumerations

First of all, notice that the clause (iv) of the definition of the regular enumerations ensures that a k-regular enumeration f is generic with respect to the family $\{X^i_j : i < k, j \in \mathbb{N}\}$. So we have the following Truth Lemma:

Lemma 16. *Let f be a k-regular enumeration. Then*

1. *For all $i \leq k$, $f \models_i F_e(x) \iff (\exists \tau \subseteq f)(\tau$ is i-regular $\& \ \tau \Vdash_i F_e(x))$.*
2. *For all $i < k$, $f \models_i \neg F_e(x) \iff (\exists \tau \subseteq f)(\tau$ is i-regular $\& \ \tau \Vdash_i \neg F_e(x))$.*

Let us define for every natural k the set P_k by induction on k:

Definition 17. (i) $P_0 = \overline{B}_0$;
(ii) $P_{k+1} = (P_k)'_e \oplus \overline{B}_{k+1}$.

Denote by $\mathcal{R}_i$ the set of all i-regular finite parts.
For $j \in \mathbb{N}$ let $\mu_i(\tau, j) \simeq \mu_i(\tau, X^i_j)$,

$$Y^i_j = \{\tau : (\exists \rho \supseteq \tau)(\rho \text{ is } i\text{-regular } \& \ \rho \Vdash_i F_{(j)_0}((j)_1))\}$$

$$Z^i_j = \{\tau : \tau \text{ is } i\text{-regular } \& \ \tau \Vdash_i \neg F_{(j)_0}((j)_1)\}.$$

Proposition 18. *For every $i \in \mathbb{N}$ the following assertions hold:*

1. *There exists an enumeration operators R_i such that for every sequence $\{B_i\}$ of sets of natural numbers, $\mathcal{R}_i = R_i(P_i)$.*
2. *There exist computable functions $x_i(j)$ and $y_i(j)$ such that for every j and every sequence $\{B_i\}$ of sets of natural numbers,*

$$X^i_j = \Phi_{x_i(j)}(P_i) \text{ and } Y^i_j = \Phi_{y_i(j)}(P_i).$$

3. *There exists a computable function $z_i(j)$ such that for every j and every sequence $\{B_i\}$ of sets of natural numbers,*

$$Z^i_j = \{z_i(j)\}^{P'_i}.$$

4. *There exists an Oracle Turing Machine m_i such that for every sequence $\{B_i\}$ of sets of natural numbers,*

$$\mu_i = \{m_i\}^{P'_i}.$$

The following proposition is important for the proof of Theorem 11.

Proposition 19. *For every $i \in \mathbb{N}$ there exists an Oracle Turing Machine b_i such that for every sequence $\{B_i\}$ of sets of natural numbers and every k-regular with respect to $\{B_i\}$ enumeration f,*

$$(\forall i \leq k)(B_i = W^{f^{(i)}}_{b_i}).$$

Proof. We shall define the machines b_i by induction on i. Clearly for every sequence $\{B_i\}$ of sets of natural numbers and every k-regular with respect to $\{B_i\}$ enumeration f, $B_0 = \{x : 2x+1 \in \overline{B}_0\}$, $\overline{B}_0 = f(B_0^f)$ and B_0^f is equal to the set of all odd numbers.

So we may define the machine b_0 as follows:

```
input X;
 Y:= 0;
 while (2X + 1 =\= f(2Y+1)) do
 Y := Y+1;
end.
```

Suppose that $i < k$ and the machines $b_0, \ldots, b_i$ are defined. Following the definition of P_i we can define an oracle machine p' which given a sequence $\{B_i\}$ and a k-regular f computes the characteristic function of P_i' using $f^{(i+1)}$ as an oracle. So it is sufficient to show that we can enumerate the set $\overline{B}_i$ by means of P_i' and f, uniformly in P_i' and f.

Since f is $(i+1)$-regular, for every finite part δ of f there exists an $(i+1)$-regular $\tau \subseteq f$ such that $\delta \subseteq \tau$. Hence there exist natural numbers

$$0 < n_0 < l_0 < m_0 < b_0 < n_1 < l_1 < m_1 < b_1 < \cdots < n_r < l_r < m_r < b_r < \ldots,$$

such that for every $r \geq 0$, the finite part $\tau_r = f \upharpoonright n_{r+1}$ is $(i+1)$-regular and $n_0, l_0, m_0, b_0, \ldots, n_r, l_r, m_r, b_r, n_{r+1}$ are the numbers satisfying the conditions a)–e) from the definition of the $(i+1)$-regular finite part τ_r. Clearly $B_{i+1}^f = \{b_0, b_1 \ldots\}$. We shall describe a procedure which lists $n_0, l_0, m_0, b_0, \ldots$ in an increasing order using the oracles P_i' and f.

Clearly $f \upharpoonright n_0$ is i-regular and $|f \upharpoonright n_0|_i = 1$. By Lemma 18 $\mathcal{R}_i$ is uniformly computable in P_i'. Using f we can generate consecutively the finite parts $f \upharpoonright q$ for $q = 1, 2 \ldots$. By Lemma 14 $f \upharpoonright n_0$ is the first element of this sequence which belongs to $\mathcal{R}_i$. Clearly $n_0 = \mathrm{lh}(f \upharpoonright n_0)$.

Suppose that $r \geq -1$ and $n_0, l_0, m_0, b_0, \ldots, n_r, l_r, m_r, b_r, n_{r+1}$ have already been listed. Since $f \upharpoonright l_{r+1}$ is a normal i-regular extension of $f \upharpoonright n_{r+1}$ it is the shortest finite part of f which extends $f \upharpoonright n_{r+1}$ and belongs to $\mathcal{R}_i$. So we can find the number l_{r+1}. Now, we have to consider two cases:

a) $f(n_{r+1}) = 0$ or $f(n_{r+1}) = \langle j, p\rangle + 1$, where $j \neq i+1$. Then again $f \upharpoonright m_{r+1}$ is the shortest finite part of f which belongs to $\mathcal{R}_i$ and extends $f \upharpoonright l_{r+1}$.

b) $f(n_{r+1}) = \langle i+1, p\rangle + 1$. Then $f \upharpoonright m_{r+1} = \mu_i(f \upharpoonright (l_{r+1}+1), X^i_{\langle p, l_{r+1}\rangle})$.

In both cases we can find $f \upharpoonright m_{r+1}$ effectively in f and P_i'. Clearly $m_{r+1} = \mathrm{lh}(f \upharpoonright m_{r+1})$. From m_{r+1} we reach b_{r+1} in a way similar to the previous one. Finally, from b_{r+1} we reach n_{r+2} using the fact that $f \upharpoonright n_{r+2}$ is a normal i-regular extension of $f \upharpoonright b_{r+1}$. Now we have a machine which decides the set B_{i+1}^f using the oracle $f^{(i+1)}$. From here, since $\overline{B} = f(B_{i+1}^f)$ we can easily obtain the machine b_{i+1}. □

4 Constructions of Regular Enumerations

Suppose that a sequence $\{B_i\}$ of sets of natural numbers is fixed.

Given a finite mapping τ defined on $[0, q-1]$, by $\tau * z$ we shall denote the extension ρ of τ defined on $[0, q]$ and such that $\rho(q) \simeq z$.

Lemma 20. *Let τ be an i-regular finite part defined on $[0, q-1]$. Let $x, y_1, \dots y_i \in \mathbb{N}$ and $z_0 \in \overline{B}_0, \dots, z_i \in \overline{B}_i$. There exists a normal i-regular extension ρ of τ such that:*

1. $\rho(q) \simeq x$;
2. $(\forall j < i)(y_{j+1} \in \rho(M^{\rho}_{j+1}))$.
3. $(\forall j \leq i)(z_j \in \rho(B^{\rho}_j))$.

Proof. Induction on i. The assertion is obvious for $i = 0$. Let τ be an $(i+1)$-regular finite part s.t. $\mathrm{dom}(\tau) = [0, q-1]$. Let $x, y_1, \dots, y_{i+1} \in \mathbb{N}, z_0 \in \overline{B}_0, \dots, z_{i+1} \in \overline{B}_{i+1}$ be given. Suppose that $|\tau|_{i+1} = r+1$ and $n_0, l_0, m_0, b_0, \dots, n_r, l_r, m_r, b_r, n_{r+1}$ are the natural numbers satisfying the conditions a)–e) from the definition of the $(i+1)$-regular finite parts. Notice that $n_{r+1} = q$. Since τ is also i-regular, by the induction hypothesis there exists a normal i-regular extension ρ_0 of $\tau * x$ such that $(\forall j < i)(y_{j+1} \in \rho(M^{\rho}_{j+1}))$ and $(\forall j \leq i)(z_j \in \rho(B^{\rho}_j))$. Let $l_{r+1} = \mathrm{lh}(\rho_0)$. Clearly there exists a normal i-regular extension δ of $\rho_0 * 0$ and hence the function $\mu_i(\rho_0 * 0, X^i_p)$ is defined for all $p \in \mathbb{N}$. Set

$$\rho_1 = \begin{cases} \delta, & \text{if } x = 0 \vee (\exists j)(x = \langle j, p\rangle + 1 \ \&\ j \neq i+1), \\ \mu_i(\rho_0 * 0, X^i_p), & \text{if } x = \langle i+1, p\rangle. \end{cases}$$

Set $m_{r+1} = \mathrm{lh}(\rho_1)$. Let ν be a normal extension of $\rho_1 * y_{i+1}$ and set

$$\rho_2 = \begin{cases} \nu, & \text{if } y_{i+1} = 0, \\ \mu_i(\rho_0 * 0, X^i_{y_{i+1}-1}), & \text{if } y_{i+1} > 0. \end{cases}$$

Set $b_{r+1} = \mathrm{lh}(\rho_2)$ and let ρ be a normal i-regular extension of $\rho_2 * z_{i+1}$. □

Corollary 21. *If $i \leq k$, then every i-regular finite part of rank 1 can be extended to a k-regular finite part of rank 1 and to a k-regular enumeration.*

Using similar arguments we may prove and the following proposition.

Proposition 22. *Let δ be an i-regular finite part. Let $y = 0$ or $y = \langle j, p\rangle + 1$ for some $j > i$. There exists a normal i-regular extension ρ of $\delta * y$ such that $(\forall x \in dom(\rho))(x > lh(\delta) \Rightarrow \rho(x) \simeq 0)$.*

Corollary 23. *For every $i \in \mathbb{N}$ there exists a* canonical *i-regular finite part δ_i of rank 1 such that $(\forall x \in dom(\delta_i))(\delta_i(x) \simeq 0)$.*

Now we are ready to present a proof of Theorem 11.

Proof (of Theorem 11).

Let us fix natural numbers $k_0 < \cdots < k_r$ and k. Given sets $A_0, \ldots, A_r$ of natural numbers, we define the sequence $\{B_i\}$ by setting

$$B_i = \begin{cases} A_j, \text{ if } i = k_j, \\ \emptyset, \quad \text{if } i \notin \{k_0, \ldots, k_r\}. \end{cases}$$

We call a finite part or an enumeration i-regular with respect to $A_0, \ldots, A_r$ if it is i-regular with respect to the sequence $\{B_i\}$.

As in the previous sections by $\overline{B}_i$ we denote $\mathbb{N} \oplus B_i$ and by P_i we denote the set

$$(...((\overline{B}_0)'_{\mathrm{e}} \oplus \overline{B}_1)'_{\mathrm{e}} \oplus \cdots \oplus \overline{B}i - 1)'_{\mathrm{e}} \oplus \overline{B}_i.$$

Clearly there exist computable functions $p_1(i)$ and $p_2(i)$ which do not depend on the choice of the sets $A_0, \ldots, A_r$ and such that

$$P_i = \Phi_{p_1(i)}(\mathcal{P}^{(i)}_{k_0,\ldots,k_r}(A_0, \ldots, A_r)) \text{ and } \mathcal{P}^{(i)}_{k_0,\ldots,k_r}(A_0, \ldots, A_r) = \Phi_{p_2(i)}(P_i).$$

Now let us consider a uniform operator Γ of type $(k_0, \ldots, k_r \to k)$. Let γ be the respective index function of Γ. By Proposition 19 there exist Oracle Turing Machines $b_{k_0}, \ldots, b_{k_r}$ such that for every $i \geq k_r$, every sequence of sets $A_0, \ldots, A_r$ and every i-regular enumeration f,

$$A_0 = W_{b_{k_0}}^{f^{(k_0)}}, \ldots, A_r = W_{b_{k_r}}^{f^{(k_r)}}.$$

Let $b = \gamma(b_{k_0}, \ldots, b_{k_r})$. Clearly for every sequence $A_0, \ldots, A_r$ of sets, for every $i \geq k_r$ and every i-regular enumeration f we have that

$$\Gamma(A_0, \ldots, A_r) = W_b^{f^{(k)}}.$$

Therefore there exists a c such that for every sequence $A_0, \ldots, A_r$ of sets, every $i \geq k_r$ and every i-regular enumeration f,

$$(\forall n)(f(n) \in \Gamma(A_0, \ldots, A_r) \iff f \models_k F_c(n)).$$

Consider the canonical k-regular finite part δ_k of rank 1. Set $n_0 = \mathrm{lh}(\delta_k)$. Let δ be a normal k-regular extension of $\delta_k * (\langle k+1, c\rangle + 1)$ such that $(\forall x \in \mathrm{dom}(\delta))(x > \mathrm{lh}(\delta_k) \Rightarrow \delta(x) \simeq 0)$. Let $\mathrm{lh}(\delta) = l_0$. We shall show that

$$x \in \Gamma(A_0, \ldots, A_r) \iff (\exists \tau \supseteq \delta)(\tau \text{ is } k\text{-regular with respect to } A_0, \ldots, A_r \ \& \ \tau(l_0) \simeq x \ \& \ \tau \Vdash_k F_c(l_0)).$$

Indeed, suppose that there exist a $\tau \supseteq \delta$ which is k-regular with respect to $A_0, \ldots A_r$, $\tau(l_0) \simeq x$ and $\tau \Vdash_k F_c(l_0)$. Then there exists a $\max(k_r, k+1)$-regular with respect to $A_0, \ldots, A_r$ enumeration f which extends the least such τ. Clearly $f \models_k F_c(l_0)$ and $f(l_0) \simeq x$. So, $x \in \Gamma(A_0, \ldots, A_r)$.

Suppose now that $x \in \Gamma(A_0, \ldots, A_r)$. Consider a $\max(k_r, k+1)$-regular enumeration f which extends $\delta * x$. Then $f(l_0) \simeq x$ and hence $f \models_k F_c(l_0)$. Then

there exists a $\tau \subseteq f$ such that $\tau \Vdash_k F_c(l_0)$. Clearly we may assume that $\delta \subseteq \tau$ and $\tau(l_0) \simeq x$.

From here using Proposition 18 one can find easily an enumeration operator Φ such that for all $A_1, \ldots, A_r$,

$$\Gamma(A_0, \ldots, A_r) = \Phi(\mathcal{P}^{(k)}_{k_0,\ldots,k_r}(A_0, \ldots, A_r)).$$

By this we have proved the nontrivial part of the theorem. The proof of the rest is routine. □

References

1. Rogers Jr., H.: Theory of recursive functions and effective computability. McGraw-Hill Book Company, New York (1967)
2. Selman, A.: Arithmetical reducibilities I. Z. Math. Logik Grundlag. Math. **17** (1971) 335–350
3. Case, J.: Maximal arithmetical reducibilities. Z. Math. Logik Grundlag. Math. **20** (1974) 261–270
4. Ash, C.: Generalizations of enumeration reducibility using recursive infinitary propositional sentences. Ann. Pure Appl. Logic **58** (1992) 173–184
5. Cooper, S.: Partial degrees and the density problem. Part 2: The enumeration degrees of the Σ_2 sets are dense. J. Symbolic Logic **49** (1984) 503–513
6. McEvoy, K.: Jumps of quasi-minimal enumeration degrees. J. Symbolic Logic **50** (1985) 839–848
7. Soskov, I.: A jump inversion theorem for the enumeration jump. Arch. Math. Logic **39** (2000) 417–437
8. Soskov, I., Baleva, V.: Regular enumerations. J. Symbolic Logic **67** (2002) 1323–1343

Minimal Pairs and Quasi-minimal Degrees for the Joint Spectra of Structures

Alexandra A. Soskova

Faculty of Mathematics and Computer Science,
Sofia University, 5 James Bourchier Blvd.
1164 Sofia, Bulgaria
asoskova@fmi.uni-sofia.bg

Abstract. Two properties of the Co-spectrum of the Joint spectrum of finitely many abstract structures are presented — a Minimal Pair type theorem and the existence of a Quasi-Minimal degree with respect to the Joint spectrum of the structures.

1 Introduction

Let $\mathfrak{A}$ be a countable abstract structure. The Degree spectrum $\mathrm{DS}(\mathfrak{A})$ of $\mathfrak{A}$ is the set of all enumeration degrees generated by all enumerations of $\mathfrak{A}$. The Co-spectrum of the structure $\mathfrak{A}$ is the set of all enumeration degrees which are lower bounds of the $\mathrm{DS}(\mathfrak{A})$. A typical example of a spectrum is the cone of the total degrees greater then or equal to some enumeration degree $\mathbf{a}$ and the respective Co-spectrum which is equal to the set all degrees less than or equal to $\mathbf{a}$. There are examples of structures with more complicated degree spectra e.g. [5, 4, 1, 3, 7]. The properties of the Degree spectra are presented in [7] which show that the degree spectra behave with respect to their Co-spectra like the cones of enumeration degrees.

In [8] a generalization of the notions of Degree spectra and Co-spectra for finitely many structures is presented. Let $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ be countable abstract structures. The Joint spectrum of $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ is the set $\mathrm{DS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$ of all elements of $\mathrm{DS}(\mathfrak{A}_0)$, such that $\mathbf{a}^{(k)} \in \mathrm{DS}(\mathfrak{A}_k)$, for each $k \leq n$.

Here we shall prove two properties of the Co-spectrum of $\mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$ — the Minimal Pair type theorem and the existence of a quasi-minimal degree with respect to the Joint spectrum.

The proofs use the technique of regular enumerations introduced in [6], combined with partial generic enumerations used in [7].

2 Preliminaries

Let $\mathfrak{A} = (\mathbb{N}; R_1, \ldots, R_k)$ be a partial structure over the set of all natural numbers $\mathbb{N}$, where each R_i is a subset of $\mathbb{N}^{r_i}$ and $=$ and $\neq$ are among $R_1, \ldots, R_k$.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 451–460, 2005.

An enumeration f of $\mathfrak{A}$ is a total mapping from $\mathbb{N}$ onto $\mathbb{N}$.

For every $A \subseteq \mathbb{N}^a$ define $f^{-1}(A) = \{\langle x_1 \dots x_a \rangle : (f(x_1), \dots, f(x_a)) \in A\}$. Denote by $f^{-1}(\mathfrak{A}) = f^{-1}(R_1) \oplus \dots \oplus f^{-1}(R_k)$.

For any sets of natural numbers A and B the set A is enumeration reducible to B ($A \leq_e B$) if there is an enumeration operator Γ_z such that $A = \Gamma_z(B)$. By $d_e(A)$ we denote the enumeration degree of the set A and by $\mathcal{D}_e$ the set of all enumeration degrees. The set A is total if $A \equiv_e A^+$, where $A^+ = A \oplus (\mathbb{N} \backslash A)$. A degree $\mathbf{a}$ is called total if $\mathbf{a}$ contains a total set. The jump operation "$'$" denotes here the enumeration jump introduced by Cooper [2].

Definition 1. *The Degree spectrum of* $\mathfrak{A}$ is the set

$$\mathrm{DS}(\mathfrak{A}) = \{d_e(f^{-1}(\mathfrak{A})) : f \text{ is an enumeration of } \mathfrak{A}\} \ .$$

Let $B_0, \dots, B_n$ be arbitrary subsets of $\mathbb{N}$. Define the set $\mathcal{P}(B_0, \dots, B_i)$ as follows:

1. $\mathcal{P}(B_0) = B_0$;
2. If $i < n$, then $\mathcal{P}(B_0, \dots, B_{i+1}) = (\mathcal{P}(B_0, \dots, B_i))' \oplus B_{i+1}$.

In the construction of minimal pair we shall use a modification of the "type omitting" version of Jump Inversion Theorem from [6]. In fact, the result follows from the proof of the Theorem 1.7 in [6].

Theorem 2 ([6]). *Let $\{A_r^k\}_r$, $k = 0, \dots, n$ be a sequence of subsets of $\mathbb{N}$ such that for every r and for all k, $0 \leq k < n$, $A_r^k \not\leq_e \mathcal{P}(B_0, \dots, B_k)$. Then there exists a total set F having the following properties:*

1. *$B_i \leq_e F^{(i)}$, for all $i \leq n$;*
2. *$A_r^k \not\leq_e F^{(k)}$, for all r and all $k < n$.*

Definition 3. A set F of natural numbers is called *quasi-minimal over* B_0 if the following conditions hold:

1. $B_0 <_e F$;
2. For any total set $A \subseteq \mathbb{N}$, if $A \leq_e F$, then $A \leq_e B_0$.

In the construction of the quasi-minimal degree we shall use the following fact:

Theorem 4. *There exists a set of natural numbers F having the following properties:*

1. *$B_0 <_e F$;*
2. *For all $1 \leq i \leq n$, $B_i \leq_e F^{(i)}$;*
3. *For any total set A, if $A \leq_e F$, then $A \leq_e B_0$.*

The set F from Theorem 4 is quasi-minimal over B_0. We shall prove this theorem in the last section using the technique of partial regular enumerations.

3 Joint Spectra of Structures

Let $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ be abstract structures on $\mathbb{N}$.

Definition 5. *The Joint spectrum of* $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ is the set

$$\mathrm{DS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n) = \{\mathbf{a} : \mathbf{a} \in \mathrm{DS}(\mathfrak{A}_0), \mathbf{a}' \in \mathrm{DS}(\mathfrak{A}_1), \ldots, \mathbf{a}^{(n)} \in \mathrm{DS}(\mathfrak{A}_n)\} \ .$$

Definition 6. For every $k \leq n$, *the k th Jump spectrum of* $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ is the set

$$\mathrm{DS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n) = \{\mathbf{a}^{(k)} : \mathbf{a} \in \mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)\} \ .$$

In [8] is shown that $\mathrm{DS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$ is closed upwards, i.e. if $\mathbf{a}^{(k)} \in \mathrm{DS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$, $\mathbf{b}$ is a total e-degree and $\mathbf{a}^{(k)} \leq \mathbf{b}$, then $\mathbf{b} \in \mathrm{DS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$.

Definition 7. *The kth Co-spectrum of* $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$, $k \leq n$, is the set of all lower bounds of $\mathrm{DS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$, i.e.

$$\mathrm{CS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n) = \{\mathbf{b} : \mathbf{b} \in \mathcal{D}_\mathrm{e} \& (\forall \mathbf{a} \in \mathrm{DS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n))(\mathbf{b} \leq \mathbf{a})\} \ .$$

From [8] we know that the kth Co-spectrum for $k \leq n$ depends only of the first k structures:

$$\mathrm{CS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_k, \ldots, \mathfrak{A}_n) = \mathrm{CS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_k) \ .$$

Let $f_0, \ldots, f_n$ be enumerations of $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$. Denote by $\bar{f} = (f_0, \ldots, f_n)$ and $\mathcal{P}_k^{\bar{f}} = \mathcal{P}(f_0^{-1}(\mathfrak{A}_0), \ldots, f_k^{-1}(\mathfrak{A}_k))$, $k = 0, \ldots, n$.

Let $W_0, \ldots, W_z, \ldots$ be a Gödel's enumeration of the c.e. sets and D_v be the finite set having canonical code v.

For every $i \leq n$, e and x in $\mathbb{N}$ define the relations $\bar{f} \models_i F_e(x)$ and $\bar{f} \models_i \neg F_e(x)$ by induction on i:

1. $\bar{f} \models_0 F_e(x) \iff (\exists v)(\langle v, x\rangle \in W_e \ \& \ D_v \subseteq f_0^{-1}(\mathfrak{A}_0))$;
2. $\bar{f} \models_{i+1} F_e(x) \iff (\exists v)(\langle v, x\rangle \in W_e \ \& \ (\forall u \in D_v)(u = \langle 0, e_u, x_u\rangle \ \&$ $\bar{f} \models_i F_{e_u}(x_u) \ \vee u = \langle 1, e_u, x_u\rangle \ \&\ \bar{f} \models_i \neg F_{e_u}(x_u) \ \vee \ u = \langle 2, x_u\rangle \ \&$ $x_u \in f_{i+1}^{-1}(\mathfrak{A}_{i+1})))$;
3. $\bar{f} \models_i \neg F_e(x) \iff \bar{f} \not\models_i F_e(x)$.

It is easy to check that for any $A \subseteq \mathbb{N}$ and $k \leq n$

$$A \leq_\mathrm{e} \mathcal{P}_k^{\bar{f}} \iff (\exists e)(A = \{x : \bar{f} \models_k F_e(x)\}) \ .$$

The forcing conditions which we shall call *finite parts* are $n+1$ tuples $\bar{\tau} = (\tau_0, \ldots, \tau_n)$ of finite mappings $\tau_0, \ldots, \tau_n$ of $\mathbb{N}$ in $\mathbb{N}$.

For any $i \leq n$, e and x in $\mathbb{N}$ and every finite part $\bar{\tau}$ we define the forcing relations $\bar{\tau} \Vdash_i F_e(x)$ and $\bar{\tau} \Vdash_i \neg F_e(x)$ following the definition of relation "$\models_i$".

Definition 8. 1. $\bar{\tau} \Vdash_0 F_e(x) \iff (\exists v)(\langle v, x\rangle \in W_e \ \&\ D_v \subseteq \tau_0^{-1}(\mathfrak{A}_0))$;
2. $\bar{\tau} \Vdash_{i+1} F_e(x) \iff \exists v(\langle v, x\rangle \in W_e \ \&\ \ (\forall u \in D_v)(u = \langle 0, e_u, x_u\rangle \ \&$
$\bar{\tau} \Vdash_i F_{e_u}(x_u) \ \vee\ u = \langle 1, e_u, x_u\rangle \ \&\ \bar{\tau} \Vdash_i \neg F_{e_u}(x_u) \ \vee\ \ u = \langle 2, x_u\rangle \ \&$
$x_u \in \tau_{i+1}^{-1}(\mathfrak{A}_{i+1})))$;
3. $\bar{\tau} \Vdash_i \neg F_e(x) \iff (\forall \bar{\rho} \supseteq \bar{\tau})(\bar{\rho} \nVdash_i F_e(x))$.

For any $i \leq n, e, x \in \mathbb{N}$ denote by $X^i_{\langle e,x\rangle} = \{\bar{\rho} : \bar{\rho} \Vdash_i F_e(x)\}$.

Definition 9. An enumeration $\bar{f}$ of $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ is *i-generic* if for every $j < i$, $e, x \in \mathbb{N}$

$$(\forall \bar{\tau} \subseteq \bar{f})(\exists \bar{\rho} \in X^j_{\langle e,x\rangle})(\bar{\tau} \subseteq \bar{\rho}) \Longrightarrow (\exists \bar{\tau} \subseteq \bar{f})(\bar{\tau} \in X^j_{\langle e,x\rangle}) \ .$$

In [8] the following properties of the k-generic enumertions are shown:

1. If $\bar{f}$ is an k-generic enumeration, then

$$\bar{f} \models_k F_e(x) \iff (\exists \bar{\tau} \subseteq \bar{f})(\bar{\tau} \Vdash_k F_e(x)) \ .$$

2. If $\bar{f}$ is an $(k+1)$-generic enumeration, then

$$\bar{f} \models_k \neg F_e(x) \iff (\exists \bar{\tau} \subseteq \bar{f})(\bar{\tau} \Vdash_k \neg F_e(x)) \ .$$

Definition 10. The set $A \subseteq \mathbb{N}$ is *forcing k-definable* on $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ if there exist a finite part $\bar{\delta}$ and $e \in \mathbb{N}$ such that

$$x \in A \iff (\exists \bar{\tau} \supseteq \bar{\delta})(\bar{\tau} \Vdash_k F_e(x)) \ .$$

In [8] the following characterization of the sets which generates the elements of the kth Co-spectrum of $\mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$ is given:

Theorem 11 ([8]). *For every $A \subseteq \mathbb{N}$, the following are equivalent:*

1. $d_{\mathrm{e}}(A) \in \mathrm{CS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$.
2. $A \leq_{\mathrm{e}} \mathcal{P}_k^{\bar{f}}$, *for all* $\bar{f} = (f_0, \ldots, f_k)$ *enumerations of* $\mathfrak{A}_0, \ldots, \mathfrak{A}_k$.
3. A *is forcing k-definable on* $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$.

Theorem 12. *Let $\{X_r^k\}_r$, $k = 0, \ldots, n$ be $n+1$ sequences of sets of natural numbers. There exists a $(n+1)$-generic enumeration $\bar{f}$ of $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ such that for any $k \leq n$ and for all $r \in \mathbb{N}$, if the set X_r^k is not forcing k-definable on $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$, then $X_r^k \not\leq_{\mathrm{e}} \mathcal{P}_k^{\bar{f}}$.*

4 Minimal Pair Theorem

In [7] a Minimal Pair Theorem for Degree spectrum of a structure $\mathfrak{A}$ is presented. Using the technique of splitting generic enumerations it is proven there that for

each constructive ordinal α there exist elements $\mathbf{f}$ and $\mathbf{g}$ of $DS(\mathfrak{A})$ such that for any enumeration degree $\mathbf{a}$ and any $\beta < \alpha$

$$\mathbf{a} \leq \mathbf{f}^{(\beta)} \ \& \ \mathbf{a} \leq \mathbf{g}^{(\beta)} \Rightarrow \mathbf{a} \in CS_\beta(\mathfrak{A}) \ .$$

We shall prove an analogue of the Minimal Pair Theorem for the Joint spectrum.

Theorem 13. *For all structures $\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n$, there exist enumeration degrees $\mathbf{f}$ and $\mathbf{g}$ in $\mathrm{DS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$, such that for any enumeration degree $\mathbf{a}$ and $k \leq n$:*

$$\mathbf{a} \leq \mathbf{f}^{(k)} \ \& \ \mathbf{a} \leq \mathbf{g}^{(k)} \Rightarrow \mathbf{a} \in \mathrm{CS}_k(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n) \ .$$

Proof. We shall construct two total sets F and G, such that $d_{\mathrm{e}}(F) \in \mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$, $d_{\mathrm{e}}(G) \in \mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$ and for each $k \leq n$, if a set X, $X \leq_{\mathrm{e}} F^{(k)}$ and $X \leq_{\mathrm{e}} G^{(k)}$, then $d_{\mathrm{e}}(X) \in \mathrm{CS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$. And take $\mathbf{f} = d_{\mathrm{e}}(F)$ and $\mathbf{g} = d_{\mathrm{e}}(G)$.

First we construct enumerations $\bar{f}$ and $\bar{h}$ of $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$ such that for any $k \leq n$, if a set $A \subseteq \mathbb{N}$, $A \leq_{\mathrm{e}} \mathcal{P}_k^{\bar{f}}$ and $A \leq_{\mathrm{e}} \mathcal{P}_k^{\bar{h}}$, then A is a forcing k-definable on $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$.

Let $g_0, \ldots, g_n$ be arbitrary enumerations of $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$. By Theorem 2 for $B_0 = g_0^{-1}(\mathfrak{A}_0), \ldots, B_n = g_n^{-1}(\mathfrak{A}_n)$ there exists a total set F, such that: $g_0^{-1}(\mathfrak{A}_0) \leq_{\mathrm{e}} F$, $g_1^{-1}(\mathfrak{A}_1) \leq_{\mathrm{e}} F', \ldots, g_n^{-1}(\mathfrak{A}_n) \leq_{\mathrm{e}} F^{(n)}$. Since $\mathrm{DS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$ is closed upwards, then $d_{\mathrm{e}}(F) \in \mathrm{DS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$, i.e. $d_{\mathrm{e}}(F) \in \mathrm{DS}(\mathfrak{A}_0), d_{\mathrm{e}}(F') \in \mathrm{DS}(\mathfrak{A}_1), \ldots, d_{\mathrm{e}}(F^{(n)}) \in \mathrm{DS}(\mathfrak{A}_n)$.

Hence, there exist enumerations $h_0, h_1, \ldots, h_n$ of $\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n$, respectively, such that $h_0^{-1}(\mathfrak{A}_0) \equiv_{\mathrm{e}} F, h_1^{-1}(\mathfrak{A}_1) \equiv_{\mathrm{e}} F', \ldots, h_n^{-1}(\mathfrak{A}_n) \equiv_{\mathrm{e}} F^{(n)}$. Notice, that for each $k \leq n$, $F^{(k)} \equiv_{\mathrm{e}} \mathcal{P}_k^{\bar{h}}$.

For each $k \leq n$, let $\{X_r^k\}_r$ be the sequence of all sets enumeration reducible to $\mathcal{P}_k^{\bar{h}}$.

By Theorem 12 there is an $(n+1)$-generic enumeration $\bar{f}$ such that for all r, and all $k = 0, \ldots, n$ if the set X_r^k is not forcing k-definable then $X_r^k \not\leq_{\mathrm{e}} \mathcal{P}_k^{\bar{f}}$.

Suppose now that the set $A \leq_{\mathrm{e}} \mathcal{P}_k^{\bar{f}}$ and $A \leq_{\mathrm{e}} \mathcal{P}_k^{\bar{h}}$. Then $A = X_r^k$ for some r. From the omitting condition of $\bar{f}$ it follows that A is forcing k-definable on $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$.

Now we apply again the Theorem 2. Let $B_0 = f_0^{-1}(\mathfrak{A}_0), \ldots, B_n = f_n^{-1}(\mathfrak{A}_n)$ and $B_{n+1} = N$. For each $k \leq n$ consider the sequence $\{A_r^k\}_r$ of these sets among the sets $\{X_r^k\}_r$, which are not forcing k-definable on $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$. From the choice of the enumeration $\bar{f}$ it follows that each of these sets A_r^k, $A_r^k \not\leq_{\mathrm{e}} \mathcal{P}_k^{\bar{f}}$. Then by Theorem 2 there is a total set G, such that

1. For all $k \leq n$, $f_k^{-1}(\mathfrak{A}_i) \leq_{\mathrm{e}} G^{(k)}$;
2. For all r and all $k \leq n$, $A_r^k \not\leq_{\mathrm{e}} G^{(k)}$.

Note, that since G is a total set, and because of the fact that each spectrum is closed upwards, we have that $d_{\mathrm{e}}(G) \in \mathrm{DS}(\mathfrak{A}_0), \ldots, d_{\mathrm{e}}(G^{(n)}) \in \mathrm{DS}(\mathfrak{A}_n)$, and hence $d_{\mathrm{e}}(G) \in \mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$.

Suppose now, that a set X, $X \leq_e F^{(k)}$ and $X \leq_e G^{(k)}$, $k \leq n$. From $X \leq_e F^{(k)}$ and $F^{(k)} \equiv_e \mathcal{P}_k^{\bar{h}}$, it follows that $X = X_r^k$ for some r. It is clear that $X \leq_e \mathcal{P}_k^{\bar{f}}$. Otherwise from the choice of G it follows that $X \not\leq_e G^{(k)}$. Hence X is forcing k-definable on $\mathfrak{A}_0, \ldots, \mathfrak{A}_n$. By the normal form of the sets which enumeration degrees are in $\mathrm{CS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$, we have that $d_e(X) \in \mathrm{CS}_k(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$. □

5 Quasi-minimal Degree

Given a set $\mathcal{A}$ of enumeration degrees denote by $co(\mathcal{A})$ the set of all lower bounds of $\mathcal{A}$. Say that the degree $\mathbf{q}$ is *quasi-minimal with respect to* $\mathcal{A}$ if the following conditions hold ([7]):

1. $\mathbf{q} \notin co(\mathcal{A})$.
2. If $\mathbf{a}$ is a total degree and $\mathbf{a} \geq \mathbf{q}$, then $\mathbf{a} \in \mathcal{A}$.
3. If $\mathbf{a}$ is a total degree and $\mathbf{a} \leq \mathbf{q}$, then $\mathbf{a} \in co(\mathcal{A})$.

In [7] it is shown that there is a quasi-minimal degree $\mathbf{q}_0$ with respect to $\mathrm{DS}(\mathfrak{A}_0)$, i.e. $\mathbf{q}_0 \notin \mathrm{CS}(\mathfrak{A}_0)$ and for every total degree $\mathbf{a}$: if $\mathbf{a} \geq \mathbf{q}_0$, then $\mathbf{a} \in \mathrm{DS}(\mathfrak{A}_0)$ and if $\mathbf{a} \leq \mathbf{q}_0$, then $\mathbf{a} \in \mathrm{CS}(\mathfrak{A}_0)$.

We are going to prove the existence of a quasi-minimal degree with respect to $\mathrm{DS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$.

Theorem 14. *For all structures $\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n$ there exists an enumeration degree $\mathbf{q}$ such that:*

1. $\mathbf{q}' \in \mathrm{DS}(\mathfrak{A}_1), \ldots, \mathbf{q}^{(n)} \in \mathrm{DS}(\mathfrak{A}_n)$, $\mathbf{q} \notin \mathrm{CS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$;
2. *If $\mathbf{a}$ is a total degree and $\mathbf{a} \geq \mathbf{q}$, then $\mathbf{a} \in \mathrm{DS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$;*
3. *If $\mathbf{a}$ is a total degree and $\mathbf{a} \leq \mathbf{q}$, then $\mathbf{a} \in \mathrm{CS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$.*

Proof. Let $\mathbf{q}_0$ be a quasi-minimal degree $\mathbf{q}_0$ with respect to $\mathrm{DS}(\mathfrak{A}_0)$ from [7].

Let $B_0 \subseteq \mathbb{N}$, such that $d_e(B_0) = \mathbf{q}_0$, and $f_1, \ldots, f_n$ be fixed total enumerations of $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$. Set $B_1 = f_1^{-1}(\mathfrak{A}_1), \ldots, B_n = f_n^{-1}(\mathfrak{A}_n)$. By Theorem 4 there is quasi-minimal over B_0 set F, such that $B_0 <_e F$, $B_i \leq_e F^{(i)}$, for each $1 \leq i \leq n$, and moreover for any total set A, if $A \leq_e F$, then $A \leq_e B_0$. We will show that $\mathbf{q} = d_e(F)$ is quasi-minimal with respect to $\mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$.

Since $\mathbf{q}_0$ is quasi-minimal with respect to $\mathrm{DS}(\mathfrak{A}_0)$, $\mathbf{q}_0 \notin \mathrm{CS}(\mathfrak{A}_0)$. But $\mathbf{q}_0 < \mathbf{q}$ and thus $\mathbf{q} \notin \mathrm{CS}(\mathfrak{A}_0)$. Hence $\mathbf{q} \notin \mathrm{CS}(\mathfrak{A}_0, \mathfrak{A}_1, \ldots, \mathfrak{A}_n)$.

For each $1 \leq i \leq n$, the set $F^{(i)}$ is total and $f_i^{-1}(\mathfrak{A}_i) \leq_e F^{(i)}$. Since any degree spectrum is closed upwards it follows that $d_e(F^{(i)}) \in \mathrm{DS}(\mathfrak{A}_i)$, i.e. $\mathbf{q}^{(i)} \in \mathrm{DS}(\mathfrak{A}_i)$.

Consider a total set X, such that $X \geq_e F$. Then $d_e(X) \geq \mathbf{q}_0$. From the fact that $\mathbf{q}_0$ is quasi-minimal with respect to $\mathrm{DS}(\mathfrak{A}_0)$ it follows that $d_e(X) \in \mathrm{DS}(\mathfrak{A}_0)$. Moreover for each $1 \leq i \leq n$, $X^{(i)} \geq_e F^{(i)} \geq_e f_i^{-1}(\mathfrak{A}_i)$, and $X^{(i)}$ is a total set. Then for each $i \leq n$, $d_e(X^{(i)}) \in \mathrm{DS}(\mathfrak{A}_i)$, and hence $d_e(X) \in \mathrm{DS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$.

Suppose that X is a total set and $X \leq_e F$. Then, from the choice of F, since X is total, $X \leq_e B_0$. Apply again the quasi-minimality of $\mathbf{q}_0$ and then $d_e(X) \in \mathrm{CS}(\mathfrak{A}_0)$. But $\mathrm{CS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n) = \mathrm{CS}(\mathfrak{A}_0)$ and therefore $d_e(X) \in \mathrm{CS}(\mathfrak{A}_0, \ldots, \mathfrak{A}_n)$. □

In the rest of the paper we shall present the proof of Theorem 4.

6 Partial Regular Enumerations

Let $B_0, \ldots, B_n$ be fixed sets of natural numbers. Combining the technique of the (total) regular enumerations from [6] with the partial generic enumerations, introduced in [7], we shall construct a partial regular enumeration f, which graph will be quasi-minimal over the set B_0 and such that $B_i \leq_e f^{(i)}$, for $0 \leq i \leq n$. In [7] a partial generic enumeration of B_0 is constructed, which is quasi-minimal over B_0. In addition, the enumeration f we are going to obtain, will code the sets $B_1, \ldots, B_n$ in its jumps ($B_i \leq_e f^{(i)}$).

Definition 15. *A partial enumeration f of B_0* is a partial surjective mapping from $\mathbb{N}$ onto $\mathbb{N}$ with the following properties:

1. For all odd x, if $f(x)$ is defined, then $f(x) \in B_0$;
2. For all $y \in B_0$, there is an odd x, such that $f(x) \simeq y$.

It is clear that if f is a partial enumeration of B_0, then $B_0 \leq_e f$.

Let $\perp \notin \mathbb{N}$.

Definition 16. A *partial finite part* τ is a finite mapping of $\mathbb{N}$ into $\mathbb{N} \cup \{\perp\}$, such that $(\forall x)(x \in \mathrm{dom}(\tau)\ \&\ x \text{ is odd} \Rightarrow (\tau(x) = \perp \vee \tau(x) \in B_0))$.

If τ is a partial finite part and f is a partial enumeration of B_0, say that

$$\tau \subseteq f \iff (\forall x \in \mathrm{dom}(\tau))((\tau(x) = \perp \Rightarrow f(x) \text{ is not defined }) \ \&\ (\tau(x) \neq \perp \Rightarrow \tau(x) \simeq f(x)) \ .$$

A 0-*regular partial finite part* is a partial finite part τ such that $\mathrm{dom}(\tau) = [0, 2q+1]$ and for all odd $z \in \mathrm{dom}(\tau)$, $\tau(z) \in B_0$ or $\tau(z) = \perp$. The 0-rank of τ, $|\tau|_0 = q+1$ we call the number of the odd elements of $\mathrm{dom}(\tau)$. If ρ is a 0-regular partial extention of τ we shall denote this by $\tau \subseteq_0 \rho$. It is clear that if $\tau \subseteq_0 \rho$ and $|\tau|_0 = |\rho|_0$, then $\tau = \rho$. Let

$$\tau \Vdash_0 F_e(x) \iff \exists v(\langle v, x\rangle \in W_e \ \&\ (\forall u \in D_v)(u = \langle s, t\rangle, \ \&\ \tau(s) \simeq t \ \&\ t \neq \perp))$$

$$\tau \Vdash_0 \neg F_e(x) \iff (\forall \rho)(\tau \subseteq_0 \rho \Rightarrow \rho \not\Vdash_0 F_e(x)) \ .$$

The $(i+1)$-regular partial finite part τ, the $(i+1)$-rank $|\tau|_{i+1}$ of τ and the relations $\tau \Vdash_{i+1} F_e(x)$ and $\tau \Vdash_{i+1} \neg F_e(x)$ are defined by induction on i, in the same way as in [6]. The only difference is that instead of i-regular finite parts we use i-regular partial finite parts. Denote by $\mathcal{R}_i$ the set of all i-regular partial finite parts.

For any i-regular finite part τ and any set X of i-regular finite parts, denote by $\mu_i(\tau, X) = \mu\rho[\tau \subseteq \rho \ \&\ \rho \in \mathcal{R}_i \ \&\ \rho \in X]$ if any, and $\mu_i(\tau, X) = \mu\rho[\tau \subseteq \rho \ \&\ \rho \in \mathcal{R}_i]$, otherwise.

Denote by $X^i_{\langle e,x\rangle} = \{\rho : \rho \text{ is } i\text{-regular } \&\ \rho \Vdash_i F_e(x)\}$.

Let τ be a finite part and $m \geq 0$. The finite part δ is called an *i-regular m omitting extension* of τ if $\delta \supseteq \tau$, $\delta \in \mathcal{R}_i$, $\mathrm{dom}(\delta) = [0, q-1]$ and there exist natural numbers $q_0 < \cdots < q_m < q_{m+1} = q$ such that:

1. $\delta \upharpoonright q_0 = \tau$.
2. For all $p \leq m$, $\delta \upharpoonright q_{p+1} = \mu_i(\delta \upharpoonright (q_p + 1), X^i_{\langle p,q_p \rangle})$.

If δ and ρ are two i-regular m omitting extensions of τ and $\delta \subseteq \rho$ then $\delta = \rho$. Given an index j, by S^i_j we shall denote the intersection $\mathcal{R}_i \cap \Gamma_j(\mathcal{P}(B_0, \ldots, B_i))$, where Γ_j is the jth enumeration operator.

Let τ be a finite part defined on $[0, q-1]$ and $r \geq 0$. Then τ is $(i+1)$-*regular* with $(i+1)$-*rank* $r+1$ if there exist natural numbers

$$0 < n_0 < l_0 < b_0 < n_1 < l_1 < b_1 \cdots < n_r < l_r < b_r < n_{r+1} = q$$

such that $\tau \upharpoonright n_0$ is an i-regular finite part with i-rank equal to 1 and for all j, $0 \leq j \leq r$, the following conditions are satisfied:

(a) $\tau \upharpoonright l_j \simeq \mu_i(\tau \upharpoonright (n_j + 1), S^i_j)$;
(b) $\tau \upharpoonright b_j$ is an i-regular j omitting extension of $\tau \upharpoonright l_j$;
(c) $\tau(b_j) \in B_{i+1}$;
(d) $\tau \upharpoonright n_{j+1}$ is an i-regular extension of $\tau \upharpoonright (b_j + 1)$ with i-rank equal to $|\tau \upharpoonright b_j|_i + 1$.

If τ is an i-regular partial finite part, then τ is a j-regular partial finite part for each $j < i$ and $|\tau|_j > |\tau|_i$.

Definition 17. *A partial regular enumeration* is a partial enumeration, such that:

1. For every partial finite part $\delta \subseteq f$, there exists an n-regular partial extension τ of δ such that $\tau \subseteq f$.
2. If $i \leq n$ and $z \in B_i$, then there exists an i-regular partial finite part $\tau \subseteq f$, such that $z \in \mathrm{dom}(\tau)$.

If f is a partial regular enumeration, $\delta \subseteq f$ and $i \leq n$, then there exists an i-regular partial finite part τ of an arbitrary large rank such that $\delta \subseteq \tau$ and $\tau \subseteq f$.

Denote by $\mathcal{P}_i = \mathcal{P}(B_0, \ldots, B_i)$. It is clear that $\mathcal{R}_i \leq_e \mathcal{P}_i$.

Definition 18. A partial enumeration f is *i-generic* if for any $j < i$ and for every enumeration reducible to $\mathcal{P}_j$ set S of j-regular partial finite parts the following condition holds:

$$(\exists \tau \subseteq f)(\tau \in S \vee (\forall \rho \supseteq \tau)(\rho \in \mathcal{R}_i \Rightarrow \rho \notin S)) \ .$$

Proposition 19. *Every partial regular enumeration is $(i+1)$-generic enumeration, for every $i < n$.*

Proposition 20. *Suppose that f is a partial regular enumeration. Then*

1. *For each $i \leq n$, $B_i \leq_e f^{(i)}$.*
2. *If $i < n$, then $f \not\leq_e \mathcal{P}_i$.*

Definition 21. If f is a partial enumeration define:

$$f \models_0 F_e(x) \iff \exists v(\langle v, x\rangle \in W_e \ \&\ (\forall u \in D_v)(f((u)_0) \simeq (u)_1)) \ .$$

Proof of Theorem 4. By Proposition 20 it is sufficient to show that there exists a partial regular enumeration f which is quasi-minimal over B_0.

We shall construct f as a union of n-regular partial finite parts δ_s such that for all s, $\delta_s \subseteq \delta_{s+1}$ and $|\delta_s|_n = s+1$. Suppose that for $i \leq n$, σ_i is a recursively in B_i enumeration of B_i.

Let δ_0 be a 0-regular partial finite part such that $|\delta_0|_n = 1$. Suppose that δ_s is defined. Set $z_0 = \sigma_0(s), \ldots, z_n = \sigma_n(s)$. We can construct effectively in $\mathcal{P}'_{n-1}$ a n-regular partial finite part $\rho \supseteq \delta_s$ such that $|\rho|_n = |\delta_s|_n + 1$, $\rho(\mathrm{lh}(\delta_s)) = s$ and $z_0 = \rho(x_0)$ for some $x_0 \in B_0, \ldots, z_n = \rho(x_n)$ for some $x_n \in B_n$. Set $\delta_{s+1} = \rho$.

The obtained enumeration f is surjective on $\mathbb{N}$ and it is a union of n-regular partial finite parts. From the construction is obvious that for every $z \in B_i$ there is an i-regular partial finite part τ of f, such that $z \in \mathrm{dom}(\tau)$. Hence f is a partial regular enumeration. By Proposition 19 f is $(i+1)$-generic for each $i < n$.

Then by Proposition 20, for $i \leq n$, $B_i \leq f^{(i)}$. Moreover f is a partial 1-generic enumeration and hence $B_0 <_{\mathrm{e}} f$.

To prove that f is quasi-minimal over B_0, it is sufficient to show that if ψ is a total function and $\psi \leq_{\mathrm{e}} f$, then $\psi \leq_{\mathrm{e}} B_0$. It is clear that for any total set $A \subseteq \mathbb{N}$ one can construct a total function ψ, $\psi \equiv_{\mathrm{e}} A$. Let ψ be a total function and $\psi = \Gamma_e(f)$. Then

$$(\forall x, y \in \mathbb{N})(f \models_0 F_e(\langle x, y\rangle) \iff \psi(x) \simeq y) \ .$$

Consider the set

$$S_0 = \{\rho : \rho \in \mathcal{R}_0 \ \&\ (\exists x, y_1 \neq y_2 \in \mathbb{N})(\rho \Vdash_0 F_e(\langle x, y_1\rangle) \ \&\ \rho \Vdash_0 F_e(\langle x, y_2\rangle))\} \ .$$

Since $S_0 \leq_{\mathrm{e}} B_0$, we have that there exists a 0-regular partial finite part $\tau_0 \subseteq f$ such that either $\tau_0 \in S_0$ or $(\forall \rho \supseteq_0 \tau_0)(\rho \notin S_0)$. Assume that $\tau_0 \in S_0$. Then there exist $x, y_1 \neq y_2$ such that $f \models_0 F_e(\langle x, y_1\rangle)$ and $f \models_0 F_e(\langle x, y_2\rangle)$. Then $\psi(x) \simeq y_1$ and $\psi(x) \simeq y_2$ which is impossible. So, $(\forall \rho \supseteq_0 \tau_0)(\rho \notin S_0)$.

Let

$$\begin{aligned} S_1 = \{\rho : \rho \in \mathcal{R}_0 \ \&\ (\exists \tau \supseteq_0 \tau_0)(\exists \delta_1 \supseteq_0 \tau)(\exists \delta_2 \supseteq_0 \tau)(\exists x, y_1 \neq y_2)(\tau \subseteq_0 \rho \ \&\ \\ \delta_1 \Vdash_0 F_e(\langle x, y_1\rangle) \ \&\ \delta_2 \Vdash_0 F_e(\langle x, y_2\rangle) \ \&\ \mathrm{dom}(\rho) = \mathrm{dom}(\delta_1) \cup \mathrm{dom}(\delta_2) \\ \&(\forall x)(x \in \mathrm{dom}(\rho) \setminus \mathrm{dom}(\tau) \Rightarrow \rho(x) \simeq \perp))\} \ . \end{aligned}$$

We have that $S_1 \leq_{\mathrm{e}} B_0$ and hence there exists a 0-regular partial finite part $\tau_1 \subseteq f$ such that either $\tau_1 \in S_1$ or $(\forall \rho \supseteq_0 \tau_1)(\rho \notin S_1)$.

Assume that $\tau_1 \in S_1$. Then there exists a 0-regular partial finite part τ such that $\tau_0 \subseteq_0 \tau \subseteq_0 \tau_1$ and for some $\delta_1 \supseteq_0 \tau$, $\delta_2 \supseteq_0 \tau$ and $x_0, y_1 \neq y_2 \in \mathbb{N}$ we have

$$\begin{aligned} &\delta_1 \Vdash_0 F_e(\langle x_0, y_1\rangle) \ \&\ \delta_2 \Vdash_0 F_e(\langle x_0, y_2\rangle) \ \&\ \mathrm{dom}(\tau_1) = \mathrm{dom}(\delta_1) \cup \mathrm{dom}(\delta_2) \ \& \\ &\&\ (\forall x)(x \in \mathrm{dom}(\tau_1) \setminus \mathrm{dom}(\tau) \Rightarrow \tau_1(x) \simeq \perp) \ . \end{aligned}$$

Let $\psi(x_0) \simeq y$. Then $f \models_0 F_e(\langle x_0, y\rangle)$. Hence there exists a $\rho \supseteq_0 \tau_1$ such that $\rho \Vdash_0 F_e(\langle x_0, y\rangle)$. Let $y \neq y_1$. Define the partial finite part ρ_0 as follows:

$$\rho_0(x) \simeq \begin{cases} \delta_1(x) \text{ if } x \in \mathrm{dom}(\delta_1), \\ \rho(x) \text{ if } x \in \mathrm{dom}(\rho) \setminus \mathrm{dom}(\delta_1). \end{cases}$$

Then $\tau_0 \subseteq_0 \rho_0$, $\delta_1 \subseteq_0 \rho_0$ and notice that for all $x \in \mathrm{dom}(\rho)$ if $\rho(x) \not\simeq \perp$, then $\rho(x) \simeq \rho_0(x)$. Hence $\rho_0 \Vdash_0 F_e(\langle x_0, y_1\rangle)$ and $\rho_0 \Vdash_0 F_e(\langle x_0, y\rangle)$. So, $\rho_0 \in S_0$. A contradiction.

Thus, $(\forall \rho)(\rho \supseteq_0 \tau_1 \Rightarrow \rho \notin S_1)$.

Let $\tau = \tau_1 \cup \tau_0$. Notice that $\tau \subseteq f$. We shall show that

$$\psi(x) \simeq y \iff (\exists \delta \supseteq_0 \tau)(\delta \Vdash_0 F_e(\langle x, y\rangle)) \ .$$

And hence $\psi \leq_{\mathrm{e}} B_0$.

If $\psi(x) \simeq y$, then $f \models_0 F_e(x)$, and since f is regular, $(\exists \rho \subseteq f)(\rho \Vdash_0 F_e(x))$ and ρ is 0-regular. Then take $\delta = \tau \cup \rho$.

Assume that $\delta_1 \supseteq_0 \tau$, $\delta_1 \Vdash_0 F_e(\langle x, y_1\rangle)$. Suppose that $\psi(x) \simeq y_2$ and $y_1 \neq y_2$. Then there exists a $\delta_2 \supseteq_0 \tau$ such that $\delta_2 \Vdash_0 F_e(\langle x, y_2\rangle)$. Set

$$\rho(x) \simeq \begin{cases} \tau(x) \text{ if } x \in \mathrm{dom}(\tau), \\ \perp \quad \text{if } x \in (\mathrm{dom}(\delta_1) \cup \mathrm{dom}(\delta_2)) \setminus \mathrm{dom}(\tau). \end{cases}$$

Clearly $\rho \supseteq_0 \tau_1$ and $\rho \in S_1$. A contradiction. □

References

1. C. J. Ash, C. Jockush, and J. F. Knight, *Jumps of orderings*, Trans. Amer. Math. Soc. **319** (1990), 573–599.
2. S. B. Cooper, *Partial degrees and the density problem. Part 2: The enumeration degrees of the Σ_2 sets are dense*, J. Symbolic Logic **49** (1984), 503–513.
3. R. G. Downey and J. F. Knight, *Orderings with αth jump degree $\mathbf{0}^{(\alpha)}$*, Proc. Amer. Math. Soc. **114** (1992), 545–552.
4. J. F. Knight, *Degrees coded in jumps of orderings*, J. Symbolic Logic **51** (1986), 1034–1042.
5. L. J. Richter, *Degrees of structures*, J. Symbolic Logic **46** (1981), 723–731.
6. I. N. Soskov, *A jump inversion theorem for the enumeration jump*, Arch. Math. Logic **39** (2000), 417–437.
7. ———, *Degree spectra and co-spectra of structures*, Ann. Univ. Sofia, **96**, 2003, 45–68.
8. A. A. Soskova, I. N. Soskov, *Co-spectra of joint spectra of structures*, Ann. Univ. Sofia, **96**, 2003, 35–44.

Presentations of K-Trivial Reals and Kolmogorov Complexity

Frank Stephan[1,⋆] and Guohua Wu[2,3,⋆⋆]

[1] School of Computing, National University of Singapore,
3 Science Drive 2, Singapore 117543, Singapore
fstephan@comp.nus.edu.sg
[2] School of Mathematics, Statistics and Computer Science,
Victoria University of Wellington,
Wellington 6001, New Zealand
[3] School of Physical and Mathematical Sciences,
Nayang Technological University, Singapore 637616
wu.guohua@mcs.vuw.ac.nz

Abstract. For given real $\alpha \in \{0,1\}^\infty$, a presentation V of α is a prefix-free and recursively enumerable subset of $\{0,1\}^*$ such that $\alpha = \sum_{\sigma \in V} 2^{-|\sigma|}$. So, α has a presentation iff α is a left-r.e. real. Let $\mathcal{A}$ be the class of all reals which have only computable presentations. Downey and LaForte proved that $\mathcal{A}$ has an incomputable member. Call α strongly Kurtz-random if there does not exist any recursive function f with $K(\alpha(0)\ldots\alpha(f(n)-1)) < f(n)-n$ for all n. It is shown that every $\alpha \in \mathcal{A}$ is either computable or strongly Kurtz-random. In particular, all K-trivial members of $\mathcal{A}$ are computable where α is K-trivial iff there is a c such that, for all n, $K(\alpha(0)\ldots\alpha(n-1)) < K(n)+c$. Thus there is a natural and nontrivial Turing ideal of left-r.e. α not containing any incomputable member of $\mathcal{A}$.

1 Introduction

In this paper, we work with machines with input and output alphabets $\{0,1\}$. A Turing machine M is *prefix-free* if $M(\tau)\!\downarrow \Rightarrow M(\tau')\!\uparrow$ for all finite binary strings $\tau' \succeq \tau$ where $\succeq$ denotes string extension. It is *universal* if for each prefix-free machine N there is a constant c such that, for all binary strings τ, if $N(\tau)\downarrow$ then there is some σ such that $|\sigma| \leq |\tau| + c$ and $M(\sigma)\downarrow = N(\tau)$. c is called the *coding constant* of N. For a prefix-free machine M and a binary string τ, $K_M(\tau)$ denotes the length of the shortest binary string σ such that $M(\sigma)\!\downarrow = \tau$, if such σ exists and let $K_M(\tau)$ be undefined otherwise. In the following we fix

⋆ F. Stephan is supported in part by NUS grant number R252–000–212–112.
⋆⋆ G. Wu is supported by the New Zealand FRST Post-Doctoral Fellowship, the International Joint Project No. 60310213 of NSFC of China, and the Fast-Start Grant from Nanyang Technological University.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 461–469, 2005.

a universal prefix-free machine U and let $K(\tau) = K_U(\tau)$, which is called the *Kolmogorov complexity* of τ. We remark here that the choice of U does not affect the Kolmogorov complexity by more than an additive constant.

For convenience, we identify subsets of the natural numbers with the reals between 0 and 1. So $\alpha \in \{0,1\}^\infty$ stands for both, for the set $\{n : \alpha(n) = 1\}$ and for the real number $\sum_{n=0,1,\ldots} \alpha(n) \cdot 2^{-n-1}$. We write $\alpha < \beta$ iff α is less than β as a real number. By $\alpha[m]$ we denote the string $\alpha(0)\alpha(1)\ldots\alpha(m-1)$ which we identify with the finite set $\{n \in \alpha : n < m\}$ and the number $\sum_{n<m} \alpha(n) \cdot 2^{-n-1}$. α is called recursively enumerable iff it is recursively enumerable as a set and it is called left-r.e. iff it has a recursive approximation $\alpha_0, \alpha_1, \ldots$ such that $\alpha_n \leq \alpha$ for all n; the left-r.e. sets are also known as "nearly computable" and in the case that they are more viewed upon as real numbers than as sets, they are often just referred to as "r.e. reals". In order to keep the notation between numbers and sets consistent, we do not follow this notation and our "r.e. reals" correspond to those which some people call "strongly r.e." in order to distinguish them from the left-r.e. reals. Note that we can choose the approximation such that $\alpha_0 < \alpha_1 < \ldots < \alpha$ and $\alpha_n \subset \{0, 1, \ldots, h(n)\}$ for some computable function h and all n.

Calude, Hertling, Khoussainov and Wang [4] proved that α is left-r.e. if and only if there is an infinite *recursively enumerable* prefix-free set of strings V such that $\alpha = \sum_{\sigma \in V} 2^{-|\sigma|}$ if and only if there is an infinite computable prefix-free set of strings V such that $\alpha = \sum_{\sigma \in V} 2^{-|\sigma|}$, where a set V is *prefix-free* means that if $\sigma \in V$, then no extension of σ can be in V. So, based on the convention that $\alpha = \sum_{n=0,1,\ldots} 2^{-1-n}\alpha(n)$, Downey and LaForte [10] introduced the following notion.

Definition 1. *V is a presentation of α iff V is a recursively enumerable and prefix-free set with $\alpha = \sum_{\sigma \in V} 2^{-|\sigma|}$.*

Downey and LaForte [10] proved that there are incomputable sets having only computable presentations.

In this paper, we will give a characterization for the existence of an incomputable presentation in terms of the existence of a certain type of approximation. This characterization will be the basis for showing that the Kolmogorov complexity of any incomputable left-r.e. α without incomputable presentations cannot be bounded for any recursively selected set of prefixes of α, neither from below nor from above. More precisely, if α is incomputable and left-r.e. set which has only computable presentations and f is a computable function then neither $\forall n\,(K(\alpha[f(n)]) > n)$ nor $\forall n\,(K(\alpha[f(n)]) < f(n) - n)$.

The first condition can be interpreted as α being unable to wtt-compute fixed-point free functions, that is, of α being wtt-incomplete. This condition follows already from the result of Downey and LaForte [10] that every left-r.e. set of promptly simple degree has an incomplete presentation.

So the really interesting condition is the second one. The second notion is a stronger version of Kurtz-randomness and thus we assign to it the corresponding name.

Definition 2. *An $\alpha \in \{0,1\}^\infty$ is* strongly Kurtz-random *iff there is no computable function f such that $K(\alpha[f(n)]) < f(n) - n$ for all n.*

Normally Kurtz-random is defined in terms of computable martingales. We drop the computability constraint and consider the martingale M given as the sum over all $M_{\tau,\sigma}$ with $U(\tau) = \sigma$ and

$$M_{\tau,\sigma}(\eta) = \begin{cases} 2^{|\sigma|-|\tau|} & \text{if } \sigma \prec \eta; \\ 2^{|\eta|-|\tau|} & \text{if } \eta \preceq \sigma; \\ 0 & \text{otherwise.} \end{cases}$$

Then α is strongly Kurtz-random if there is no computable function f with $M(\alpha[f(n)]) \geq n$ for all n. Since M is a universal martingale, this condition is stronger than the following one: $(*)$ There is no computable martingale $\widetilde{M}$ and no computable function f such that $\widetilde{M}(\alpha[f(n)]) \geq n$ for all n. It follows from Wang's martingale characterization [16] that $(*)$ is equivalent to Kurtz-randomness. This justifies our notion "strongly Kurtz-random" which we use to state the first main result that every left-r.e. set is either computable or has an incomputable presentation or is strongly Kurtz-random.

Theorem 3. *If α is left-r.e., incomputable and every presentation of α is computable then α is strongly Kurtz-random.*

As a direct corollary, every such α is also Kurtz-random with respect to the original definition. Theorem 3 is also the basis for the other main result of the paper. We say that α is *K-trivial* if the Kolmogorov complexity of each initial segment of α is minimal, that is, if there is a constant c such that for all n, $K(\alpha[n]) \leq K(n)+c$. K-trivial reals are interesting due to the connection between algorithmic complexity and effective randomness. We, for example, exploit the fact that no K-trivial set is Kurtz-random. Chaitin [5] proved that if an α is K-trivial then α must be Δ_2. In 1974, Solovay [15] proved that there is an incomputable K-trivial real. Downey, Hirschfeldt, Nies and Stephan [9] proved the existence of an incomputable recursively enumerable K-trivial real. Recently, Nies [12] proved that all the K-trivials are low and that α is K-trivial if and only if α is low for random if and only if there is a constant c such that for all x, $K(x) \leq K^\alpha(x) + c$, that is, adding the oracle α does not change the prefix-free complexity by more than a constant. In this paper, we show that every incomputable left-r.e. K-trivial has always an incomputable presentation and that thus Downey and LaForte's reals cannot be K-trivial.

Theorem 4. *Every K-trivial and incomputable left-r.e. real α has an incomputable presentation.*

Our notation is standard and generally follows the books of Cutland [7], Downey and Hirschfeldt [8], Odifreddi [13] and Soare [14]; but we differ from the notation in [7] by requiring that computable functions are total. Since we identify subsets of the natural numbers with reals, we use the names of notions in the version which is defined for sets.

2 A Characterization

The existence of an r.e. but incomputable presentation can be obtained from the following characterization.

Theorem 5. *Let α be left-r.e. and incomputable. Then α has an incomputable presentation iff there is a recursive approximation $\alpha_0, \alpha_1, \ldots$ such that*

(1) $\alpha_0 < \alpha_1 < \ldots < \alpha$;
(2) *there is a recursive function h such that $\alpha_n \subseteq \{0, 1, \ldots, h(n)\}$ for all n;*
(3) *for every computable function g and every m there exist $n \geq m$ and $s \geq g(n)$ with $\alpha_{s+1} \geq \alpha_s + 2^{-n}$.*

Proof. Assume that α has an incomputable presentation V. Now let $\sigma_0, \sigma_1, \ldots$ be a one-one recursive enumeration of V; this sequence is infinite since V is not computable. Now define an approximation α_s by the equation

$$\alpha_s = \sum_{t<s} 2^{-|\sigma_t|}$$

where $\alpha_0 = 0$. Clearly (1) is satisfied. (2) is obtained by defining

$$h(s) = \max\{|\sigma_t| : 0 \leq t \leq s\}.$$

Now consider any computable function g and any m. Since V is not computable there is a length $n > m$ and an element of V of length n which is not enumerated into V within $g(n)$ steps. It follows that this element is of the form σ_s for some $s \geq g(n)$. Thus $\alpha_{s+1} \geq \alpha_s + 2^{-n}$. So (3) is satisfied.

For the converse direction, let $\alpha_0, \alpha_1, \ldots$ be a recursive approximation of α satisfying the three conditions together via the computable function h. Now define recursive sets T_s to consist of all strings τ such that

(a) $\alpha_s \leq \tau$, where τ is interpreted as the real number $\sum_{k<|\tau|} 2^{-1-k}\tau(k)$;
(b) the reals given by k and τ differ at a digit and the least position k of such a digit satisfies $k < |\tau|$, $\tau(k) = 0$ and $\alpha_{s+1}(k) = 1$;
(c) every proper prefix $\tau' \prec \tau$ violates either (a) or (b).

Here $|\tau|$ is the length of τ and τ can end with a 0. Note that the $\tau \in T_s$ are the shortest prefixes of reals between β with $\alpha_s \leq \beta < \alpha_{s+1}$ such that every binary extension γ of τ, including the extension $\tau 1111\ldots$, also satisfies $\alpha_s \leq \gamma < \alpha_{s+1}$. Thus every set T_s is prefix-free and satisfies $\sum_{\tau \in T_s} 2^{-|\tau|} = \alpha_{s+1} - \alpha_s$. Let V be the union of all T_s. It is easy to see that V is also prefix-free. Since the α_s are monotonically increasing, the sets T_s are not empty and the equations

$$\alpha = \sum_s (\alpha_{s+1} - \alpha_s) = \sum_s \sum_{\tau \in T_s} 2^{-|\tau|} = \sum_{\tau \in V} 2^{-|\tau|}$$

hold. By the existence of h, the sets T_s are uniformly recursive and thus V is recursively enumerable. So V is a presentation for α.

For any given s, let $l(s)$ be the length of the shortest string of T_s. It is easy to see that $-2 - \log(\alpha_{s+1} - \alpha_s) < l(s) < 2 - \log(\alpha_{s+1} - \alpha_s)$. For example, assume that $\alpha_s = \{2, 3, 5, 7\}$ and $\alpha_{s+1} = \{2, 3, 4, 5, 6, 9\}$. Then α_s and α_{s+1} are the real numbers 0.00110101 and 0.0011111001 in binary notation. The value $\alpha_{s+1} - \alpha_s$ is 0.0000100101 and $-\log(\alpha_{s+1} - \alpha_s)$ is between 5 and 6. The shortest string in T_s is 001110 since the binary numbers since $\alpha_s \leq 0.001110000\ldots$ and $0.0011101111\ldots = 0.001111000\ldots < \alpha_{s+1}$. The value of $l(s)$ is $|001110| = 6$.

To complete the proof, assume by way of contradiction that V is recursive. One can compute for any given n the value $g(n) = \max\{t : t = 0 \text{ or } t = s + 1 \text{ and } \exists \tau \in T_s\, (|\tau| \leq n + 2)\}$ by simply taking all $\tau \in V$ with $|\tau| \leq n + 2$ and then searching for the s with $\tau \in T_s$, which is possible since the T_s are uniformly recursive. It follows that for every $s \geq g(n)$, T_s does not contain any string of length $n+2$ or less and thus $\alpha_{s+1} < \alpha_s + 2^{-n}$. This contradicts the third property of the approximation $\alpha_0, \alpha_1, \ldots$ from the statement of the theorem. □

An application of this result is that every α which has only computable presentations is either recursive or hyperimmune. The proof of the following result is a model for the proof of the first main result that every left-r.e. and incomputable set without incomputable presentation is Kurtz-random.

Theorem 6. *Let α be left-r.e., incomputable and not hyperimmune. Then α has an incomputable presentation.*

Proof. Since α is not hyperimmune, there is a computable function f such that $\alpha \cap \{n+1, n+2, \ldots, f(n)\}$ is not empty for every n. Let $\beta_0, \beta_1, \ldots$ be a recursive approximation of α from below. Now one defines inductively a new approximation $\alpha_0, \alpha_1, \ldots$ and a recursive function h such that $\alpha_n = \beta_s[f(t) + 1]$ and $h(n) = t$ for the first pair (s, t) in some fixed enumeration of all pairs of natural numbers with

(d) $s > h(m)$ and $t > h(m)$ for all $m < n$;
(e) $\alpha_m < \beta_s[f(t) + 1]$ for all $m < n$;
(f) β_s intersects $\{m + 1, m + 2, \ldots, f(m)\}$ for all $m \leq t$.

Then the resulting approximation $\alpha_0, \alpha_1, \ldots$ of α has the following properties.

(g) $\alpha_0 < \alpha_1 < \ldots < \alpha$;
(h) $h(n + 1) > h(n)$ and $\alpha_n \subseteq \{0, 1, \ldots, f(h(n))\}$ for all n;
(i) the intersection $\alpha_n \cap \{m + 1, m + 2, \ldots, f(m)\}$ is not empty for any n and any $m \leq h(n)$.

It remains to show that (3) of Theorem 5 is true for every computable function g and m. Suppose not. Let g be a computable function with $g(x) > x$ for all x and m be a number at which (3) is false. Then, for any $n \geq m$ and $s \geq g(n)$, $\alpha_{s+1} < \alpha_s + 2^{-n}$. Thus, for any given x, $f(x) + m + 2 > m$, and for any $s > g(f(x) + m + 2)$, $\alpha_{s+1} < \alpha_s + 2^{-f(x)-m-2}$.

Now fix x. Then by (h), $h(x) > x$. Let $t(x) = \max\{h(x), g(f(x) + m + 2)\}$. We prove that for any $s > t(x)$, $\alpha_s(x) = \alpha_{t(x)}(x)$ and hence $\alpha(x) = \alpha_{t(x)}(x)$.

Suppose not, and let $s > t(x)$ be a stage with $\alpha_{s+1}(x) \neq \alpha_s(x)$. Then $h(s+1) > h(x) > x$. By the choice of the approximation $\alpha_0, \alpha_1, \ldots$ and (i), there is some y in $\{x+1, x+2, \ldots, f(x)\}$ such that $\alpha_{s+1}(y) = 1$. As a consequence,

$$\alpha_{s+1} \geq \alpha_s + 2^{-y} \geq \alpha_s + 2^{-f(x)} > \alpha_s + 2^{-f(x)-m-2},$$

a contradiction.

Thus, for any x, $\alpha(x) = \alpha_{t(x)}(x)$ and α would be computable, contradicting the assumption of α. Thus (3) of Theorem 5 is also satisfied and hence α has an incomputable but recursively enumerable presentation. □

Since every wtt-complete left-r.e. set has promptly simple degree, it has an incomputable presentation, by a result of Downey and LaForte [10]. An application of Theorem 6 gives an alternative proof for this fact. The connection to Kolmogorov complexity is based on the fact that a left-r.e. α is wtt-complete iff it can wtt-compute an DNR function [2, 3]. Note that Arslanov's criterion also applies to left-r.e. sets since those are tt-equivalent to r.e. ones. Furthermore, Kjos-Hanssen, Merkle and Stephan [11] proved that a set wtt-computes a DNR function iff there is a computable function f such that $K(\alpha[f(n)]) > n$ for all n. Theorem 7 is given in terms of Kolmogorov complexity and not in terms of wtt-completeness.

Theorem 7. (Downey and LaForte [10]) *If α is left-r.e. and there is a computable function f such that, for all n, $K(\alpha[f(n)]) > n$ then α has an incomputable presentation.*

Proof. Without loss of generality one can take f to be monotonically increasing; so assume that for α there is a monotonically increasing function f such that $K(\alpha[f(n)]) > n$ for all n. Now let $g(0) = 1$ and $g(n+1) = f(4^{g(n)})$. Given a string $\tau \in \{0,1\}^{g(n)}$, one can compute from τ the number n, the value $g(n+1)$ and the string $\tau 0^{g(n+1)-g(n)}$. Since there is a constant c with $K(\tau) < 3^{|\tau|} + c$ for all $\tau \in \{0,1\}^*$, it holds for almost all n that

$$K(\alpha[g(n+1)]) = K(\alpha[f(4^{g(n)})]) > 4^{g(n)} \geq K(\alpha[g(n)] \cdot 0^{g(n+1)-g(n)}).$$

Thus, $\alpha(g(n))\alpha(g(n)+1)\ldots\alpha(g(n+1)-1)$ differs from $0^{g(n+1)-g(n)}$ for almost all n and α cannot be a hyperimmune set. It follows from Theorem 6 that α has an incomputable presentation. □

3 The Main Results

We now prove that every α which has a recursively enumerable but not incomputable presentation is either computable or strongly Kurtz-random. That is, the following second main result is proven on the way to get the first one. Since strongly Kurtz-random sets are Kurtz-random, a direct corollary is that every left-r.e. and incomputable α where every presentation of α is computable, is also Kurtz-random.

Theorem 3. *If α is left-r.e., incomputable and every presentation of α is computable then α is strongly Kurtz-random.*

Proof. Assume that α is incomputable, left-r.e. and not strongly Kurtz-random. We will prove that α has an incomputable presentation. This is done by showing that there is an approximation $\alpha_0, \alpha_1, \ldots$ satisfying the three conditions of Theorem 5.

Let f be a computable function witnessing that α is not strongly Kurtz-random: $\forall n\,(K(\alpha[f(n)]) < f(n) - n)$. Furthermore, let $\beta_0, \beta_1, \ldots$ be an approximation of α from below. Now one defines inductively a new approximation $\alpha_0, \alpha_1, \ldots$ and a recursive function h such that $\alpha_n = \beta_s[f(t)]$ and $h(n) = t$ for the first pair (s, t) with

(j) $s > h(m)$ and $t > h(m)$ for all $m < n$;
(k) $\alpha_m < \beta_s[f(t)]$ for all $m < n$;
(l) $K_s(\beta_s[f(m)]) < f(m) - m$ for all $m \leq t$.

The resulting approximation $\alpha_0, \alpha_1, \ldots$ of α has the following properties:

(m) $\alpha_0 < \alpha_1 < \ldots < \alpha$;
(n) $h(n+1) > h(n)$ and $\alpha_n \subseteq \{0, 1, \ldots, f(h(n))\}$ for all n;
(o) $K_s(\alpha_s[f(m)]) < f(m) - m$ for all n and $m \leq h(n)$.

It remains to show that (3) of Theorem 5 is also true.

Assume by way of contradiction that this is false for a function g and an m; without loss of generality, $g(n) > n$ for all n. Now for any $n \geq m$, search for the first pair (o, t) found such that

(p) $o > n$;
(q) $h(t) > o + 1$;
(r) $t > g(f(o+1))$;
(s) $\alpha_t(o) = 0$.

This search terminates since there α has the digit 0 infinitely often and the search condition is satisfied whenever $\alpha(o) = 0$ and t is sufficiently large. So the search for o and t can be realized by a computable function.

For every $s \geq t$, *by (q)*, $K(\alpha_s[f(o+1)]) < f(o+1) - o - 1$. Thus, by the choice of m and $o > n \geq m$, there are less than $2^{f(o+1)-o-1}$ many stages $s \geq t$ with $\alpha_{s+1}[f(o+1)] \neq \alpha_s[f(o+1)]$. Each of them satisfies that $\alpha_{s+1} - \alpha_s < 2^{-f(o+1)}$ by $s \geq g(f(o+1))$. Therefore, $\alpha_{s+1}[f(o+1)] = 2^{-f(o+1)} + \alpha_s[f(o+1)]$. Since there are less than $2^{f(o+1)-o-1}$ many places $s \geq t$ where $\alpha_{s+1}[f(o+1)] \neq \alpha_s[f(o+1)]$ and since the value goes up each time by exactly $2^{-f(o+1)}$, one has

$$\alpha[f(o+1)] < \alpha_t[f(o+1)] + 2^{f(o+1)-o-1} \cdot 2^{-f(o+1)} = \alpha_t[f(o+1)] + 2^{-o-1}.$$

Since $\alpha_t(o) = 0$, the finite $\beta = \{o\} \cup \alpha_t[f(o+1)]$ satisfies $\alpha_s[f(o+1)] \leq \beta$ for all s. It follows that $\alpha(n) = \alpha_t(n)$ and one can compute $\alpha(n)$ by searching for (o, t) as above and taking the approximation $\alpha_t(n)$ as the desired value. Since α is incomputable, the assumed g does not exist. Thus the conditions on the approximation $\alpha_0, \alpha_1, \ldots$ from Theorem 5 are all satisfied and hence α has an incomputable presentation. □

One might ask whether also randomness notions stronger than Kurtz-randomness are implied. The answer is negative. Kurtz-randomness is something special since for each not Kurtz-random set there is a computable martingale M and a recursive function f such that $M(\alpha[f(n)]) > n$ for all n. The other randomness notions require the corresponding condition only for infinitely many n. Thus randomness happens "much less frequently" and no hyperimmune set is Schnorr-random or Martin-Löf-random. So it follows from Theorem 6 that every Schnorr-random and every Martin-Löf-random left-r.e. set has an incomputable presentation.

In order to see the other main result, note that there is a constant c with $K(n) < c + n/2$ for all n. If α is K-trivial then there is a constant c' such that $K(\alpha[n]) < K(n) + c'$. Taking $f(n) = 2n + c + c'$, one has $K(\alpha[f(n)]) < f(n) - n$ for all n. Thus a K-trivial set is not strongly Kurtz-random and so one has the following.

Theorem 4. *Every K-trivial and incomputable left-r.e. real α has an incomputable presentation.*

4 A Degree-Theoretic Conclusion

Theorem 4 has some degree-theoretic consequences since the K-trivial sets are closed downward under Turing reduction and since below every incomputable r.e. degree there is an incomputable, K-trivial r.e. degree. Let $\mathbf{R}$ be the set of nonzero r.e. Turing degrees and $\mathbf{A}$ be the subset of those degrees in $\mathbf{R}$ which contain an left-r.e. α without an incomputable presentation. Then every $\mathbf{a}$ in $\mathbf{R}$ bounds an $\mathbf{b} \in \mathbf{R}$ not bounding any degree in $\mathbf{A}$. That is,

$$\forall \mathbf{a} \in \mathbf{R}\, \exists \mathbf{b} \in \mathbf{R}\, \forall \mathbf{c} \in \mathbf{R}\, (\mathbf{b} \leq \mathbf{a} \wedge (\mathbf{c} \leq \mathbf{b} \Rightarrow \mathbf{c} \notin \mathbf{A})).$$

In particular, every $\mathbf{a} \in \mathbf{R}$ bounds a degree in $\mathbf{R} - \mathbf{A}$. Nies [12] showed that there is a K-trivial and promptly simple set, let $\mathbf{s}$ be its degree. As seen, no $\mathbf{a} \leq \mathbf{s}$ is in $\mathbf{A}$. Ambos-Spies, Jockusch, Shore and Soare [1] showed that the degrees of promptly simple sets form a filter in the r.e. Turing degrees. Following the result of Downey and LaForte [10] that every left-r.e. of promply simple degree has an incomputable presentation, one has that the degrees of incomputable left-r.e. sets without incomputable presentation are incomparable to $\mathbf{s}$:

$$\forall \mathbf{a} \in \mathbf{A}\, (\mathbf{a} \not\leq \mathbf{s} \wedge \mathbf{a} \not\geq \mathbf{s}).$$

So, the Turing degrees of the incomputable left-r.e. sets having only computable presentations are disjoint from a filter and from a ideal in the Turing degrees where this filter and this ideal intersect. Both the ideal and the filter are important: Nies [12] showed that the K-trivial sets are closed under Turing reduciblity and that they coincide with several other natural notions being around for many years [12]; Ambos-Spies, Jockusch, Shore and Soare [1] showed that the degrees of promptly simple set coincide with the non-cappable degrees and the low cuppable degrees, see [14–Chapter XIII] for a discussion.

References

1. Klaus Ambos-Spies, Carl Jockusch, R. Shore and Robert Soare. *An algebraic decomposition of the recursively enumerable degrees and classes equal to the promptly simple degrees*, Transactions of the American Mathematical Society 281:109–128, 1984.
2. Marat M. Arslanov, *On some generalizations of the Fixed-Point Theorem*, Soviet Mathematics (Iz. VUZ), Russian, 25(5):9–16, 1981, English translation, 25(5):1–10, 1981.
3. Marat M. Arslanov, *M-reducibility and fixed points*, Complexity problems of mathematical logic, Collection of scientific Works, Russian, Kalinin, pages 11-18, 1985.
4. Cristian Calude, Peter Hertling, Bakhadyr Khoussainov, Yongge Wang, *Recursively enumerable reals and Chaitin's Ω number*, in STACS 1998, Lecture Notes in Computer Science 1373:596–606, 1998.
5. Gregory Chaitin, *A theory of program size formally identical to information theory*, Journal of the Association for Computing Machinery 22:329–340, 1975, reprinted in [6].
6. Gregory Chaitin, *Information, Randomness & Incompleteness*, 2nd edition, Series in Computer Science 8, World Scientific, River Edge, NJ, 1990.
7. Nigel Cutland, *Computability, an introduction to recursive function theory*, Cambridge University Press, Cambridge, 1980.
8. Rod Downey and Denis Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, in preparation.
9. Rod Downey, Denis Hirschfeldt, André Nies and Frank Stephan, *Trivial reals*, Proceedings of the 7th and 8th Asian Logic Conferences, World Scientific, River Edge, NJ, pages 103–131, 2003.
10. Rod Downey, Geoffrey LaForte, *Presentations of computably enumerable reals*, Theoretical Computer Science, to appear.
11. Bjørn Kjos-Hanssen, Wolfgang Merkle and Frank Stephan, *Kolmogorov complexity and the Recursion theorem*, Manuscript, 2005.
12. André Nies, *Lowness properties and randomness*, Advances in Mathematics, to appear.
13. Piergiorgio Odifreddi, *Classical recursion theory,* Volume 1, North-Holland, Amsterdam 1989, Volume 2, Elsevier, Amsterdam 1999.
14. Robert Soare, *Recursively enumerable sets and degrees*, Springer, Heidelberg, 1987.
15. Robert Solovay, *Draft of a paper* (*or series of papers*) *on Chaitin's work*, unpublished manuscript, 215 pages, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1975.
16. Yongge Wang, *Randomness and Complexity*, PhD Dissertation, University of Heidelberg, 1996.
17. Guohua Wu, *Prefix-free languages and initial segments of computably enumerable degrees*, in: *Computing and Combinatorics*, Lecture Notes in Computer Science 2108:576-585, 2001.

Presentations of Structures in Admissible Sets*

Alexey Stukachev

Sobolev Institute of Mathematics,
Siberian Branch of the Russian Academy of Sciences,
4 Acad. Koptyug avenue,
630090 Novosibirsk, Russia
aistu@math.nsc.ru

Abstract. We consider copies and constructivizations of structures in admissible sets. In the first section we survey results about copies of countable structures in hereditary finite superstructures and definability (so called syntactical conditions of intrinsically computable properties) and state some conjectures about the uncountable case. The second section is devoted to constructivizations of uncountable structures in "simplest" uncountable admissible sets (more precisely, in hereditary finite superstructures over the models of c-simple theories). The third section contains some results on constructivizations of admissible sets within themselves.

1 Copies of Structures in Admissible Sets

The classical computable model theory studies computable properties of structures by considering their presentations on natural numbers. This means that only countable structures can be studied this way. One of the most natural ways to avoid this restriction is to consider presentations of structures in admissible sets. The theory of admissible sets [2] gives a beautiful example of interaction between model theory, computability theory and set theory.

We consider copies and constructivizations of structures in admissible sets. It is well known that in classical computable model theory (on natural numbers) these approaches are equivalent: a structure has a computable (decidable) copy if and only if it is constructivizable (strongly constructivizable). However, in admissible sets the "if" part of this statement is not true in general, so we must consider two different cases.

The notations we use in this paper are standard and corresponds to [1, 2]. For a structure $\mathfrak{M}$ and admissible set $\mathbb{A}$ their domains are denoted by M and A respectively. We denote by $F(\sigma)$ the set of finite first order formulas of a signature σ. We also fix some Gödel numbering $\ulcorner\cdot\urcorner : F(\sigma) \to \omega$ (so $\ulcorner\varphi\urcorner$ is the

* This work was supported by the INTAS YSF (Grant 04-83-3310), the Program "Universities of Russia" (Grant UR.04.01.488), the Russian Foundation for Basic Research (Grant 02-01-00540) and the Council for the Grants of the President of RF and the State Support of the Leading Scientific Schools (Grant NS.2069.2003.1).

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 470–478, 2005.

Gödel number of a formula φ). In all that follows we consider only computable signatures and suppose that Gödel numberings are effective. We also denote by $F_n(\sigma)$ $(n \leqslant \omega)$ the set of (finite first order) formulas of signature σ with no more than n alterating groups of quantifiers in prenex normal form. $F_0(\sigma)$ is the set of quantifier-free formulas of signature σ.

Let $\mathfrak{M}$ be a structure of signature σ, $\mathbb{A}$ an admissible set, and let $M \subseteq A$. Then the atomic diagram

$$D(\mathfrak{M}) = \{\langle \ulcorner \varphi \urcorner, \bar{m} \rangle |\ \varphi \in F_0(\sigma) - \text{atomic formula}, \bar{m} \in M^{<\omega},\ \mathfrak{M} \models \varphi(\bar{m})\}$$

is in fact a subset of A.

Definition 1. *Let $\mathfrak{M}$ be a structure of computable signature σ, $\mathbb{A}$ an admissible set, and let $M \subseteq A$. The structure $\mathfrak{M}$ is n-decidable in $\mathbb{A}$ $(n \leqslant \omega)$ if*

$$\{\langle \ulcorner \varphi \urcorner, \bar{m} \rangle \mid \varphi \in F_n(\sigma), \bar{m} \in M^{<\omega}, \mathfrak{M} \models \varphi(\bar{m})\}$$

is Δ-definable in $\mathbb{A}$.

A structure $\mathfrak{M}$ is *computable in* $\mathbb{A}$ if $\mathfrak{M}$ is 0-decidable in $\mathbb{A}$, and *decidable in* $\mathbb{A}$ if $\mathfrak{M}$ is ω-decidable in $\mathbb{A}$. It is obvious that if $\mathfrak{M}$ is n-decidable in $\mathbb{A}$ for some n then M is Δ-definable in $\mathbb{A}$.

It is easy to show that a structure $\mathfrak{M}$ is computable (decidable) in the classical sense if and only if it is computable (decidable) in the least admissible set $\mathbb{HF}(\varnothing)$.

Definition 2. *Let $\mathbb{A}$ be an admissible set and $P(A)$ be the set of all subsets of A. $F : P(A)^n \to P(A)$ is a Σ-operator if there exists a Σ-formula $\Phi(x_0, \ldots, x_{n-1}, y)$ such that for any $S_0, \ldots, S_{n-1} \in P(A)$*

$$F(S_0, \ldots, S_{n-1}) = \{a \mid \exists a_0 \ldots \exists a_{n-1}(\bigwedge_{i<n} a_i \subseteq S_i \wedge \mathbb{A} \models \Phi(a_0, \ldots, a_{n-1}, a))\}.$$

Let $F : P(A)^n \to P(A)$ be a Σ-operator and $\delta_{\mathrm{c}}(F)$ be a set of elements of $P(A)^n$ in which F is strongly continous [1]. It is easy to show that in $\mathbb{HF}(\mathfrak{M})$ any subset belongs to $\delta_{\mathrm{c}}(F)$ for any Σ-operator F.

Definition 3. *Suppose B, C are subsets of an admissible set $\mathbb{A}$. B is Σ-reducible to C $(B \leqslant_\Sigma C)$ if there exists a binary Σ-operator F_0 such that $\langle C, A \setminus C \rangle \in \delta_c(F_0)$ and $B = F_0(C, A \setminus C)$. If besides there exists binary Σ-operator F_1 such that $\langle C, A \setminus C \rangle \in \delta_c(F_1)$ and $A \setminus B = F_1(C, A \setminus C)$ then B is said to be $T\Sigma$-reducible to C $(B \leqslant_{T\Sigma} C)$.*

Let $\mathbb{A}$ be an admissible set, $\mathfrak{M}$ a structure such that $M \subseteq A$, and let $P \subseteq M^n$. P is *relatively computable in* $\mathbb{A}$ if P is $T\Sigma$-reducible to $D(\mathfrak{M})$ in $\mathbb{A}$, and *relatively c.e. in* $\mathbb{A}$ if P is Σ-reducible to $D(\mathfrak{M})$ in $\mathbb{A}$.

Definition 4. *Let $\mathfrak{M}$ be a structure of computable signature σ, $\mathbb{A}$ an admissible set, and let $M \subseteq A$. The structure $\mathfrak{M}$ is relatively n-decidable in $\mathbb{A}$ $(n \leqslant \omega)$ if*

$$\{\langle \ulcorner \varphi \urcorner, \bar{m} \rangle \mid \varphi \in F_n(\sigma), \bar{m} \in M^{<\omega}, \mathfrak{M} \models \varphi(\bar{m})\}$$

is $T\Sigma$-reducible to $D(\mathfrak{M})$ in $\mathbb{A}$.

Definition 5. *A copy of a structure* $\mathfrak{M}$ *in an admissible set* $\mathbb{A}$ *is a structure* $\mathfrak{N}$ *such that* $\mathfrak{N} \simeq \mathfrak{M}$ *and* $N \subseteq A$.

The following theorem characterizes so called relatively intrinsically c.e. relations of countable structures in terms of definability in hereditary finite superstructures.

Theorem 6 (Ash, Knight, Manasse, Slaman [3], Chisholm [4]). *Let* $\mathfrak{M}$ *be a countable structure and let* $P \subseteq M^n$. *Then the following are equivalent:*

- P *is* Σ*-definable in* $\mathbb{HF}(\mathfrak{M})$;
- *for any copy* $\mathfrak{N}$ *of* $\mathfrak{M}$ *in* $\mathbb{HF}(\mathfrak{M})$ *and any isomorphism* f *from* $\mathfrak{M}$ *onto* $\mathfrak{N}$, $f(P)$ *is relatively c.e.;*
- *for any copy* $\mathfrak{N}$ *of* $\mathfrak{M}$ *in* $\mathbb{HF}(\varnothing)$ *and any isomorphism* f *from* $\mathfrak{M}$ *onto* $\mathfrak{N}$, $f(P)$ *is relatively c.e..*

However, for (absolutely) intrinsically c.e. relations there is no such syntactical criteria. We recall

Theorem 7 (Goncharov [5], Manasse [6]). *There exists a countable structure* $\mathfrak{M}$ *with a computable copy in* $\mathbb{HF}(\varnothing)$ *and* $P \subseteq M$ *such that for any computable copy* $\mathfrak{N}$ *of* $\mathfrak{M}$ *in* $\mathbb{HF}(\varnothing)$ *and any isomorphism* f *from* $\mathfrak{M}$ *onto* $\mathfrak{N}$, $f(P)$ *is c.e., but* P *is not* Σ*-definable in* $\mathbb{HF}(\mathfrak{M})$.

Now we consider the notion of intrinsically decidable structure.

Theorem 8. *Let* $\mathfrak{M}$ *be a countable structure,* $n \leqslant \omega$. *Then the following are equivalent:*

- $\mathfrak{M}$ *is* n*-decidable in* $\mathbb{HF}(\mathfrak{M})$;
- *any copy of* $\mathfrak{M}$ *in* $\mathbb{HF}(\mathfrak{M})$ *is relatively* n*-decidable;*
- *any copy of* $\mathfrak{M}$ *in* $\mathbb{HF}(\varnothing)$ *is relatively* n*-decidable.*

Theorem 9 (Nurtazin [7]). *Let* $\mathfrak{M}$ *be a countable structure with computable copy in* $\mathbb{HF}(\varnothing)$, $n \leqslant \omega$. *Then the following are equivalent:*

- $\mathfrak{M}$ *is* n*-decidable in* $\mathbb{HF}(\mathfrak{M})$;
- *any copy of* $\mathfrak{M}$ *in* $\mathbb{HF}(\varnothing)$ *is relatively* n*-decidable;*
- *any computable copy of* $\mathfrak{M}$ *in* $\mathbb{HF}(\varnothing)$ *is* n*-decidable.*

The previous theorem shows that in case of decidability it is impossible to construct an analog of the Goncharov-Manasse example from Theorem 7. About the existence of relatively decidable copies we recall

Theorem 10 (Harizanov, Knight, Morozov [8]). *Let* $\mathfrak{M}$ *be a countable structure. Then in* $\mathbb{HF}(\varnothing)$ *there exists a relatively decidable copy of* $\mathfrak{M}$.

We prove the following

Theorem 11. *Let* $\mathfrak{M}$ *be a structure (of computable signature). Then in* $\mathbb{HF}(\mathfrak{M})$ *there exists a relatively decidable copy of* $\mathfrak{M}$.

Suppose that $\mathfrak{M}$ is an arbitrary (possibly uncountable) structure of computable signature and $\mathcal{S}$ be a structure of empty signature of the same cardinality as $\mathfrak{M}$.

Conjecture 12. There exists a relatively decidable copy of $\mathfrak{M}$ in $\mathbb{HF}(\mathcal{S})$.

Conjecture 13. For any $n \leqslant \omega$ the following are equivalent:
- $\mathfrak{M}$ is n-decidable in $\mathbb{HF}(\mathfrak{M})$;
- any copy of $\mathfrak{M}$ in $\mathbb{HF}(\mathfrak{M})$ is relatively n-decidable;
- any copy of $\mathfrak{M}$ in $\mathbb{HF}(\mathcal{S})$ is relatively n-decidable.

Conjecture 14. Suppose $\mathfrak{M}$ is a structure with computable copy in $\mathbb{HF}(\mathcal{S})$, $n \leqslant \omega$. Then the following are equivalent:
- $\mathfrak{M}$ is n-decidable in $\mathbb{HF}(\mathfrak{M})$;
- any copy of $\mathfrak{M}$ in $\mathbb{HF}(\mathcal{S})$ is relatively n-decidable;
- any computable copy of $\mathfrak{M}$ in $\mathbb{HF}(\mathcal{S})$ is n-decidable.

We now give some examples of structures decidable in hereditary finite admissible sets and demonstrate some connections of decidability with various computable properties of such sets.

A theory T is *regular* [1] if it is model complete and decidable.

Proposition 15. *If* $\mathrm{Th}(\mathfrak{M})$ *is regular then* $\mathfrak{M}$ *is decidable in* $\mathbb{HF}(\mathfrak{M})$.

Example 16. $\mathbb{R}$, $\mathbb{Q}_p$, $\mathbb{C}$ are structures with regular elementary theories.

We describe decidable linear orders in the following way:

Theorem 17. *A linear order* $\mathfrak{L}$ *is 1-decidable in* $\mathbb{HF}(\mathfrak{L})$ *iff* $\mathfrak{L}$ *is a sum of a finite number of dense linear orders and points.*

A structure $\mathfrak{M}$ is *n-complete* [5] ($n \leqslant \omega$) if for any formula $\varphi(\bar{x}) \in F_n(\sigma)$ and for any $\bar{m} \in M^{<\omega}$ s.t. $\mathfrak{M} \models \varphi(\bar{m})$ there exists a $\exists$-formula $\psi(\bar{x})$ such that $\mathfrak{M} \models \psi(\bar{m})$ and $\mathfrak{M} \models \forall \bar{x}(\psi(\bar{x}) \to \varphi(\bar{x}))$.

Proposition 18. *Suppose* $\mathfrak{M}$ *is n-decidable in* $\mathbb{HF}(\mathfrak{M})$ *($n \leqslant \omega$). Then* $\mathfrak{M}$ *is n-complete in some finite constant expansion.*

Proposition 19. *Suppose* $\mathfrak{M}$ *is n-complete and* $\mathrm{Th}(\mathfrak{M})$ *is decidable. Then* $\mathfrak{M}$ *is n-decidable in* $\mathbb{HF}(\mathfrak{M})$.

Suppose $\mathfrak{M}$ is 1-decidable in $\mathbb{HF}(\mathfrak{M})$. Then $\mathbb{HF}(\mathfrak{M})$ has a universal Σ-function and the reduction property, but not necessarily the uniformization property.

Let $\mathfrak{M}$ be a structure of signature σ and let signature σ_* consists of all symbols of σ and new functional symbols $f_\varphi(x_1, \ldots, x_n)$ for all existential formulas $\varphi(x_0, x_1, \ldots, x_n)$ of signature σ. The structure $\mathfrak{M}_*$ of signature σ_* is called *existential Skolem expansion of* $\mathfrak{M}$ if $M_* = M$, $\mathfrak{M} \restriction_\sigma = \mathfrak{M}_* \restriction_\sigma$ and for any existential formula $\varphi(x_0, x_1, \ldots, x_n)$ of signature σ

$$\mathfrak{M}_* \models \forall x_1 \ldots \forall x_n(\exists x \varphi(x, x_1, \ldots, x_n) \to \varphi(f_\varphi(x_1, \ldots, x_n), x_1, \ldots, x_n)).$$

The next theorem is a generalization of the main result from [13].

Theorem 20. *Suppose $\mathfrak{M}$ is 1-decidable in $\mathbb{HF}(\mathfrak{M})$. Then $\mathbb{HF}(\mathfrak{M})$ has the uniformization property iff some existential Skolem expansion of $\mathfrak{M}$ is computable in $\mathbb{HF}(\mathfrak{M})$.*

Theorem 21. *For any $n \in \omega$ there exists ω-categorical structure $\mathfrak{M}$ such that $\mathfrak{M}$ is n-decidable in $\mathbb{HF}(\mathfrak{M})$ but not $(n+1)$-decidable $\mathbb{HF}(\mathfrak{M})$. There also exists ω-categorical structure $\mathfrak{M}$ such that for any $n \in \omega$ $\mathfrak{M}$ is n-decidable in $\mathbb{HF}(\mathfrak{M})$ but $\mathfrak{M}$ is not decidable in $\mathbb{HF}(\mathfrak{M})$.*

An admissible set $\mathbb{A}$ is *quasiresolvable* [1] if there exists a sequence $B_0 \subseteq B_1 \subseteq \ldots \subseteq B_\alpha \subseteq \ldots,\ \alpha \in Ord\,\mathbb{A}$ of transitive subsets of A such that $\cup_{\alpha \in Ord\,\mathbb{A}} B_\alpha = A$, and the subsets $\{\langle \alpha, a\rangle | a \in B_\alpha\}$ and

$$\{\langle \alpha, \ulcorner \Phi \urcorner, \bar{a}\rangle \mid\ \alpha \in Ord\,\mathbb{A}, \Phi(\bar{x}) \in F(\sigma_{\mathbb{A}}), \bar{a} \in B_\alpha^{<\omega},\ \mathbb{A} \restriction B_\alpha \models \Phi(\bar{a})\}$$

are Δ-definable in $\mathbb{A}$.

An admissible set $\mathbb{A}$ is *1-quasiresolvable* if there exists a sequence $B_0 \subseteq B_1 \subseteq \ldots \subseteq B_\alpha \subseteq \ldots,\ \alpha \in Ord\,\mathbb{A}$ of transitive subsets of A such that $\cup_{\alpha \in Ord\,\mathbb{A}} B_\alpha = A$, and the subsets $\{\langle \alpha, a\rangle | a \in B_\alpha\}$ and

$$\{\langle \alpha, \ulcorner \Phi \urcorner, \bar{a}\rangle \mid\ \alpha \in Ord\,\mathbb{A}, \Phi(\bar{x})\ -\ \Pi\text{-formula of } \sigma_{\mathbb{A}}, \bar{a} \in B_\alpha^{<\omega},\ \mathbb{A} \restriction B_\alpha \models \Phi(\bar{a})\}$$

are Δ-definable in $\mathbb{A}$.

If admissible set $\mathbb{A}$ is 1-quasiresolvable then $\mathbb{A}$ has a universal Σ-function and the reduction property [1]. If $\mathfrak{M}$ is (1-)decidable in $\mathbb{HF}(\mathfrak{M})$ then $\mathbb{HF}(\mathfrak{M})$ is (1-)quasiresolvable. The converse is not true in general.

Theorem 22. *Suppose $\mathfrak{M}$ is an ω-categorical. Then*

1) $\mathfrak{M}$ is decidable in $\mathbb{HF}(\mathfrak{M})$ iff $\mathbb{HF}(\mathfrak{M})$ is quasiresolvable;
2) $\mathfrak{M}$ is 1-decidable in $\mathbb{HF}(\mathfrak{M})$ iff $\mathbb{HF}(\mathfrak{M})$ is 1-quasiresolvable.

2 Constructivizations of Structures in Admissible Sets

Let $\mathfrak{N}$ be a structure of relational computable signature $\langle P_0^{n_0}, \ldots, P_k^{n_k}, \ldots\rangle$ and let $\mathbb{A}$ be an admissible set.

Definition 23 (Ershov [1]). *$\mathfrak{N}$ is Σ-definable (constructivizable) in $\mathbb{A}$ if there exists a computable sequence of Σ-formulas*

$$\Phi(x_0, y), \Psi(x_0, x_1, y), \Psi^*(x_0, x_1, y), \Phi_0(x_0, \ldots, x_{n_0-1}, y),$$

$$\Phi_0^*(x_0, \ldots, x_{n_0-1}, y), \ldots, \Phi_k(x_0, \ldots, x_{n_k-1}, y), \Phi_k^*(x_0, \ldots, x_{n_k-1}, y), \ldots$$

such that for some parameter $a \in A$, and letting

$$N_0 \leftrightharpoons \Phi^{\mathbb{A}}(x_0, a),\ \ \eta \leftrightharpoons \Psi^{\mathbb{A}}(x_0, x_1, a) \cap N_0^2$$

one has that $N_0 \neq \varnothing$ *and* η *is a congruence relation on the structure*

$$\mathfrak{N}_0 \leftrightharpoons \langle N_0, P_0^{\mathfrak{N}_0}, \ldots, P_k^{\mathfrak{N}_0}, \ldots \rangle,$$

where $P_k^{\mathfrak{N}_0} \leftrightharpoons \Phi_k^{\mathbb{A}}(x_0, \ldots, x_{n_k-1}) \cap N_0^{n_k}$, $k \in \omega$,

$$\Psi^{*\mathbb{A}}(x_0, x_1, a) \cap N_0^2 = N_0^2 \setminus \Psi^{\mathbb{A}}(x_0, x_1, a),$$
$$\Phi_k^{*\mathbb{A}}(x_0, \ldots, x_{n_k-1}, a) \cap N_0^{n_k} = N_0^{n_k} \setminus \Phi_k^{\mathbb{A}}(x_0, \ldots, x_{n_k-1})$$

for all $k \in \omega$ *and the structure* $\mathfrak{N}$ *is isomorphic to the quotient structure* $\mathfrak{N}_0/\eta$.

Definition 24. *A theory* T *is c-simple [1] if it is* ω*-categorical, model complete, decidable and has decidable set of complete formulas.*

It is well known that c-simple theories have very nice computable properties. For example, any two decidable (in classical sense) models of a c-simple theory are computably isomorphic. One of the most important examples of c-simple theories is the theory of dense linear orders (without endpoints).

Conjecture 25 (Ershov [9]). If T is c-simple theory then some uncountable model of T is constructivizable in $\mathbb{HF}(\mathfrak{L})$ for some (uncountable) dense linear order $\mathfrak{L}$.

In connection with this conjecture we mention the following result about ω-categoricity and orderings.

Theorem 26 (Schmerl [10]). *If* $\mathfrak{A}$ *is countable infinite* ω*-categorical structure then there is a linear order* $<$ *of* A *with order type of the rationals such that* $\langle \mathfrak{A}, < \rangle$ *is* ω*-categorical.*

The next definition is a generalization of the well-known notions of order indiscernibility and total indiscernibility from model theory.

Definition 27 ([14]). *For arbitrary structures* $\mathfrak{A}$ *and* $\mathfrak{B}$ *a set* $I \subseteq A \cap B$ *is called a set of* $\mathfrak{A}$*-indiscernibles in* $\mathfrak{B}$ *if for any tuples* $\bar{i}, \bar{i}' \in I^{<\omega}$ *of the same length*

$$\langle \mathfrak{A}, \bar{i} \rangle \equiv \langle \mathfrak{A}, \bar{i}' \rangle \text{ implies } \langle \mathfrak{B}, \bar{i} \rangle \equiv \langle \mathfrak{B}, \bar{i}' \rangle.$$

Let T and T' be c-simple theories. If some uncountable model of T' is constructivizable in the HF-superstructure over some model of T, then there are decidable models $\mathfrak{A}$ and $\mathfrak{B}$ of T and T' respectively such that there is an infinite computable set of $\mathfrak{A}^*$-indiscernibles in $\mathfrak{B}$, where $\mathfrak{B}^*$ is an expansion of $\mathfrak{B}$ by finite number of constants.

For some c-simple theories this necessary condition of constructivizability is also sufficient. We denote by T_{DLO} the theory of dense linear orders and by T_E the theory of infinite models of equality.

Theorem 28 ([14]). *Let* T *be c-simple theory and* $\mathfrak{A}$ *be any decidable model of* T. *Then*

1) T has an uncountable model which is constructivizable in $\mathbb{HF}(\mathfrak{L})$ for some $\mathfrak{L} \models T_{DLO}$ if and only if there exists an infinite computable set of order indiscernibles in $\mathfrak{A}$;
2) T has an uncountable model which is constructivizable in $\mathbb{HF}(\mathcal{S})$ for some $\mathcal{S} \models T_E$ if and only if there exists an infinite computable set of total indiscernibles in $\mathfrak{A}$.

Theorem 29 (Kierstead, Remmel [11]). *There exists a c-simple theory T s.t. any infinite set of order indiscernibles in any decidable model of T is not computable.*

By using this result we obtain a counterexample for Ershov conjecture.

Corollary 30 ([14]). *There exists a c-simple theory (of infinite signature) such that none of its uncountable models is constructivizable in $\mathbb{HF}(\mathfrak{L})$, where $\mathfrak{L}$ is a dense linear order.*

Conjecture 31. For any c-simple theory T there exists a c-simple theory T' such that for any uncountable $\mathfrak{M} \models T$ and $\mathfrak{M}' \models T'$ $\mathfrak{M}'$ is not constructivizable in $\mathbb{HF}(\mathfrak{M})$.

3 Inner Constructivizability of Admissible Sets

Consider a signature σ and let P be unary predicate symbol not in σ. For a QR-formula (i.e. formula which possibly contain restricted quantifiers of the form $\forall x \in y$ and $\exists x \in y$) Φ of signature $\sigma \cup \{\in\}$ we define inductively the *relativization* Φ^P of the formula Φ by the predicate P as follows:
– if Φ is atomic then $\Phi^P = \Phi$;
– if $\Phi = (\Phi_1 * \Phi_2)$, $* \in \{\wedge, \vee, \to\}$ then $\Phi^P = (\Phi_1^P * \Phi_2^P)$;
– if $\Phi = \neg\Psi$ then $\Phi^P = \neg\Phi^P$;
– if $\Phi = (Qx \in y)\Psi$, $Q \in \{\forall, \exists\}$ then $\Psi^P = (Qx \in y)\Psi^P$;
– if $\Phi = \exists x\Psi$ then $\Phi^P = \exists x(P(x) \wedge \Psi^P)$;
– if $\Phi = \forall x\Psi$ then $\Phi^P = \forall x(P(x) \to \Psi^P)$.

In case $\mathbb{A}$ is an admissible set, $B \subseteq A$ and $\Phi(x_0, \ldots, x_{n-1})$ is a QR-formula of signature $\sigma_{\mathbb{A}}$, we define

$$(\Phi(x_0, \ldots, x_{n-1}))^B = \{\langle a_0, \ldots, a_{n-1}\rangle \in A^n \mid \langle \mathbb{A}, B\rangle \models \Phi^P(a_0, \ldots, a_{n-1})\}.$$

Definition 32. *A structure $\mathfrak{M}$ of computable predicate signature $\langle P_0^{n_0}, P_1^{n_1}, \ldots\rangle$ is constructivizable in an admissible set $\mathbb{A}$ inside $B \subseteq A$ if there exists computable sequence of formulas*

$$\Phi(x_0, y), \Psi(x_0, x_1, y), \Psi^*(x_0, x_1, y), \Phi_0(x_0, \ldots, x_{n_0-1}, y),$$

$$\Phi_0^*(x_0, \ldots, x_{n_0-1}, y), \ldots, \Phi_k(x_0, \ldots, x_{n_k-1}, y), \Phi_k^*(x_0, \ldots, x_{n_k-1}, y), \ldots$$

and $b \in B$ such that, letting

$$M_0 \leftrightharpoons \Phi^B(x_0, b),\ M_0 \subseteq B,\ \ \eta \leftrightharpoons \Psi^B(x_0, x_1, b) \cap M_0^2$$

one has that $M_0 \neq \varnothing$ and η is a congruence relation on the structure

$$\mathfrak{M}_0 \leftrightharpoons \langle M_0, P_0^{\mathfrak{M}_0}, \ldots, P_k^{\mathfrak{M}_0}, \ldots \rangle,$$

where $P_k^{\mathfrak{M}_0} \leftrightharpoons (\Phi_k(x_0, \ldots, x_{n_k-1}))^B \cap M_0^{n_k},\ k \in \omega,$

$$(\Psi^*(x_0, x_1, a))^B \cap M_0^2 = M_0^2 \setminus (\Psi(x_0, x_1, a))^B,$$

$$(\Phi_k^*(x_0, \ldots, x_{n_k-1}, a))^B \cap M_0^{n_k} = M_0^{n_k} \setminus (\Phi_k(x_0, \ldots, x_{n_k-1}))^B$$

for all $k \in \omega$, and $\mathfrak{M}$ is isomorphic to $\mathfrak{M}_0/\eta$.

If $\mathbb{A}$ is an admissible set then for arbitrary $B \subseteq A$ we define $\mathrm{rnk}(B)$ in the usual way:

$$\mathrm{rnk}(B) = \sup\{\mathrm{rnk}(b) | b \in B\}.$$

Definition 33. *The rank of inner constructivizability of an admissible set $\mathbb{A}$ is the ordinal*

$$\mathrm{cr}(\mathbb{A}) = \inf\{\mathrm{rnk}(B) \mid \mathbb{A} \textit{ is constructivizable in } \mathbb{A} \textit{ inside } B\}.$$

The next theorem gives the precise estimates of the rank of inner constructivizability for hereditary finite superstructures.

Theorem 34 ([15]). *Suppose $\mathfrak{M}$ is a structure of computable signature. Then*
1) if $\mathfrak{M}$ is finite then $\mathrm{cr}(\mathbb{HF}(\mathfrak{M})) = \omega$,
2) if $\mathfrak{M}$ is infinite then $\mathrm{cr}(\mathbb{HF}(\mathfrak{M})) \leqslant 2$.

From this theorem we obtain effective analogs of some results from [12] about definability in multisorted languages.

Examples of structures $\mathfrak{M}$ for which $\mathrm{cr}(\mathbb{HF}(\mathfrak{M})) = 2$ are infinite models of empty signature, dense linear orders, and, more interesting, the structure $\langle \omega, s \rangle$ of natural numbers with successor function. Indeed, if we denote by $\mathrm{Th}_{\mathrm{WM}}(\mathfrak{M})$ the theory of $\mathfrak{M}$ in the language of weak monadic second order logic, then the following lemma is true.

Lemma 35. *If $\mathrm{Th}_{\mathrm{WM}}(\mathfrak{M})$ is decidable then $\mathrm{cr}(\mathbb{HF}(\mathfrak{M})) = 2$.*

From Büchi result about decidability of $\mathrm{Th}_{\mathrm{WM}}(\langle \omega, s \rangle)$ and the previous lemma we get that

$$\mathrm{cr}(\mathbb{HF}(\langle \omega, s \rangle)) = 2.$$

An example of structure $\mathfrak{M}$ for which $\mathrm{cr}(\mathbb{HF}(\mathfrak{M})) = 0$ is, obviously, the standard model of arithmetic $\mathbb{N}$. An example of structure for which rank of inner constructivizability is equal to 1 is the field $\mathbb{R}$ of real numbers.

Theorem 36 ([15]).

$$\mathrm{cr}(\mathbb{HF}(\mathbb{R})) = 1.$$

References

1. Ershov Yu.L.: Definability and Computability. Plenum. New York. (1996)
2. Barwise J.: Admissible Sets and Strucrures. Springer-Velag. Berlin. (1975)
3. Ash C., Knight J., Manasse M., Slaman T.: Generic copies of countable structures. Ann. Pure. Appl. Logic. **42** (1989) 195 – 205
4. Chisholm J.: Effective model theory vs. recursive model theory. J.Symbolic Logic. **55** (1990) 1168 – 1191
5. Goncharov S.S.: The quantity of nonautoequivalent constructivizations. Algebra and Logic. **16** (1977) 169 – 185
6. Mannasse M.: Techniques and counterexamples in almost categorical computable model theory. Ph.D. Thesis. University of Wisconsin. Madison. WI 1982
7. Nurtazin A.T.: Strong and weak constructivizations and computable families. Algebra and Logic. **13** (1974) 177 – 184
8. Harizanov V., Knight J., Morozov A.: Sequences of n-diagrams. J.Symbolic Logic. **67** (2002) 1227 – 1248
9. Ershov Yu.L.: Σ-definability of algebraic structures. Handbook of Recursive Mathematics. **1** (1998) 235 – 260
10. Schmerl J.H.: Decidability and $\aleph_0$-categoricity of theories of partially ordered sets. J. Symbolic Logic. **45** (1980) 585 – 611
11. Kierstead H.A., Remmel J.B.: Degrees of indiscernibles in decidable models. TAMS. **289** (1985) 41 – 57
12. Montague R.: Recursion theory as a branch of model theory. Proceedings of the Third International Congress for Logic, Methodology and Philosophy of Science, Amsterdam. (1967) 63 – 86
13. Stukachev A.I.: Uniformization property in heredidary finite superstructures. Siberian Advances in Mathematics. **7** (1997) 123 – 132
14. Stukachev A.I.: Σ-definability in hereditary finite superstructures and pairs of models. Algebra and Logic. **46** (2004) 258 – 270
15. Stukachev A.I.: On inner constructivizability of admissible sets. Vestnik NGU (to appear)

An Environment Aware P-System Model of Quorum Sensing

German Terrazas[1], Natalio Krasnogor[1], Marian Gheorghe[2], Francesco Bernardini[2], Steve Diggle[3], and Miguel Cámara[3]

[1] ASAP Group, School of Computer Science and IT,University of Nottingham, UK
gzt@cs.nott.ac.uk
Natalio.Krasnogor@Nottingham.ac.uk
[2] Department of Computer Science, University of Sheffield, UK
{M.Gheorghe, F.Bernardini}@dcs.shef.ac.uk
[3] Molecular Medical Sciences, Institute of Infection, Immunity and Inflamation, University of Nottingham, UK
{Steve.Diggle, Miguel.Camara}@Nottingham.ac.uk

Abstract. "Quorum Sensing" has been identified as one of the most consequential microbiology discoveries of the last 10 years. Using Quorum Sensing bacterial colonies synchronize gene expression and phenotype change allowing them, among other things, to protect their niche, coordinate host invasion and bio-film formation. In this contribution we briefly describe the elementary microbiology background and present a P-systems based model for Quorum Sensing which includes environmental rules and a topological representation.

1 Introduction

Recent advances in analytical biotechnology, computational biology, bioinformatics and computational modeling promise ever deeper understanding of the complexity of biological systems, particularly the computations they perform in order to survive in dynamic and hostile environments. These insights will ultimately enable researchers to harness the living cell as a computational device with its own sensors, internal states, transition functions, actuators, etc, and to program them as "nano-bots" for particular tasks such as targeted drug delivery, chemical factories, nano-structures repairs, bio-film scaffolding and self-assembling, to name but a few.

In this paper we will focus on one of the most important mechanisms for bacterial cell-to-cell communication and behavior coordination under changing environments: "quorum sensing (QS)". QS have been described as "the most consequential molecular microbiology story of the last decade" [21, 3]. It relies on the activation of a sensor kinase or response regulator protein by a diffusible, low molecular weight, signal molecule (a "pheromone" or "autoinducer") [12]. In QS, the concentration of the signal molecule reflects the number of bacterial cells in a particular niche and perception of a threshold concentration of that

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 479–485, 2005.

signal molecule indicates that the population is "quorated", i.e. ready to make a behavioral decision [20].

Natural QS is a powerful computational mechanism[5] that endows *Pseudomonas aeruginosa* with the capabilities to coordinate a population-wide attack necessary to breach host's immunological defences. Other bacteria (both Gram-negative and Gram-positive), like *V. fischeri*, *A. tumefaciens*, *E. carotovora*, *V. harveyi*, *B. subtilis*, *S. aureus*, *S. pneumoniae*, etc., also use QS for different purposes and it is usually mediated by a variety of sensors/receptors, regulons, etc.

In this paper we will present an overview of Quorum Sensing in *P. aeruginosa* and we will also show a more "computationally flavored" approach for Quorum Sensing which is based on a modified version of P-systems that takes into consideration topological aspects of the environment where cells live.

2 Cell-to-Cell Communication by Means of Quorum Sensing

The QS mechanism is a communication strategy based on diffusible signals, S, which kick-in under high cellular density. Bacteria use this mechanism to obtain a population-wide coordination of infection, invasion, and evasion of a host's defenses.

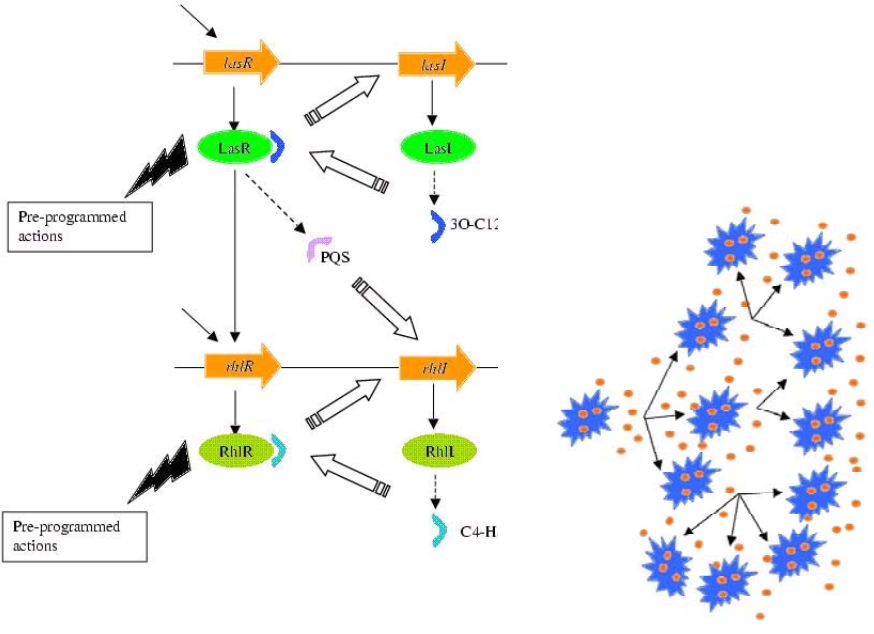
Once the mechanism is activated it usually triggers a cascade of transcriptional activity which results in phenotypic changes that are frequently related to the activation of virulence encoded regulons. As we mentioned before both Gram-negative and Gram-positive bacteria employ similar coordination mechanism albeit with different messenger molecules. The messenger molecules are often called (auto)inducers or pheromones(to be denoted by S). Under low bacterial densities molecule S is synthesized and accumulate. According to the specific geometry of the inducer molecule, more precisely its length, the synthesized S are either pumped out of the cell or simply diffuse into the surrounding environment[1].

Once in the environment, the inducer molecules that are usually much smaller than small proteins (and certainly tiny compared to the bacterium itself), disperse quickly and sometimes get in contact with other individual bacteria who occasionally ends up absorbing the inducer molecule. In addition to the inducer molecule, bacteria also produce a receptor molecule R. At high inducer's concentrations (within the cellular membranes) and once a specific threshold concentration is reached, the receptor molecules R binds to the inducers S forming a molecular complex. In turn the pheromone bound version of R, $R \circ S$, binds to a specific chromosome region thus activating or repressing the transcription of certain genes. Moreover, as the gene encoding the syntethase I for the inducer S (denoted with *i*, i.e. it is represented with the same letter as the synthetase, I, but in italics) is positively regulated by the complex $R \circ S$, a rapid signal ampli-

[1] In Gram-positive bacteria S is always actively transported out of the cellular membrane.

fication, i.e. positive feedback and hence the name autoinducer, takes place. The transcription of i into I results in the synthesis of excess molecules S that diffuse out of the bacteria and into the local environment. It is important to note that although it is possible to speak of "diffusion" in the case of Gram-negative, in Gram-positives the signal molecules don't diffuse out of the cell, instead they are secreted using active transport systems which in some cases activate the appropriate signals as they are getting out (e.g.*Staphylococcus aureus*). Also in Gram-positives the active molecule does not get into the cell; instead it activates cellular surface receptors which in turns relays the activation to other proteins resulting in the transcriptional response.

The amplification loop is shown in Figure 1(a). Under high cellular density, and once the QS is activated, the positive feedback effectively triggers a chain reaction that bridges the gap between various physical scales. That is, Quorum Sensing is a mechanism which processes and integrates information that ranges from the nano-level of the cell interior to the macro-level of a bacterial colony (sometimes visible with the naked eye) in a short period of time. Figure 1(b) gives a graphical representation for this phenomenom.



(a) (b)

Fig. 1. (a) Overlapped Quorum Sensing systems in *P. aeruginosa*s. (b) Multi-scale effects mediated by Quorum Sensing. A cell (from among a group of cells) senses an increase in inducer molecules (*small dots*) in the surrounding environment. The internal feedback loop is activated and, in turn, deposits more inducers into the external medium. The increase in inducers concentration in turn triggers other cells to react leading to a chain reaction

The chain reaction, mediated by the high mobility of S in the environment, ensures that more and more individual bacteria are activated within a short period of time producing a population wide behavioral shift. This behavioral shift is possible because QS activates the coordinated transcription of multiple genes. Consequently, in QS, the concentration of the signal molecule reflects the number of bacterial cells in a particular niche and perception of a threshold concentration of that signal molecule indicates that the population is "quorated", i.e. ready to make a behavioral decision. Wagner et al. [17] report that in *P. aeruginosa* up to 394 genes are activated by QS while 222 are repressed. In a recent review [7] it is estimated that between 6% to 10% of the whole genome is affected by this mechanism. Some of the phenotypic changes actuated by QS are increase in virulence, changes in the production of secondary metabolites, conjugation, growth inhibition, motility, swarming and bio-film formation. The reader must note that the autoinducers mentioned before are mainly used for bacterial intraspecies communication. A newly reported autoinducer called AI-2 has been proposed as a potential universal signals that mediate interspecies communication.

Important in governing the size of the "quorum" is 'compartment sensing' [21]. As noted above, the concentration of a given QS signal molecule may be a reflection of bacterial cell number, or at least the minimal number of cells (quorum) in a particular physiological state. To achieve the accumulation of a QS signal there is a need for a diffusion barrier, which ensures that more molecules are produced than lost from a given microhabitat. This 'compartment sensing' enables the QS signal molecule to be both a measure of the degree of compartmentalization and the means to distribute this information through the entire population. Likewise, the diffusion of QS signal molecules between detached sub-populations may convey information about their numbers, physiological state and the specific environmental conditions encountered. QS is thus a natural efficient, robust and simple mechanism for cell-to-cell communication.

3 An Environment-Aware P-System for Quorum Sensing

In this section we present an environment-aware P-system to simulate the process which occur in bacterial colonies which are capable of quorum sensing communications.

An environment-aware P-system Ω is defined as a collection of "environments", which contain both cells and metabolites, and communication channels between the environments. Both the environments and the channels are limited in their capacity of metabolite storage and transmission respectively. Formally: $\Omega = (\Pi_1, \ldots, \Pi_n, \tau_1, \ldots, \tau_n, \Gamma_1, \ldots, \Gamma_n, \Theta_1, \ldots, \Theta_n)$ where:

1. Π_i is an environment defined as $\Pi_i = (V, w_{E_i}, R_{E_i}, C_{E_{i_1}}, \ldots, C_{E_{i_n}})$
2. τ_i is the maximum amount of metabolites that Π_i can contain. The limit could arise for example from diffusion rate constraints. The metabolites are represented by objects in w_{E_i}.

3. $\Gamma_i = (\Pi_o, \Pi_t) 1 \leq o, t \leq n$ is a transmission channel between 2 environments.
4. Θ_i is the "bandwidth" of channel Γ_i.

An environment Π_i has:

1. V is a finite alphabet of symbols which represent secreted "metabolites" (e.g. signaling molecules or mRNA molecules).
2. $w_{E_i} \in V^*$ is a finite multiset of metabolites initially assigned to it.
3. R_i is a finite set of transformation rules associated with the environment. These rules can be of the form:
   - synthesis rules, $a \rightarrow y$, for $a \in V$, and $y \in V^*$
   - carriers construction rules (see [13]), $v_i m_1, \ldots, m_p \rightarrow [v_i m_1, \ldots, m_p]$ for $v_i \in V$, and $m_i \in V$.
   - carriers deconstruction rules, $[v_i m_1, \ldots, m_p] \rightarrow v_i m_1, \ldots, m_p$ for $v_i \in V$, and $m_i \in V$.
4. $C_i = (w_i, S_i, R_i, >)$, for each $1 \leq i \leq m_i$, a cell with:
   (a) $w_i \in V^*$ is a finite multiset of metabolites internal to cell C_i;
   (b) S_i is a finite set of communication rules; each rule has the form $(x; y, enter)$, where $x, y \in V^*$. These rules are used by the cell C_i to receive objects y from the environment when x is present in the cell.
   (c) R_i is a finite set of transformation-communication rules of the form $b_1 \ldots b_r \rightarrow (a_1)_{t_1} \ldots (a_q)_{t_q}$, for $b_i \in V$, and $1 \leq i \leq r$, $a_i \in V$, and $t_i \in \{here, out\}$, $1 \leq i \leq q$;
   (d) $>$ is a partial order on R_i. These rules are used by a cell to consume a multiset $b_1 \ldots b_r$ in order to produce a new multiset $a_1 \ldots a_q$ of which those with $t_j = here$ remain inside of the cell C_i and those with $t_j = out$ go out in the environment. A rule r_1 from R_i is used in a step if there is no rule r_2 in R_i which can be applied at the same step and $r_2 > r_1$.

In turn, each environment Π_i has a set of neighboring (i.e. overlapping regions) environments. This neighborhood set is involved in the "channel rules":

$N(\Pi_i) = \{\Pi_j | \exists \Gamma = (\Pi_i, \Pi_j)\ or\ \Gamma = (\Pi_j, \Pi_i)\}$. The channel rule is composed of three steps:

- $\Pi_i \circ [v_i m_1, \ldots, m_p] \overrightarrow{\Gamma_i} N(\Pi_i)$, for $v_i \in V$, and $m_i \in V$
- $\Pi_j = (V, w_{E_j} + m_1 + \ldots + m_p, R_j, C_{E_{j_1}}, \ldots, C_{E_{j_n}})$.
- $\Pi_i = (V, w_{E_i} - m_1 - \ldots - m_p, R_i, C_{E_{i_1}}, \ldots, C_{E_{i_n}})$

These rules state that a channel $\Gamma_i = (\Pi_i, \Pi_j)$ will be able to transfer already formed carriers (e.g. $[v_i m_1, \ldots, m_p]$) from, let say, Π_i to Π_j if the capacity of the Γ_i channel , Θ_i, is large enough to contain the p metabolites in the carriers and if the available storage in the target environment is enough to contain the additional metabolites once the carrier is unbuild. The target environment Π_j is non-deterministically chosen from $N(\Pi_i)$. After the movement of the carrier from one of the environments to the other, the appropriate number of metabolites

are substracted and added in each one[2]. Please note that this is just one rule composed of various steps not three independent rules, as such it should be considered atomic.

In general, biological Quorum Sensing once switched on it is never turned off but rather fine tuned and regulated by other concurrent processes within the cells. As such, we will not consider here halting computations but rather non-halting processes, we thus refrain from specifying an output membrane. This simple P-system model can mimic the basic behavior of a "quorated" system.

4 Conclusions and Future Research

A deeper understanding of biological Quorum Sensing could have an important impact not only in the biological sciences but also in computer science applications. Quorum sensing is a mechanism that, although complex in its biological details, is simple in its fundamental principles. Its appeal resides in the fact that, by exploiting a simple feedback loop and the limited capacity of both the cell and the environment to diffuse and carry a signal molecule, it is possible to bridge the "scale gap" between the individual bacterium and the colony. Such a mechanism should be useful in many applications beyond biological ones where multiple agents needs to robustly and efficiently coordinate their collective behavior based only on very limited information of the local environment. We are actively following several lines of research on both biological Quorum Sensing per se, modeling techniques based on P-systems, and we are also considering a range of computational applications. In particular we will investigate in the future the computational power of environment-aware P-systems and we will extend them to allow cell migration through the channels and channel/environment creation and removal. All of these will be reported elsewhere.

References

1. E. Ben Jacob. Bacterial Wisdom, Gödel's Theorem and Creative Genomic Webs. *Physica A*, 48:57-76, 1998.
2. F. Bernardini and M. Gheorghe. Population P Systems. *J. Univ. Comp. Sci.*, 10:509-539, 2004.
3. S. Busby and V. de Lorenzo. Cell regulation - putting together pieces of the big puzzle. *Curr. Op. Microbiol*, 4:117-118, 2001.
4. J.D. Dockery and J.P. Keener. A Mathematical Model for Quorum Sensing in Pseudomonas aeruginosa. *Bulletin of Mathematical Biology*, 18:95-116, 2001.
5. N. Krasnogor, M. Gheorghe, G. Terrazas, S. Diggle, P. Williams and M. Camara. An Appealing Computational Mechanism Drawn From Bacterial Quorum Sensing. *Bulleting of the European Association of Theoretical Computer Science*, in press.

[2] The '+' operator indicates the addition of an object to the multiset w_{E_j} and similarly with '-'.

6. J.U. Kref, G. Booth and J.W. Wimpenny. BacSim, a simulator for individual-based modelling of bacterial colony growth. *Microbiology*, 144:3275-3287, 1998.
7. A.M. Lazdunski, I. Ventre and J. N. Sturgis. Regulatory Circuits and Communication in Gram-Negative Bacteria. *Nature Reviews.*, 2:581-592, 2004.
8. J.B. Lyczak, C.L. Cannon and G.B. Pier. Establishment of Pseudomonas Aeruginosa infection: lessons from a versatile opportunistic. *Microbes Infect.*, 2:1051-1060, 2000.
9. G. Păun. Membrane Computing: An Introduction. Springer, 2002.
10. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, E. Shapiro. BioAmbients: An Abstraction for Biological Compartments. *Theoretical Computer Science*, Special Issue on Computational Methods in Systems Biology. Volume 325(1):141-167, 2004. Elsevier.
11. M.L. Simpson, C.D. Cox, G.D. Peterson and G.S. Sayler. Engineering in the Biological Substrate: Information Processing in Genetic Circuits. *Proceedings of the IEEE*, 92(5):848-863, 2004.
12. S. Swift, J.A. Downie, N.A. Whitehead, A.M.L. Barnard,G.P.C. Salmond and P. Williams. Quorum sensing as a population-density-dependent determinant of bacterial physiology. *Adv Micro Physiol*, 45:199-270, 2001.
13. C. Martin-Vide, G. Paun and G. Rozenberg. Membrane Systems with Carriers. *Theoretical Computer Science*, 270:779-796, 2002.
14. A.U. Viretta and M. Fussenenger. Modeling the Quorum Sensing Regulatory Network of Human-Pathogenic Pseudomonas aeruginosas. *Biotechnol. Prog.*, 20:670-678, 2004.
15. J.P. Ward, J.R. King, A.J. Koerber, J.M. Croft, R.E. Socket and P. Williams. Cell-signalling Repression in Bacterial Quorum Sensing. *Mathematical Medicine and Biology*, 21:169-204, 2004.
16. J.P. Ward, J.R. King, A.J. Koerber, P. Williams, J.M. Croft and R.E. Socket. Mathematical Modelling of Quorum Sensing in Bacteria. *IMA Journal of Mathematics Applied in Medicine and Biology*, 18:263-292, 2001.
17. V.E. Wagner, D. Bushnell, L. Passador, L. Brooks and A.I. Iglewski. Microarray Analysis of Pseudomonas aeruginosa quorum-sensing regulons:effects of growth phase and environment. *J. Bacteriology* , 185:2080-2095, 2003.
18. R. Weiss, S. Basu, A. Kalmbach, D. Karig, R. Mehreja and I. Netravali. Genetic Circuit Building Blocks for Cellular Computation, Communications and Signal Processing. *Natural Computing*, 2(1):47-84, 2003.
19. R. Weiss and T.F. Knight. Engineered Communications for Microbial Robotics. *Lecture Notes in Computer Science*, 2054, 2001.
20. P. Williams, M. Camara, A. Hardman, S. Swift, D. Milton, V.J. Hope, K. Winzer, B. Middleton, D.I. Pritchard and B.W. Bycroft. Quorum sensing and the population dependent control of virulence. *Phil. Trans Roy Soc London B*, 355(1397):667-680 2000.
21. K. Winzer, K.H. Hardie and P. Williams. Bacterial cell-to-cell communication: sorry can't talk now – gone to lunch!. *Curr Op. Microbiol*, 5:216-222 2002.

Kripke Models, Distributive Lattices, and Medvedev Degrees*

Sebastiaan A. Terwijn

Institute for Discrete Mathematics and Geometry,
Technical University of Vienna,
Wiedner Hauptstrasse 8–10 / E104,
A-1040 Vienna, Austria
terwijn@logic.at
http://www.logic.at/people/terwijn

Abstract. We define a simple variation of the standard Kripke semantics motivated by the connection between constructive logic and the Medvedev lattice. We show that while the new semantics is still complete, it gives a simple and direct correspondence between Kripke models and algebraic structures such as factors of the Medvedev lattice.

1 Introduction

Immediately after Heyting had isolated the axioms of intuitionistic logic in [1] people began to wonder what the precise interpretation of these axioms was. In the course of history many (complete) interpretations have been given, one of the most famous being Kripke semantics [6]. Also several interpretations were suggested that later turned out to be incomplete. Among these are Kleene's realizability [4] and the approach by Kolmogorov and Medvedev that will be discussed below. Early on certain complete algebraic interpretations were given, for example in Jaśkowski [3]. It is to be noted that these algebraic interpretations, as well as the later Kripke semantics, have little or nothing to do with the basic intuitions surrounding intuitionistic logic, namely those of effectivity, constructivity, or computability (cf. the informal Brouwer-Heyting-Kolmogorov interpretation [16]). Kolmogorov [5] suggested to interpret the propositional connectives constructively using a "calculus of problems". This idea was later implemented in various ways by Medvedev [7, 8]. Although this approach initially failed to give a complete semantics for the intuitionistic propositional logic IPC, the structures introduced by Medvedev turned out to be interesting for different reasons as well. In particular, there are interesting connections with the Turing

* We are grateful to Rosalie Iemhoff for helpful remarks concerning intuitionistic logic and to Andrea Sorbi for discussions about the Medvedev lattice.
The author was supported by the Austrian Research Fund (FWF) under grant P17503.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 486–494, 2005.

degrees and other structures from computability theory, cf. Sorbi [13]. Moreover, Skvortsova [10] showed that the main idea of the Kolmogorov/Medvedev approach could be used to give a complete computational semantics for IPC after all (see below). The results in this area are a particularly attractive blend of results and methods from computability theory (such as lattice embedding results) with proof-theoretic ones (coming from the study of intuitionistic and intermediate logics).

In this note we present a variant of Kripke semantics in which only the interpretation of "or" is different. We show that this alternative interpretation is sound and complete for "distributive" structures, where the notion of distributivity is a rather liberal one, applying to partial orders in general instead of just lattices. The new notion of forcing will be denoted by $\Vdash^*$, and the new kind of Kripke models under this forcing notion will be called Kripke* models. The variation is motivated by the observation that every configuration in the Medvedev lattice can be interpreted as a Kripke* model. This gives a direct relation between Kripke* models on the one hand and the Medvedev lattice on the other.

We briefly recall the definition of the Medvedev lattice $\mathfrak{M}$. Let ω denote the natural numbers and let ω^ω be the set of all functions from ω to ω (Baire space). A *mass problem* is a subset of ω^ω. One can think of such subsets as a "problem", namely the problem of producing an element of it, and so we can think of the elements of the mass problem as its set of solutions. A mass problem $\mathcal{A}$ *Medvedev reduces* to mass problem $\mathcal{B}$ if there is an effective procedure of transforming solutions to $\mathcal{B}$ into solutions to $\mathcal{A}$: $\mathcal{A} \leq \mathcal{B}$ if there is a partial computable functional $\Psi : \omega^\omega \to \omega^\omega$ such that for all $f \in \mathcal{B}$, $\Psi(f)$ is defined and $\Psi(f) \in \mathcal{A}$.[1] This can be seen as an implementation of Kolmogorov's idea of a calculus of problems. The relation $\leq$ induces an equivalence relation on the mass problems: $\mathcal{A} \equiv \mathcal{B}$ if $\mathcal{A} \leq \mathcal{B}$ and $\mathcal{B} \leq \mathcal{A}$. The equivalence class of $\mathcal{A}$ is denoted by $[\mathcal{A}]$ and is called the *Medvedev degree* of $\mathcal{A}$. We denote Medvedev degrees by boldface symbols. Note that there is a smallest Medvedev degree, denoted by $\mathbf{0}$, namely the degree of any mass problem containing a computable function. There is also a largest degree $\mathbf{1}$, the degree of the empty mass problem, of which it is impossible to produce an element by whatever means. Finally, it is possible to define a meet operator $\times$ and a join operator $+$ on mass problems: For functions f and g, as usual define the function $f \oplus g$ by $f \oplus g(2x) = f(x)$ and $f \oplus g(2x+1) = g(x)$. Let $n^\frown\mathcal{A} = \{n^\frown f : f \in \mathcal{A}\}$, where $^\frown$ denotes concatenation.

Define

$$\mathcal{A} + \mathcal{B} = \{f \oplus g : f \in \mathcal{A} \wedge g \in B\}$$

[1] Note that although Medvedev reducibility is designed especially for sets of functions, it is close in spirit to ordinary Turing reducibility $\leq_T$ for individual functions: $g \leq_T f$ if there is a partial computable functional Φ that is defined at least on f such that $\Phi(f) = g$. One may also compare Medvedev reducibility to Wadge reducibility $\leq_W$ from descriptive set theory: For sets of reals (or mass problems) $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \leq_W \mathcal{B}$ if there is a continuous functional Φ such that $\Phi^{-1}(\mathcal{B}) = \mathcal{A}$. Note that this notion is quite different: the continuity is in the other direction, there is no 'if and only if', and no effectivity.

and

$$\mathcal{A} \times \mathcal{B} = 0^\frown\mathcal{A} \cup 1^\frown\mathcal{B}.$$

It is not hard to show that $\times$ and $+$ indeed define a greatest lower bound and a least upper bound operator on the Medvedev degrees. For more information and discussion on $\mathfrak{M}$ we refer to Sorbi [13], Terwijn [14, 15].

A distributive lattice $\mathfrak{L}$ with 0, 1 is called a *Brouwer algebra* if for any elements a and b it holds that the element $a \to b$ defined by

$$a \to b := \text{least}\big\{c \in \mathfrak{L} : b \leq a + c\big\}$$

always exists. $\mathfrak{L}$ is called a *Heyting algebra* if its dual is a Brouwer algebra. Medvedev [7] showed that $\mathfrak{M}$ is a Brouwer algebra, and Sorbi [11] showed that $\mathfrak{M}$ is not a Heyting algebra.

Above we have already defined the operations $\times$, $+$, and $\to$ on $\mathfrak{M}$. We can also define a negation operator $\neg$ by defining $\neg\mathbf{A} = \mathbf{A} \to \mathbf{1}$ for any Medvedev degree $\mathbf{A}$.

Given any Brouwer algebra $\mathfrak{L}$ (such as $\mathfrak{M}$) with join denoted by $+$ and meet by $\times$, we can evaluate formula's as follows. An $\mathfrak{L}$-*valuation* is a function σ : Form $\to \mathfrak{L}$ from propositional formulas to $\mathfrak{L}$ such that for all formulas A and B, $\sigma(A \vee B) = \sigma(A) \times \sigma(B)$, $\sigma(A \wedge B) = \sigma(A) + \sigma(B)$, $\sigma(A \to B) = \sigma(A) \to \sigma(B)$, $\sigma(\neg A) = \sigma(A) \to 1$. (Note the upside-down reading of $\wedge$ and $\vee$ when compared to the usual lattice theoretic interpretation. For more remarks regarding notation see [14].) Write $\mathfrak{L} \models A$ if $\sigma(A) = 0$ for any $\mathfrak{L}$-valuation σ. Finally, define

$$\text{Th}(\mathfrak{L}) = \big\{\alpha : \mathfrak{L} \models \alpha\big\}.$$

A *B-homomorphism* is a mapping between Brouwer algebras preserving $+$, $\times$, $\to$, and 0 and 1. A *B-embedding* is an injective B-homomorphism. Note that if $\mathfrak{L}_1$ is B-embeddable into $\mathfrak{L}_2$ then $\text{Th}(\mathfrak{L}_2) \subseteq \text{Th}(\mathfrak{L}_1)$ because every $\mathfrak{L}_1$-valuation is also a $\mathfrak{L}_2$-valuation. If there is a (not necessarily injective) B-homomorphism from $\mathfrak{L}_1$ onto $\mathfrak{L}_2$ we obtain the converse $\text{Th}(\mathfrak{L}_1) \subseteq \text{Th}(\mathfrak{L}_2)$. Note that the top element has to be preserved in order to fix the meaning of negation.

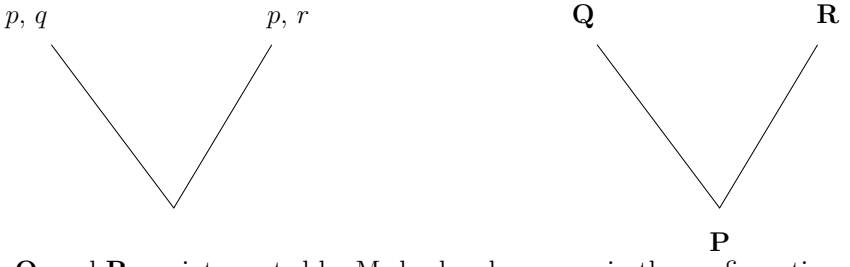
It follows from results of Medvedev [8] and Jankov [2] that $\text{Th}(\mathfrak{M})$ is the deductive closure of IPC and the weak law of the excluded middle $\neg\alpha \vee \neg\neg\alpha$ (also known as De Morgan, or Jankov logic). Although this shows that the Kolmogorov/Medvedev approach for providing a semantics for IPC does not work directly, the ideas can still be used to interpret IPC. Namely, one can consider *factors* of $\mathfrak{M}$, i.e. study $\mathfrak{M}$ modulo a filter or an ideal. Given a Brouwer algebra $\mathfrak{L}$ and an ideal I in $\mathfrak{L}$, $\mathfrak{L}/I$ is still a Brouwer algebra. If G is a filter in $\mathfrak{L}$ then $\mathfrak{L}/G$ is not necessarily a Brouwer algebra, but if G is principal then $\mathfrak{L}/G$ is again a Brouwer algebra. In such a factorized lattice G plays the role of 1. E.g. if G is the principal filter in $\mathfrak{M}$ generated by the degree $\mathbf{D}$ then negation in $\mathfrak{M}/G$ can be defined by $\neg\mathbf{A} = \mathbf{A} \to \mathbf{D}$. The algebraic properties of $\mathfrak{M}/G$ are directly related to the theories $\text{Th}(\mathfrak{M}/G)$. For example, whether the weak law of the excluded middle holds in $\text{Th}(\mathfrak{M}/G)$ is related to whether the element that generates G is join-reducible in $\mathfrak{M}$. (For more information see [13, 14].) Skvortsova [10] showed that there are indeed factors that capture IPC: There exists a principal filter G such that $\text{Th}(\mathfrak{M}/G) = \text{IPC}$.

2 A Variant of Kripke Semantics

The reader may observe that a configuration in the Medvedev degrees bears much resemblance to a Kripke model. We have already seen that negation in $\mathfrak{M}$ causes problems, but this can be remedied by considering suitable factors $\mathfrak{M}/G$. Let us for the moment consider only formulas without negation, so that Medvedev's theorem (that the positive fragments of $\mathrm{Th}(\mathfrak{M})$ and IPC coincide [8]) applies. Consider for example the following formula φ:

$$(p \to q \vee r) \to ((p \to q) \vee (p \to r))$$

This formula is not derivable in IPC. The left part of the following figure shows a Kripke countermodel for φ, and the right part is a configuration of Medvedev degrees that shows that $\varphi \notin \mathrm{Th}(\mathfrak{M})$.



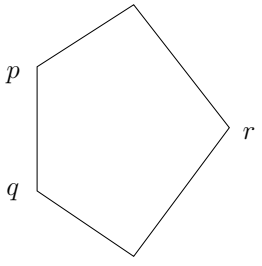
If $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$ are interpreted by Medvedev degrees as in the configuration on the right, where $\mathbf{P} = \mathbf{Q} \times \mathbf{R}$, then φ does not evaluate to $\mathbf{0}$, because $\mathbf{P} \geq \mathbf{Q} \times \mathbf{R}$ but neither $\mathbf{P} \geq \mathbf{Q}$ nor $\mathbf{P} \geq \mathbf{R}$. This example shows that there is a difference in interpretation between $\vee$ in Kripke models and $\times$ in the Medvedev lattice: $A \vee B$ holds in a node k in a Kripke model only if A or B holds in k, but in the above configuration $\mathbf{Q} \times \mathbf{R}$ "holds" in $\mathbf{P}$ but neither $\mathbf{Q}$ nor $\mathbf{R}$ does. We now show that a rather harmless variation of the standard Kripke semantics, where only the interpretation of $\vee$ is changed, gives a precise connection between Kripke models on the one hand and configurations in $\mathfrak{M}$ on the other.

Let $\Vdash$ denote the usual forcing relation in Kripke models, cf. [16]. We define a variant $\Vdash^*$ where only $\vee$ has a slightly different interpretation. Namely, for a propositional Kripke model M, a node $k \in M$, and propositional formulas A and B, we let $k \Vdash^* A \vee B$ if any one of the following holds:

1. $k \Vdash^* A$ or $k \Vdash^* B$,
2. $(\exists k' \leq k)[k' \Vdash^* A \vee B]$
3. There exist k_0, $k_1 \in M$ such that $k_0 \Vdash^* A$, $k_1 \Vdash^* B$, and k is the greatest lower bound of k_0 and k_1 in M.

Also, if $k \Vdash^* A \vee A$ we let $k \Vdash^* A$. The clauses for $\wedge$, $\to$, and $\neg$ for $\Vdash^*$ are identical to those for $\Vdash$. For example, $k \Vdash^* A \to B$ if in every $k' \geq k$ with $k' \Vdash^* A$ it holds that $k' \Vdash^* B$. Thus, in the new semantics, if a node k is the meet of a node where A holds and another where B holds, then $A \vee B$ holds in k. We will call a Kripke model under this new semantics a *Kripke* model*.

First we note that the Kripke* semantics is sound only for *distributive* structures: Consider the nondistributive lattice N_5, interpreted as a Kripke* model with the atoms p, q and r holding in the nodes as in the following figure:



Then for the distributive law $A = (q \vee p) \wedge (r \vee p) \rightarrow (q \wedge r) \vee p$ it holds that in the "q-node" k we have that $k \Vdash A$ but not $k \Vdash^* A$: $k \Vdash^* (q \vee p) \wedge (r \vee p)$ because of the new interpretation of $\vee$, but still $k \nVdash^* (q \wedge r) \vee p$. Since A is of course intuitionistically valid, we see that the Kripke* semantics is in general not sound for IPC. It is so, however, if we restrict it to *distributive* structures:

Definition 1. We call a partial order $\langle P, \leq \rangle$ *distributive* if for all $k_0, k_1, l \in P$, if k_0 and k_1 have a greatest lower bound k in P such that $k \leq l$, then there are $l_0 \geq k_0$ and $l_1 \geq k_1$ in P such that l is the greatest lower bound of l_0 and l_1.

One can easily check that for lattices the property from Definition 1 is equivalent to distributivity. I.e. a lattice is distributive (satisfies the distributive laws) if and only if it is distributive as a partial order (in the sense of Definition 1). But note that the property from Definition 1 is more general since it may also hold on structures that are not lattices. Thus we will call a Kripke* model that is distributive as a partial order a *distributive Kripke* model*.

Proposition 2. (Soundness) *For all propositional formulas A, $\vdash A$ (i.e. A is derivable in* IPC*) implies that $L \Vdash^* A$ for all distributive Kripke* models L.*

Proof. We prove this by induction on derivations in the natural deduction presentation of IPC, cf. [16]. Let L be a distributive Kripke* model. Suppose that the last inference rule used in the derivation is the elimination rule $\vee$E:

$$\frac{\Gamma_1 \vdash A \vee B \qquad \Gamma_2, A \vdash C \qquad \Gamma_3, B \vdash C}{\Gamma_1, \Gamma_2, \Gamma_3 \vdash C}$$

The induction hypothesis is that for all $l \in L$, $l \Vdash^* \Gamma_1 \Rightarrow l \Vdash^* A \vee B$, $l \Vdash^* \Gamma_2, A \Rightarrow l \Vdash^* C$, and $l \Vdash^* \Gamma_3, B \Rightarrow l \Vdash^* C$. Suppose that for $k \in L$ we have $k \Vdash^* \Gamma_1, \Gamma_2, \Gamma_3$. We have to prove that $k \Vdash^* C$. By induction hypothesis we have $k \Vdash^* A \vee B$. If $k \Vdash^* A$ or $k \Vdash^* B$ we are done. Otherwise, there are nodes k_0 and k_1 with greatest lower bound l such that $k_0 \Vdash^* A$, $k_1 \Vdash^* B$, and $l \leq k$. Now by distributivity there are $k_0' \geq k_0$ and $k_1' \geq k_1$ with meet k. We then have $k_0 \Vdash^* A$ and $k_0 \Vdash^* \Gamma_2$, hence by induction hypothesis $k_0' \Vdash^* C$. Similarly we have $k_1' \Vdash^* C$. Hence $k \Vdash^* C \vee C$, and thus $k \Vdash^* C$.

The induction steps for the other IPC-rules are all straightforward, and do not need distributivity. □

The following correspondence between Kripke* models and Brouwer algebras will be useful, both for the proof of completeness (Theorem 4) and in section 3. Let $\mathfrak{L}$ be any Brouwer algebra with largest element 1 and let $\mathfrak{K} = \mathfrak{L} - \{1\}$. Then we can interpret $\mathfrak{K}$ as a Kripke* model by stipulating that for every node $P \in \mathfrak{K}$, P (as a propositional atom) holds in every node $Q \in \mathfrak{K}$ with $Q \geq P$. So the degrees in $\mathfrak{K}$ play the role of the nodes as well as of the propositional atoms. Let A be a propositional formula, with atoms from $\mathfrak{K} \cup \{\bot\}$. Then we can interpret A as a point of $\mathfrak{K}$ as in section 1, where $\vee$ is interpreted by $\times$ and $\wedge$ by $+$, and $\bot$ by 1. The reason that 1 is left out of the subalgebras that we consider is to guarantee that no degree forces $\bot$. We now have the following simple observation about such formula's A and the forcing relation $\Vdash^*$.

Proposition 3. *For any $P \in \mathfrak{K}$, $P \Vdash^* A$ (in the Kripke* model $\mathfrak{K}$) if and only if $P \geq A$ (in the Brouwer algebra $\mathfrak{K}$). In particular we have that $\mathfrak{K} \Vdash^* A$ if and only if $\mathfrak{K} \models A$.*

Proof. By formula induction on A. The atomic case: For $Q \in \mathfrak{K}$ we have $P \Vdash^* Q \Leftrightarrow P \geq Q$ by definition of the interpretation of $\mathfrak{K}$ as a Kripke* model. For $A = \bot$ we have $P \nVdash^* \bot$ by definition of $\Vdash^*$, and $P \not\geq 1$ because $1 \notin \mathfrak{K}$.

For the case $A \vee B$ note that $P \Vdash^* A \vee B$ if and only if $P \Vdash^* A$ or $P \Vdash^* B$ (in which cases we immediately obtain $P \geq A \times B$) or there is $P' \leq P$ with $P' = Q \times R$ such that $Q \Vdash^* A$ and $R \Vdash^* B$. In the latter case we have $Q \geq A$ and $R \geq B$, and hence also $P \geq Q \times R \geq A \times B$.

For the case $A \to B$ we have

$$\begin{aligned} P \Vdash^* A \to B &\Longleftrightarrow (\forall Q \geq P)[Q \Vdash^* A \Rightarrow Q \Vdash^* B] \\ &\Longleftrightarrow (\forall Q \geq P)[Q \geq A \Rightarrow Q \geq B] \\ &\Longleftrightarrow (\forall Q \geq P)[Q + A \geq B] \\ &\Longleftrightarrow P + A \geq B \\ &\Longleftrightarrow P \geq A \to B. \end{aligned}$$

The case $\neg A$ is a special case of the previous case, and the case $A \wedge B$ is straightforward. □

The next theorem shows that under the restriction to distributive models we still have completeness for IPC, so we see that our variation does not change anything essential. In fact, we can apply an old result of Jaśkowski [3] to show completeness with respect to the smaller class of Kripke* models whose frame is a finite Brouwer algebra. Note that in such Kripke* models, by definition of $\Vdash^*$, every atom occurs essentially only once. (Namely an atom P is forced in the meet of all nodes where P is forced, so that modulo upwards consistency P occurs only once.)

Theorem 4. (Completeness) *For any propositional formula A the following are equivalent:*

(i) $\Vdash A$ *(A is valid on all Kripke models)*
(ii) $L \Vdash^* A$ *for all distributive Kripke* models L.*
(iii) $L \Vdash^* A$ *for all Kripke* models L that are also distributive lattices.*
(iv) $L \Vdash^* A$ *for all finite Kripke* models L that are also Brouwer algebras.*

Proof. (i)$\Longrightarrow$(ii). Suppose that K is a distributive Kripke* model on which A does not hold. We claim that K is automatically a Kripke countermodel for A. Namely, we prove that for all $k \in K$ and formulas A

$$k \Vdash A \Longrightarrow k \Vdash^* A$$

We prove this by formula induction on A.

The atomic case is immediate. The cases $A = B \wedge C$ and $B \vee C$ are immediate by definition of $\Vdash^*$.

Suppose that $A = B \to C$. Suppose that $k \in K$ has $k \Vdash B \to C$, and suppose for a contradiction that $k \not\Vdash^* B \to C$. This can only be if, in going from $\Vdash$ to $\Vdash^*$, B holds on more $k' \geq k$ and C holds on less $k' \geq k$. By induction hypothesis, the latter does not occur. So there must be $k' \geq k$ such that $k' \Vdash^* B$, $k' \not\Vdash B$, and $k' \not\Vdash^* C$. Ultimately, the points in K responsible for $k' \Vdash^* B$ all have $\Vdash B$. By distributivity all these points are *above* k. But $k \Vdash B \to C$, so in all these points $\Vdash C$. But then also $k' \Vdash^* C$, contradiction.

Finally, the case that $A = \neg B$ is similar (and easier) than the case $A = B \to C$. This completes the induction.

(ii)$\Longrightarrow$(iii) and (iii)$\Longrightarrow$(iv) are immediate.

(iv)$\Longrightarrow$(i). This follows from Proposition 3 and the result of Jaśkowski [3] that IPC $= \bigcap \{\mathrm{Th}(\mathfrak{L}) : \mathfrak{L}$ is a finite Brouwer algebra$\}$. (A proof of this result is in Rose [9][2].) □

We noted above that in a semantics for IPC we can restrict ourselves to finite Kripke* models were every propositional atom occurs at most once, modulo upward consistency. This will enable us in the next section to obtain the desired correspondence between Kripke* models and sets of Medvedev degrees.

We note that in infinite distributive Kripke* models it is not necessarily the case that every propositional atom occurs at most once. (For a counterexample, consider a frame with an infinite descending chain of nodes without infimum, in each of which p is forced.) However, the property that every atom occurs essentially only once holds for all Kripke* models that are also κ-complete lattices, where κ is the cardinality of the lattice.

[2] Instead of algebras, Jaśkowski and Rose use the more general notion of matrix in the proof of this result. It can be checked that the sequence of matrices J_i in [9] that give IPC is in fact a sequence of Brouwer algebras. The order can be defined by $x \leq y :\equiv x \vee y = x$.

3 Kripke Semantics and the Medvedev Lattice

After the groundwork from the previous section we can now finally give the correspondence between Kripke semantics and configurations in the Medvedev lattice.

Consider any Brouwer subalgebra $\mathfrak{K}$ of $\mathfrak{M}$. Then we can interpret $\mathfrak{K} - \{\mathbf{1}\}$, ordered by the Medvedev reducibility $\leq$, as a Kripke* model as in section 2. In particular, by Proposition 3 we have that for any $\mathbf{P} \in \mathfrak{K}$, $\mathbf{P} \Vdash^* A$ if and only if $\mathbf{P} \geq A$. This holds equally well for any factor $\mathfrak{M}/G$ of the Medvedev degrees, where $\mathbf{1}$ is interpreted by a principal filter G.

For a Kripke* model K and a subalgebra $\mathfrak{K} \subseteq \mathfrak{M} - \{\mathbf{1}\}$ let us write $K \cong \mathfrak{K}$ if $\mathfrak{K}$ is isomorphic to K as a Kripke* model.

Theorem 5. *For every Kripke* model K that is also a finite Brouwer algebra there is a finite set of degrees $\mathfrak{K} \subseteq \mathfrak{M} - \{\mathbf{1}\}$ such that $K \cong \mathfrak{K}$, and vice versa.*

Proof. We have just seen that every $\mathfrak{K} \subseteq \mathfrak{M} - \{\mathbf{1}\}$ can be seen as a Kripke* model. Conversely, if K is a Kripke* model whose frame is a finite Brouwer algebra then by Sorbi [12], K is embeddable (as a Brouwer algebra) into $\mathfrak{M}$ (maybe without preserving 1; If K has an irreducible 1 then the top can be preserved as well). □

Of course, Theorem 5 does not make it any easier to find examples of G with $\mathrm{Th}(\mathfrak{M}/G) = \mathrm{IPC}$. It merely points out the relation between Kripke semantics and validity in degree structures such as $\mathfrak{M}$.

Theorem 5 also holds for any of the factors $\mathrm{Th}(\mathfrak{M}/G)$ if G is generated by a degree different from $\mathbf{0}$ or $\mathbf{0}'$:

Theorem 6. *Let G be a principal filter generated by a degree $\mathbf{D} > \mathbf{0}'$. For every Kripke* model K that is also a finite Brouwer algebra there is a finite set of degrees $\mathfrak{K} \subseteq \mathfrak{M}/G - \{\mathbf{D}\}$ such that $K \cong \mathfrak{K}$, and vice versa.*

Proof. As noted above, every $\mathfrak{K} \subseteq \mathfrak{M}/G - \{\mathbf{D}\}$ can be interpreted as a Kripke* model. Conversely, if K is a Kripke* model whose frame is a finite Brouwer algebra then by the proof of [12–Theorem 2.11], K is embeddable (as a Brouwer algebra, but maybe without preserving $\mathbf{1}$) into $\mathfrak{M}/G$. □

References

1. A. Heyting, *Die formalen Regeln der intuitionistischen Logik*, Sitzungsberichte der Preussisischen Akademie von Wissenschaften, Physikalisch-mathematische Klasse (1930) 42–56.
2. A. V. Jankov, *Calculus of the weak law of the excluded middle*, Izv. Akad. Nauk SSSR Ser. Mat. 32 (1968) 1044–1051. (In Russian.)
3. S. Jaśkowski, *Recherches sur le système de la logique intuitioniste*, Actes du Congrès International de Philosophie Scientifique VI, Philosophie des mathématiques, Actualités Scientifiques et Industrielles 393, Paris, Hermann (1936) 58–61.
4. S. C. Kleene, *On the interpretation of intuitionistic number theory*, Journal of Symbolic Logic 10 (1945) 109–124.

5. A. Kolmogorov, *Zur Deutung der intuitionistischen Logik*, Mathematische Zeitschrift 35(1) (1932) 58-65.
6. S. Kripke, *Semantical analysis of intuitionistic logic I*, in: *Formal systems and recursive functions*, J. N. Crossley and M. A. Dummett (eds.), North-Holland (1965) 92–130.
7. Y. T. Medvedev, *Degrees of difficulty of the mass problems*, Dokl. Akad. Nauk. SSSR 104(4) (1955) 501–504.
8. Y. T. Medvedev, *Finite problems*, Dokl. Akad. Nauk. SSSR (NS) 142(5) (1962) 1015–1018.
9. G. F. Rose, *Propositional calculus and realizability*, Transactions of the American Mathematical Society 75 (1953) 1–19.
10. E. Z. Skvortsova, *A faithful interpretation of the intuitionistic propositional calculus by means of an initial segment of the Medvedev lattice*, Sibirsk. Math. Zh. 29(1) (1988) 171–178. (In Russian.)
11. A. Sorbi, *Some remarks on the algebraic structure of the Medvedev lattice*, Journal of Symbolic Logic 55(2) (1990) 831–853.
12. A. Sorbi, *Embedding Brouwer algebras in the Medvedev lattice*, Notre Dame Journal of Formal Logic 32(2) (1991) 266–275.
13. A. Sorbi, *The Medvedev lattice of degrees of difficulty*, In: S. B. Cooper, T. A. Slaman, and S. S. Wainer (eds.), Computability, Enumerability, Unsolvability: Directions in Recursion Theory, London Mathematical Society Lecture Notes 224, Cambridge University Press, 1996, 289–312.
14. S. A. Terwijn, *Constructive logic and the Medvedev lattice*, submitted.
15. S. A. Terwijn, *The Medvedev lattice of computably closed sets*, to appear in Archive for Mathematical Logic.
16. A. S. Troelstra and D. van Dalen, *Constructivism in Mathematics*, Vol. I, Studies in logic and the foundations of mathematics Vol. 121, North-Holland, 1988.

Arthur-Merlin Games and the Problem of Isomorphism Testing

Jacobo Torán

Abt. Theoretische Informatik, Universität Ulm,
D-89069 Ulm, Germany
toran@informatik.uni-ulm.de

Abstract. We survey some recent results related to the complexity of isomorphism testing in different algebraic structures. We concentrate on isomorphism problems for finite groups and rings. The complexity of these problems depends not only on the particular underlying algebraic structure, but also on the way the input instances are encoded. As in the case of better studied isomorphism questions, like graph isomorphism, Arthur-Merlin games provide a good tool for proving upper bounds for these problems in terms of complexity classes. We consider questions related to the number of random and nondeterministic bits used in the Arthur-Merlin protocols for isomorphism testing as well as the derandomization of these protocols.

1 Introduction

Arthur-Merlin games were introduced by Babai in [4] providing a better way to classify some computational problems that did not fit very well in the complexity classes of the polynomial time hierarchy. The class AM can be considered as a natural randomized version of NP. In the same way as NP is the class of languages with short membership proofs, the languages in AM have also efficient membership proofs, but in this case the process of proving membership can use randomness and it is based on statistical evidence. It turns out that several important algebraic problems are in either NP $\cap$ coAM or AM $\cap$ coAM. In particular it is well known that problem of testing isomorphism of two given graphs, GRAPH-ISO lies in NP $\cap$ coAM. Babai conjectures in [5] that some of the group problems classified in AM actually belong to NP but the proof of this fact should rely on involved results related to the classification of finite simple groups, some of which are still open. Babai's motivation for introducing the Arthur-Merlin games was to trade the group theory and unproven hypothesis needed for proving these results for the randomization in the definition of AM.

With the development of derandomization techniques, several researchers have tried to eliminate the randomness needed in the Arthur-Merlin games. Arvind and Köbler [2] use an average-case hardness assumption to construct a pseudo-random number generator that suffices for derandomizing AM and therefore for showing that graph isomorphism is in NP $\cap$ coNP. This result is

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 495–506, 2005.

improved to a worst-case hardness assumption by Klivans and van Melkebeek in [12]. Another attempt was made by Miltersen and Vinodchandran [16], using hitting sets instead of pseudo-random generators. In a surprising recent result [18] it has been shown that the three hardness hypothesis are in fact equivalent.

These results are examples of hardness versus randomness trade-offs in derandomization. Somehow they go the opposite way as Babai with his Arthur-Merlin games: they trade randomness for unproven hypothesis, but in this case unproven hypothesis related to complexity separations instead of group theory.

In this survey we consider the problem of testing isomorphism of two further finite algebraic structures: groups and rings. For the group isomorphism problem, GROUP-ISO, the input consists of the operation tables (Cayley tables) of two groups of the same order n. The input has therefore polynomial length in n. For the case of ring isomorphism, RING-ISO, the input is given in a more succinct way (explained in Section 2) so that rings of order n are encoded as strings of polylogarithmic length in n. These two different ways to encode the structures provide an example on how the input encoding affects the complexity of the problem. Intuitively speaking, the shorter the input relative to the size of the structure, the harder becomes the problem. Presented this way, GROUP-ISO is reducible to graph isomorphism while GRAPH-ISO is reducible to RING-ISO[11]

$$\text{GROUP-ISO} \leq_m^p \text{GRAPH-ISO} \leq_m^p \text{RING-ISO}.$$

On the other hand GROUP-ISO seems to be strictly easier that GRAPH-ISO. This is because groups of order n have generator sets of size bounded by $\log n$ and because of this fact an isomorphism algorithm running in time $n^{\log n + O(1)}$ can be obtained by computing a generator set of size $\log n$ in one of the groups, and mapping this set in every possible way to a set of elements in the second group. The map has to be extended to the entire group using the Cayley table and then it has to be checked that it defines an isomorphism. This algorithm is attributed to Tarjan in [15]. A stronger result showing that GROUP-ISO can be solved in space $O(\log^2 n)$ was given in [13].

RING-ISO appears to be strictly the hardest of the three problems. Kayal and Saxena prove in [11] that the problem of factoring natural numbers is reducible to the counting version of ring isomorphism. This is not known to hold for GRAPH-ISO.

The purpose of studying these two problems in connection with GRAPH-ISO is twofold. On the one hand we want to know how far the techniques known for the derandomization of the AM protocols for GRAPH-ISO can further be pushed when an easier problem is considered. The goal here is to show that GROUP-ISO is in $\text{NP} \cap \text{coNP}$. On the other hand, since RING-ISO seems to be a more structured problem than GRAPH-ISO, the use of algebraic properties of rings might provide a way to design an efficient algorithm for the ring isomorphism problem, thus solving also graph isomorphism. For example, in [11] it is shown that the ring automorphism problem is in P, a result that is not known for graph automorphism.

This survey is organized as follows: we give in Section 2 the group theoretic notions needed in the paper as well as the formal definitions of GROUP-ISO and

RING-ISO. In Section 3 we present an AM protocol for ring non-isomorphism [11], and show that GROUP-NONISO can be solved by Arthur-Merlin games of a restricted kind in which only a polylogarithmic amount of random bits (for the verifier) and nondeterministic bits (for the prover) are needed [3]. We explain in Section 4 how the existing derandomization results for AM protocols can be improved for the protocols for GROUP-NONISO. This survey ends with some conclusions and open problems.

2 Encodings of Finite Groups and Rings

A finite group of order n can be given by the Cayley table of its elements. Since each element can be encoded as a sting of $\log n$ bits, the whole table can be presented as a string of size $O(n^2 \log n)$. Two groups G_1 and G_2 are isomorphic if there is a bijection (isomorphism) φ between the groups such that for every pair $i, j \in G_1$, $\varphi(ij) = \varphi(i)\varphi(j)$. The group isomorphism problem is defined as

GROUP-ISO$=\{(G_1, G_2) \mid G_1, G_2$ are isomorphic groups given as Cayley tables$\}$

Many groups however can be encoded in a much more succinct way than the Cayley tables.

Definition 1. *A presentation of a group G with n elements is a pair (X, R) where X is a generating set for G, R is a set of relations written as equations using powers of the generators. $w \in R$ defines the equation $w = 1$ and (X, R) defines the group G in the sense that there is an algorithm that on input a presentation for a group G computes the Cayley table of a group G' isomorphic to G.*

For a constant $c > 0$ we say that a presentation (X, R) is c-short if its length $|(X, R)|$ is at most $\log^c n$.

For example, the cyclic group of order 6 can be encoded by the pair (X, R) with $X = \{a\}$ and R is the relation $a^6 = 1$. Observe that a group can have several different presentations, for example (X', R') with $X' = \{b, c\}$ and $R' = \{b^3 = 1, c^2 = 1, b^{-1}c^{-1}bc = 1\}$ is a different presentation for the same group. Every Abelian group of order n can be encoded by a presentation of size at most $\log^2 n$. This follows by the Structure Theorem for Abelian groups, that states that any such group of order n can be expressed as direct product of cyclic groups of orders $q_1, \ldots, q_m$ with $\prod q_i = n$, and the fact that the direct product of two groups with short presentations has also a short presentation (cf. [8]).

The *short presentation conjecture* (see [7–Conjecture 1] for details) states that there is a constant c such that every finite group has a c-short presentation.

Our results for derandomizing Arthur-Merlin protocols for GROUP-NONISO, work for the case groups satisfying certain conditions: the groups must have short presentations that are efficiently computable. Moreover, from the short presentation it has to be possible to compute the table of the group using only

polylogarithmic space. We showed in [3] that the class of solvable groups fulfill these properties.

Theorem 2. *[3] Let G be solvable group with n elements, given by its Cayley table, there is a polynomial (in n) time algorithm that inductively computes a short presentation (X, R) for G. Moreover, the size $|(X, R)|$ is $c \log^c n$, where c is a fixed constant independent of the group.*

Theorem 3. *Let (X, R) be a short presentation as obtained by Theorem 2, for a solvable group G with n elements. There is a polynomial time-bounded algorithm that on input (X, R) constructs the Cayley table of a group G' that is isomorphic to G.*

A ring can be encoded in a similar way as a group providing explicitly the tables for its additive and multiplicative groups. But these structures admit also succinct representations. A ring R can be encoded by giving first a short presentation of its additive group in terms of generators and relations (this is always possible since the additive group is Abelian), and then giving the multiplication by expressing the product of each pair of generators as a linear combination of the generators. $(R, +, \cdot) = \langle d_1, d_2, \ldots, d_m, A^1, \ldots A^m \rangle$. This expression represents that R is generated by m elements $a_1, \ldots, a_m$, each a_i being of additive order d_i. $(R, +) = \langle a_1 \rangle \oplus \langle a_2 \rangle \oplus \ldots \langle a_m \rangle$ and for each i, j $a_i \cdot a_j = \sum_{k=1}^{m} A_{ij}^k a_k$. Again by the Structure Theorem for Abelian groups, the values of the $d_i's$ are prime powers, and they are unique up to permutations. Observe that the size of the representation is polylogarithmic in the number of elements in R. We will refer to this representation as the *basis representation* of the ring, as opposed to the *table representation* mentioned above. The ring isomorphism problem is defined [11] as

$$\text{RING-ISO} = \{(R_1, R_2) \mid R_1, R_2 \text{ are isomorphic rings in basis representation}\}$$

An homomorphism between rings given in basis representation can also be expressed in a succinct form. If R_1 and R_2 are two rings given by their additive generators $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$, a homomorphism $\varphi : R_1 \to R_2$ can be expressed by the images of $a_1, \ldots, a_m$, and these can be represented by an $m \times m$ matrix F such that

$$\varphi(a_i) = \sum_{j=1}^{m} F_{ij} b_j.$$

The homomorphism is considered to extend linearly to all the elements. In order to test whether φ is an isomorphism, it has to be verified that for every generator pair $\varphi(a_i \cdot a_j) = \varphi(a_i)\varphi(a_j)$, and that φ does not map any non-zero element of R_1 to zero. This last condition can be tested by solving a system of linear equations. Since an isomorphism can be encoded by a string of polynomial length with respect to the ring representation, and its correctness can be tested in polynomial time, it follows:

Theorem 4. *[11]* RING-ISO $\in$ NP.

3 AM Protocols

Denote by $\mathrm{AM}(r(n), s(n))$ the class of languages accepted by 2-round AM protocols, with error probability 1/4, in which Arthur uses $O(r(n))$ random bits and Merlin uses $O(s(n))$ nondeterministic bits. Formally, a language L is in $\mathrm{AM}(r(n), s(n))$ if there is a set $B \in \mathrm{P}$ such that for all x, $|x| = n$,

$$x \in A \Rightarrow \mathrm{Prob}_{w \in_R \{0,1\}^{r'(n)}}[\exists y, |y| = s'(n) : \langle x, y, w\rangle \in B] \geq 3/4,$$
$$x \notin A \Rightarrow \mathrm{Prob}_{w \in_R \{0,1\}^{r'(n)}}[\forall y, |y| = s'(n) : \langle x, y, w\rangle \in B] \leq 1/4,$$

where r' and s' are functions in $O(r(n))$ and $O(s(n))$ respectively. Notice that the above definition is equivalent to the definition in terms of 2-round Arthur-Merlin protocols. Indeed, the standard AM class is $\mathrm{AM}(n^{O(1)}, n^{O(1)})$.

The class AM is very well suited to deal with isomorphism problems. The fact that ring non-isomorphism belongs to this class is not surprising. The protocol is similar to the one for graph non-isomorphism [4]. In fact all the protocols presented here have the same flavor, they boil down to construct a set related to the given structures with the property that the number of elements in the set is small in case the structures are isomorphic, and large otherwise.

Theorem 5. *[11]* RING-ISO $\in$ coAM

Proof. We give an AM protocol for non-isomorphism. On input two rings R_1, R_2 in basis form, Arthur can first check whether their additive groups $(R_1, +)$ and $(R_2, +)$ are isomorphic. This can be done using the fact that the basis representation for the additive group is unique. If the additive groups are non-isomorphic, then Arthur accepts. Otherwise let $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$ be the generators for $(R_1, +)$ and $(R_2, +)$ and consider the set

$$\begin{aligned} C(R_1) := \{\langle A^1, \ldots A^m, A_\phi\rangle \mid & \text{ there is an automorphism } \pi \text{ of } (R_1, +) \\ & \text{such that } \pi(a_i)\pi(a_j) = \sum\nolimits_k A_{ij}^k \text{ and } A_\phi \text{ is a matrix describing} \\ & \text{an automorphism } \phi \text{ of } (R_1, +) \text{ expressed on the basis } \{\pi(a_i)\}\} \end{aligned}$$

and define $G(R_2)$ in a similar way. It is not hard to see that $C(R_1)$ and $C(R_2)$ coincide in case $R_1 \cong R_2$ while the set are disjoint if the rings are non isomorphic. Also, the cardinality of $C(R_i)$ is exactly s, the number of automorphisms in the additive group $(R_i, +)$, which can be computed in polynomial time. A standard AM protocol using hash functions can distinguish between $C(R_1) \cup C(R_2)$ having cardinality s or $2s$.

GROUP-ISO is reducible to RING-ISO and therefore the above result also shows that GROUP-ISO $\in$ coAM. But in case of GROUP-ISO we can do much better. We present a two-round AM protocol for group non-isomorphism from [3] that has constant success probability, and Arthur uses $O(\log^6 n)$ random bits and Merlin uses $O(\log^2 n)$ nondeterministic bits. Thus group non-isomorphism is in $\mathrm{AM}(\log^6 n, \log^2 n)$.

Let G be a group with n elements. A sequence of k group elements $X = (g_1, \ldots, g_k)$ is called a *cube generating k-sequence* for G if

$$G = \{g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k} \mid \epsilon_i \in \{0,1\}\}.$$

The set $\{g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k} \mid \epsilon_i \in \{0,1\}\}$ is the *cube* $\mathrm{Cube}(X)$ generated by the sequence X. Erdős and Renyi [9] proved the following important theorem about the probability that $\mathrm{Cube}(X) = G$, for a k-sequence X chosen uniformly at random from G^k.

Theorem 6. [9] *Let G be a finite group with n elements. For $k \geq \log n + \log\log n + 2\log(1/\delta) + 5$,*

$$\mathrm{Prob}_{X \in_R G^k}[X \textit{ is a cube generating sequence for } G] > 1 - \delta.$$

For G with n elements we choose $k = 4\log n$ and obtain:

Corollary 7. *Let G be a finite group with n elements and $k = 4\log n$. Then*

$$\mathrm{Prob}_{X \in_R G^k}[u \textit{ is a cube generating sequence for } G] > 1 - 1/n.$$

Now, let G be a group with n elements and $X = (g_1, g_2, \ldots, g_k)$ be a cube generating sequence for G. There is a natural 1-1 mapping $\pi_X : G \to \{0,1\}^k$ that is defined by the cube generating sequence X. The mapping π_X is defined as follows: $\pi_X(g) = (\epsilon_1, \epsilon_2, \ldots, \epsilon_k)$, where $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k)$ is the lexicographically first k-tuple in $\{0,1\}^k$ such that $g = g_1^{\epsilon_1} g_2^{\epsilon_2} \cdots g_k^{\epsilon_k}$. Clearly π_X is an injective mapping and, given the Cayley table for G as input, π_X can be computed in polynomial time.

Lemma 8. *Let G be a group with n elements given by its Cayley table, and X is a cube generating k-sequence for G. There is a polynomial (in n) time procedure $\mathcal{B}$ that takes as input the pair (X, G) and outputs a pair (Y, H), where H is the Cayley table of a group defined on the set $\pi_X(G) \subseteq \{0,1\}^k$ and $Y = \pi_X(X)$ is a cube generating sequence for H.*

The following proposition is an important property of $\mathcal{B}$.

Proposition 9. *Let G_1 and G_2 be groups of order n and ϕ be an isomorphism from G_1 to G_2. Let X be a cube generating k-sequence of G_1. Then $\mathcal{B}(X, G_1) = (Y, H)$ implies $\mathcal{B}(\phi(X), G_2) = (Y, H)$.*

Now, for a group G with n elements we define the following set.

$$C(G) = \{(S, H, \psi) \mid \text{ there is a cube generating } 4\log n\text{-sequence } X \subset G \text{ such that } \mathcal{B}(X, G) = (S, H) \text{ and } \psi \in Aut(H)\}.$$

Lemma 10. *If G_1 and G_2 are isomorphic finite groups then $C(G_1) = C(G_2)$ and if G_1 and G_2 are non-isomorphic then $C(G_1) \cap C(G_2) = \emptyset$.*

Let $C(G_1, G_2) = C(G_1) \cup C(G_2)$. The size of $C(G_1, G_2)$ in the two cases $G_1 \cong G_2$ and $G_1 \ncong G_2$ can be estimated as:

Lemma 11. *For a group G with n elements*

$$(1 - 1/n)n^{4 \log n} \leq |C(G)| \leq n^{4 \log n}.$$

We have the immediate corollary which is crucial to the AM protocol.

Corollary 12. *Let G_1 and G_2 be groups of n elements. For $n > 4$ we have:*

1. *$G_1 \cong G_2$ implies $|C(G_1, G_2)| \leq n^{4 \log n}$.*
2. *$G_1 \ncong G_2$ implies $|C(G_1, G_2)| > 1.5n^{4 \log n}$.*

A standard AM protocol could distinguish using hash functions the cardinality of $C(G_1, G_2)$ in the two cases. However, since we we want to save on the use of random and nondeterministic bits, more careful arguments are needed. The idea is to consider the elements of $C(G_1, G_2)$ as numbers, and do the computations modulo a random prime number p of $\log^3 n$ bits. In order to approximate the size of $C(G_1, G_2)$ we estimate the number of strings in this set (modulo p) that are mapped to 0 by a randomly chosen linear transformation of small size. The AM($\log^6 n, \log^2 n$) protocol in given below. We refer the reader to [3] for a detailed analysis of the protocol.

Arthur Randomly sample numbers of $\log^3 n$ bits until a prime number p is found. If after $5 \log n$ trials no prime number has been found, then reject the input. Otherwise pick a random $\mathbb{F}_2$-linear function $h : \Sigma^t \to \Sigma^k$, where $t = \log^3 n$ and $k = \log(4 \log^2 n)$. Send p and h to Merlin.

Merlin Sends back two 7-tuples $\langle x_1, \ldots, x_7 \rangle$ and $\langle i_1, \ldots, i_7 \rangle$. Where for $j = 1, \ldots, 7$, $i_j \in \{1, 2\}$ and $x_j = (S_j, T_j, \psi_j)$, where S_j is a sequence of elements of $\{0,1\}^{4 \log n}$ of length $4 \log n$, T_j is a cube generating $4 \log n$-sequence for the group G_{i_j}, and $\psi_j : S_j \to \{0,1\}^{4 \log n}$ is a 1-1 mapping.

Arthur For each $j = 1, \ldots, 7$, Arthur does the following verification, all in polynomial time: Let $x_j = (S_j, T_j, \psi_j)$. Computes $\mathcal{B}(T_j, G_{i_j}) = (S, H_j)$ and verifies that $S = S_j$. Then using the group multiplication of H_j, Arthur extends ψ_j to all of H_j and verifies that it is an automorphism of H_j. Now, let $y_j = (S_j, H_j, \psi_j)$ for $1 \leq j \leq 7$ and let $y = \langle y_1, \ldots, y_7 \rangle$ which is an element of X. Verify that $h(\text{num}(y) \ (mod p)) = 0^k$ and if so, accept the pair (G_1, G_2) as non-isomorphic.

It follows:

Theorem 13. *There is a 2-round AM protocol with error probability $1/4$ for the Group Non-isomorphism problem in which Arthur uses $O(\log^6 n)$ random bits and Merlin uses $O(\log^2 n)$ nondeterministic bits. Hence, the problem is in* AM($\log^6 n, \log^2 n$).

There is a similar AM protocol using only a polylogarithmic number of random and nondeterministic bits for ring non-isomorphism in the case the input rings are given explicitly by their operation tables.

4 Derandomization

As mentioned in the introduction, in the last years several authors [2, 12, 16] have tried to derandomize the class AM to NP by using hardness assumptions. The assumptions state that some high complexity class requires exponential size circuits of a certain kind. These results derandomize the whole class AM. As an immediate consequence of of [12] and Theorem 5 we can state for example:

Corollary 14. *If there is a function in NE ∩ coNE requiring boolean circuits with oracle gates for Satisfiability of size* $2^{\Omega(n)}$ *then* RING-ISO ∈ NP ∩ coNP.

We present here results from [3] showing a way to use a specific property of the group isomorphism problem in order to derandomize its AM protocol under weaker hypothesis. These methods work for the case of groups that have short presentations that can be efficiently computed from the Cayley table of the group. This is for example the case of solvable groups. We assume throughout this section that the input instances (G_1, G_2) for GROUP-ISO are such that G_1 and G_2 are solvable groups. Given an instance (G_1, G_2), by applying the algorithm of Theorem 2 and then the algorithm of Theorem 3 to both G_1 and G_2 we can obtain a new pair of groups (G_1', G_2') such that $G_1 \cong G_2$ if and only if $G_1' \cong G_2'$. We call such an instance (G_1', G_2') a *reduced instance* of GROUP-ISO. The key observation is that there is a constant c such that the number of reduced instances (G_1', G_2') for pairs of graphs with n elements is bounded by $2^{\log^c n}$. This is immediate from the bound on the size of short presentations for solvable groups given in Theorem 2.

Lemma 15. *The number of reduced instances* (G_1', G_2') *is bounded by* $2^{\log^c n}$ *for a fixed constant* $c > 0$*, where* G_1' *and* G_2' *are groups with* n *elements.*

For a first derandomization of the AM protocol for GROUP-NONISO we give an easy generalization of a theorem from the Goldreich and Wigderson paper [10–Theorem 3] for a nondeterministic setting. The idea is to try and derandomize certain advice-taking randomized algorithms by extracting randomness from the input. It can be proved almost exactly as [10–Theorem 3].

Theorem 16. *Let* M *be an advice-taking* NP *machine for a problem* Π*, where the length of the advice is bounded by* $\log^c m$ *for some constant* c*, for inputs* $x \in \{0,1\}^m$*. Suppose it holds that at least* $2/3$ *fraction of the* $\log^c m$*-bit advice strings are good advice strings. More precisely*

$$\mathrm{Prob}_{w \in \{0,1\}^{\log^c m}}[\forall x \in \{0,1\}^m \text{ it holds that } M(x,w) \text{ is correct}] \geq 2/3.$$

Then for every $\epsilon > 1$*, there is an NP machine* M' *for* Π *that is incorrect on at most* $2^{\log^{c\epsilon} m}$ *inputs* $x \in \{0,1\}^m$.

In order to be able to use this result we have to transform the AM protocol for GROUP-NONISO of Section 3 into an advice taking NP machine (with short advice) for the problem. The standard amplification of the success probability

of the AM protocol would not work since the resulting advice string would be of polynomial length. We show how the AM protocol can be modified in order to avoid this problem.

Fix a standard encoding of an instance (G_1, G_2) of groups of with n elements by a binary string of length $m = Cn^2\lceil \log n \rceil$, where C is some fixed constant. Furthermore, we can assume that both this encoding and its inverse are computable in time polynomial in n. . We can assume that the AM protocol Section 3 takes as input a string $x \in \{0,1\}^m$ and first checks if it encodes an instance (G_1, G_2) of solvable groups and rejects if it does not. We can think of the binary strings x as the input to the AM protocol.

The AM protocol of Section 3 is modified as follows: on input $x \in \{0,1\}^m$, first Arthur decodes x to get (G_1, G_2) and checks that G_1 and G_2 are solvable groups. Then, applying the algorithms of Theorems 2 and 3 in succession, Arthur converts (G_1, G_2) to a reduced instance (G_1', G_2'). Now, Arthur starts the AM protocol *for the reduced instance* (G_1', G_2'). Merlin is also supposed to compute (G_1', G_2') and execute his part of the protocol for (G_1', G_2'). Observe that by Lemma 15 there are only $2^{\log^c n}$ reduced instances for a fixed constant $c > 0$. Notice that effectively the original AM protocol is now being applied only to reduced instances. By standard methods of amplifying the success probability of the AM protocol, we can convert the AM protocol to a ($\log^{O(1)} n$ size) advice taking NP machine M. We summarize the above observation as a theorem.

Theorem 17. *There is an (*$\log^{(O(1)} n$ *size) advice-taking* NP *machine* M *for* GROUP-NONISO *such that that for inputs* $x \in \{0,1\}^m$ *the following holds:*

$$\text{Prob}_{w \in \{0,1\}^{\log^c n}}[\forall x \in \{0,1\}^m \textit{ it holds that } M(x,w) \textit{ is correct}] \geq 2/3.$$

The AM protocol can be transformed using well-known techniques into a one-sided error protocol for GROUP-NONISO such that when the input groups are non-isomorphic, the protocol accepts with probability 1, where the protocol still uses only a polylogarithmic number of random bits. Consequently, the advice-taking NP machine M defined above also has only one-sided error.

Now, applying Theorems 17 and 16 we immediately have the following consequence for GROUP-ISO in the case of solvable groups.

Theorem 18. *For some constant* $c > 1$ *there is an* NP $\cap$ coNP *machine* M *for* GROUP-ISO *for solvable groups that is incorrect on at most* $2^{\log^c m}$ *inputs* $x \in \{0,1\}^m$ *for every* m.

The proof follows by combining the standard NP machine for GROUP-ISO with the NP machine M' for GROUP-NONISO given by Theorem 16. Observe that M' may be incorrect only when its input is a pair of isomorphic groups.

We show a second derandomization result applying the Nisan-Wigderson pseudorandom generator. We prove that GROUP-ISO for solvable groups is in NP $\cap$ coNP assuming EXP $\not\subseteq$ i.o$-$PSPACE/poly[1]. EXP $\subseteq$ i.o$-$PSPACE/poly

[1] A language L is in i.o$-$PSPACE/poly if there is a PSPACE machine that takes polynomial-size advice and is correct on L for infinitely many input lengths.

implies EXP $\subseteq$ i.o−PSPACE and therefore the result holds also under the uniform hardness assumption EXP $\not\subseteq$ i.o−PSPACE.

Theorem 19. *If* EXP $\not\subseteq$ i.o−PSPACE/poly *then* GROUP-ISO *for the case of solvable groups is in* NP $\cap$ coNP.

Proof. (Sketch) In order to derandomize the AM protocol for GROUP-NONISO, it suffices to build a pseudorandom generator that stretches $O(\log n)$ random bits to $O(\log^6 n)$ random bits, such that the pseudorandom string of $O(\log^6 n)$ bits cannot be distinguished from a truly random string of the same length by the protocol. The AM protocol for GROUP-NONISO can be transformed in the following way, substituting Merlin's computation by a query to oracle L.

1. On input G_1 and G_2, Arthur computes their short presentations (Y_1, R_1) and (Y_2, R_2).
2. Uniformly at random, Arthur picks a hash function h and a prime number p using $O(\log^6 n)$ random bits.
3. Arthur queries an oracle L for $((Y_1, R_1), (Y_2, R_2), h, p)$ and accepts if the string is in L.

L is a set of short representations of groups G_1, G_2 with the property that there is some element in $C(G_1, G_2)$ that is mapped by h to 0^k, thus showing that the set $C(G_1, G_2)$ is large. L can be computed within polylogarithmic space in the size of the input groups, n, On all inputs (G_1, G_2) the above sketched computation has the same acceptance probability as the AM protocol.

We shall use the Nisan-Wigderson pseudorandom generator [17], that stretches $O(\log n)$ random bits to $O(\log^6 n)$ random bits using an EXP complete language as the hard function. This is done by constructing from a hard boolean function $g : \{0,1\}^m \to \{0,1\}$ a new one $g_D : \{0,1\}^l \to \{0,1\}^r$ for suitable values of l, m and r, so that the output of g_D looks random to a small deterministic circuit, provided g is hard to approximate by deterministic circuits of a certain size.

For a set A let $\mathcal{CIR}^A(n, s)$ stand for the set of n-input boolean functions that can be computed by deterministic circuits of size at most s, having besides the normal gates oracle gates evaluating the characteristic function of A.

We state a crucial lemma due to Nisan and Wigderson [17].

Lemma 20. [17] *Let $g : \{0,1\}^m \to \{0,1\}$ be an $\mathcal{CIR}^A(m, r^2 + r2^k)$-hard function (for some oracle A). Then the function g_D has the property that for every r-input circuit c of size at most r^2,*

$$\left|\mathrm{Prob}_{y \in_R \{0,1\}^r}[c^A(y) = 1] - \mathrm{Prob}_{s \in_R \{0,1\}^l}[c^A(g_D(s)) = 1]\right| \leq 1/r.$$

In the AM protocol, the random bits are used in Phase 3 and the computation of this phase can be simulated by a polylog(n) size circuit *with oracle* L that takes as input the short presentations (Y_1, R_1) and (Y_2, R_2) and the random bits h and p.

Now, let g be the characteristic function of an EXP-complete language in E. Furthermore, let the language L defined above be the oracle A of Lemma 20. As already explained, the computation in Phase 3 can be carried out by a polylog(n) size circuit with L as oracle.

We claim that the derandomization is correct on all but finitely many inputs. Suppose not. In particular, suppose the derandomization of the AM protocol fails for some input pair (G_1, G_2) of solvable groups. Let (Y_1, R_1) and (Y_2, R_2) be their short presentations computed in Phase 1. As a consequence of the failure of the derandomization, it follows that the polylog(n) size circuit with oracle L for Phase 3 with input fixed as (Y_1, R_1) and (Y_2, R_2) is a distinguisher circuit that distinguishes between the output of g_D and the truly random bits. Applying Yao's method as explained in [17], we can convert the distinguisher into a next bit predictor, and finally obtain a polylog(n) size circuit with oracle L that computes g correctly on a $1/2 + 1/\log^{O(1)} n$ fraction of $O(\log n)$ size inputs. Notice here that we are using the fact that g_D can be computed in time polynomial in $\log n$.

By applying the methods of [6], we can use Yao's XOR lemma and the fact that EXP has random-self reducible complete sets to conclude that an EXP-complete set can be correctly computed on all $\log n$ size inputs by a polylog(n) size circuit with oracle L. This is true for infinitely many input lengths $\log n$ (since we assumed that the derandomization fails for infinitely many inputs). But that implies EXP $\subseteq$ i.o$-$PSPACE/poly, contradicting the hardness assumption.

5 Conclusions

We have considered the isomorphism problem for groups and rings encoded in different ways. While in GROUP-ISO the inputs include explicitly the group operation table, in the version of RING-ISO considered here [11] the input is a succinct representation of the rings. Both problems are related to graph isomorphism, one being easier and the other probably harder. These are only two of the many possibilities for encoding the inputs. It is well known that in general the complexity of the problem increases when the input is given is a more succinct way. For example, for an "easier" version of RING-ISO with the rings given in the table representation, the results presented here for GROUP-ISO also hold. Also, group isomorphism becomes harder with a succinct group encoding. For example, GRAPH-ISO is reducible to permutation group isomorphism, when the groups are given by generator sets [14]. But in all the mentioned examples, the complexity of the problems does not change too much: the succinct representations are in NP, while the explicit versions are not known to be in P. The number of random and nondeterministic bits in Arthur-Merlin protocols for non-isomorphism provide a good setting to measure the relative complexity of these problems. Improving upper bounds for the isomorphism problems that seem easier that GRAPH-ISO and improving the lower bounds for the harder problems (in terms of completeness for complexity classes [19], for example) might be a way to better understand the complexity of GRAPH-ISO. A related question is

the study of the complexity of GRAPH-ISO when the input graphs are given in a succinct form.

References

1. M. Agrawal, N. Kayal and N. Saxena, PRIMES is in P. Preprint, August 4, 2002, `http://www.cse.iitk.ac.in/users/manindra/`.
2. V. Arvind and J. Köbler, On resource bounded measure and pseudorandomness, *Proc. 17th FSTT Conference* Lecture Notes in Computer Science 1346 Springer Verlag, 235–249, (1997).
3. V. Arvind and J. Torán, Solvable Group Isomorphism is (almost) in NP ∩ coNP, *Proc. 19th IEEE Conference on Computational Complexity*, 91–103, (2004).
4. L. Babai, Trading group theory for randomness, *Proc. 17th ACM Symposium on Theory of Computing*, 421–429, 1985.
5. L. Babai, Bounded round interactive proofs in finite groups, *SIAM J. Disc. Math.*, 5, 1, 88–111, 1992.
6. L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, BPP has subexponential simulations unless EXPTIME has publishable proofs, *Computational Complexity,* 3, pp. 307–318, 1993.
7. L. Babai, A.J. Goodman, W. Kantor, E. Luks and P.P. Pálfy, Short presentation for finite groups, in *Journal of Algebra*, 194, 79–112, 1997.
8. H. Coxeter and W. Moser *Generators and Relations for Discrete Groups*, Springer Verlag, (1980). 127–138.
9. P. Erdős and A. Rényi, Probabilistic Methods in group theory, *Jour. Analyse Mathématique*, vol. 14, (1965), 127–138.
10. O. Goldreich and A. Wigderson, Derandomizing that is rarely wrong from short advice that is typically good, in *Proc. 6th RANDOM workshop* Lecture Notes in Computer Science 2483, 2002, 209–223.
11. N. Kayal and N. Saxena On the ring isomorphism and automorphism problems, in *Electronic Colloquium on Computational Complexity*, TR04–109, 2002.
12. A. Klivans and D. van Melkebeek, Graph Isomorphism has subexponential size provers unless the polynomial time hierarchy collapses. *In Proc. 31st ACM STOC*, 1999, 659–667.
13. R.J. Lipton, L. Snyder and Y. Zalcstein The complexity of word and isomorphism problems for finite groups. Johns Hopkins University 1976.
14. E. Luks, Permutation groups and polynomial time computations, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vo1 11, 139–175, 1993.
15. G.L. Miller, On the $n^{\log n}$ isomorphism technique, in *Proc. 10th ACM Symposium on the Theory of Computing*, 1978, 51–58.
16. P.B. Miltersen and N. Vinodchandran, Derandomizing Arthur-Merlin games using hitting sets, in *Proc. 40th IEEE Symposium on Foundations of Computer Science,* 1999, 71–80.
17. N. Nisan and A. Wigderson, Hardness vs randomness, in *Journal of Computer and System Sciences,* 49:149–167, 1994.
18. R. Shaltiel and C. Umans, Pseudorandomness for Approximate Counting and Sampling *Electronic Colloquium on Computational Complexity* TR04–86, 2004.
19. J. Torán, On the hardness of graph isomorphism, In *SIAM J. Comput.* 33, 5, 1093–1108 (2004).

Beyond the Super-Turing Snare: Analog Computation and Digital Virtuality

Giuseppe Trautteur*

Dipartimento di Scienze Fisiche, Università di Napoli Federico II,
Complesso Universitario Monte S. Angelo,
I-80126 Napoli, Italy
trau@na.infn.it

Abstract. Conceptual difficulties involved in attempts to build analog super-Turing sources are examined in the light of epistemic problems of physical measurement. Basic concepts of computability theory are appealed to in order to set up a procedural comparison between analog and digital computations which is not affected by similar conceptual difficulties: Is the interpretation of program code within the capacities of some analog computer? The epistemological significance of this "virtuality" problem is emphasized by reference to explanation requests arising in the cognitive neurosciences.

1 Analog, Digital, and Hybrid Computations

Analog computation is computation carried out by apparatuses whose action is described by means of continuous variables, that is, variables whose values can be put in correspondence with some interval of the real continuum. An analog computation may be regarded as a continuous trajectory in the appropriate state space of the computing device.

The origin of the term 'analog' is rooted in methods that were initially adopted to implement analog computers. Electronic, mechanical, fluidic, etc. devices were connected in such a way that the functional description of the resulting apparatus A *coincided* with or approximated a differential system T to be integrated. Often, but by no means in principle, system T models the behaviour of some other physical system P, so that A and P are both described by T. Historically, the system A was built as an *analogue* of some physical system P, hence the name. And indeed analog computers have been used for design purposes in the simulation of systems that are too complex and difficult to experiment with in real world situations (going as far back in time as the tide prediction systems built by Lord Kelvin out of springs, levers, and other mechanical components). Moreover, analog computers have been used as direct controlling agents for some external system (e.g.: automatic piloting, avionics,

* This paper would not have been written without the encouragement of Guglielmo Tamburrini.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 507–514, 2005.

plant control, etc.). But in the end, analog computers are to be regarded as *computing* devices performing symbol processing, insofar as they are used to find solutions to numerical problems usually associated with differential equations. In analog computers, numerical values can be extracted by *measuring* the value of some physical quantity, typically a voltage. From the viewpoint of this intended use, analog computers are on a par with present-day software packages like MATLAB, Mathematica, Maple V, and so on. Nevertheless, the computational processes carried out by analog computers significantly differ from the computational processes carried out by other kinds of computational systems, whose description requires no essential appeal to continuous variables. Let's see.

Digital computation is computation carried out by an apparatus D whose action is described in purely discrete terms: the progress of any computation by D is given by a sequence $s_0, s_1, \ldots$, of states, where each s_j is adequately described in finite terms by means of discrete variables.

We take a variable to be discrete or quantized (where 'quantized' has no implication of a quantum mechanical treatment of the physical system supporting the computation) when: (i) it takes real values within mutually disjoint intervals that are separated by non-zero measure intervals; (ii) the particular value taken on by the variables within an interval is of no empirical or theoretical consequence,[1] and (iii) the intervals can be put in correspondence with a countable set.

In the above definition of a discrete variable, condition (ii) expresses an epistemic requirement, warranting the use of a discrete variable in the description of some given system S, insofar as nothing in the way of empirical or theoretical adequacy is lost by substituting the discrete variable for a continuous one. Turing [11–1950] seems to advance a similar epistemic motivation when he remarks: "there are many kinds of machine which *can profitably be thought of* as being discrete state machines".[2] Arguably, the advantage envisaged by Turing is simplicity without loss of significant information relative to some context of inquiry fixed by particular explanatory or predictive needs.

Assuming that analog devices and digital ones are distinct kinds of devices, one may explore devices whose parts are either analog (A) or digital (D). Let us call such devices hybrid AD devices.

[1] This condition restricts the nature of the dynamical system supporting the computation in the following sense: the trajectories in the state space of the system must fulfil a stability condition such that for every interval in which the state takes value it must be the case that all trajectories issuing from states within that interval (taking all the points in the interval as initial states) must give rise to trajectories which subsequently visit the same intervals.

[2] Cp. [11–p. 439, my emphasis]. A similar point is made in Turing's 1948 report entitled *Intelligent Machinery*: "We may call a machine 'discrete' when it is natural to describe its possible states as a discrete set, the motion of the machine occurring by jumping from one state to another. The states of 'continuous' machinery on the other hand form a continuous manifold and the behaviour of the machine is described by a curve on this manifold."

Hybrid analog-digital computation is computation carried out by some *AD* apparatus, whose action is described by means of both continuous and discrete variables. A particular kind of *AD* apparatus is one whereby one process in which continuous variables occur is squeezed in between two processes that are described in purely discrete terms (from now on, a *DAD* apparatus).

2 Analog Computations Beyond Turing Computations?

In the light of the above definitions, Turing machine computations are, as expected, digital computations. The notion of a Turing machine computation embodies distinctive idealizations that are needed to make sense of the idea that Turing machines determine the values of number-theoretic functions: finite, but not *a priori* bounded resources, usually space (locations of tape) or time (number of computational steps), are to be made available to carry on a generic Turing machine computation. Distinctive idealizations are also involved in the notion of an analog computable function in general, and in (the decidedly more problematic) endeavours to envisage analog computable functions that are not Turing computable[3] in particular. In this section, we examine ontological and epistemological problems raised by these idealizations. As an entry point, let us consider John Myhill's extensive discussion [6–1963], Myhill regarded as the more promising suggestion for constructing such a machine out of conventional analog equipment (chiefly comprising adders, multipliers, and integrators). However, he was careful to note that some idealizations are required "in order to make Scarpellini's work a basis for constructing an actual computer which can solve problems which are not digitally (= recursively) solvable." [4–p. 12]. Out of the four idealizations he formulated, the following he deemed to be indispensable:

1. The perfect functioning of analog components.
2. The existence of a perfect sensing mechanism - more specifically, the existence of a perfect zero sensor, i.e., a discontinuous physical device.

Let us distinguish between three senses of Id. 1.

The first one is "tautological": it just means that the analog components (physical entities) do behave as physical entities, that is, behave according to whatever physical laws govern their behaviour.

The second sense is conformity to the blueprint. Imperfection, in this case, is perfect functioning, but with respect to some different blueprint.

The third sense is absence of any random perturbation, essentially thermal noise. The environment interacts with the system in uncontrollable ways.

Let us consider the content of Id. 1 in the light of epistemic problems of physical measurement. Id. 1 embodies an ontological assumption which is consistent

[3] It has been a long time now since the search for digital "non-Turing" machines, i.e. digital machines with stronger, more inclusive processing power, has been abandoned.

with a traditional picture of what happens in real-world measurement processes of physical quantities. The physical quantities in terms of which one describes the behaviour of some analog device, independently of its "perfection" or "imperfection" (see above senses of Id. 1), are nonetheless assumed to take on exact values within a continuous range. The relations between a physical entity, its dimensions, and its value stand in need of deeper understanding even though scientific progress seem unaffected by (and therefore oblivious to) this relative lack of clarity. In particular, when one says, as we do above, that a physical quantity assumes an exact value, it is not clear whether we are assuming that that physical entity embodies an actual infinity.

This thorny problem can be swept under the carpet, although by no means solved, in the digital treatment as well as in the measurement problem when one takes as outcome of the measurement only rational numbers—in fact integers. But in the case of an analog computational device, super-Turing power is usually sought for by an essential appeal to the assumption that physical entities possess that character of infinite precision that real numbers do.

The discrepancy between this ontological assumption and measurement operations manifests itself in the course of repeated measurements of the same physical quantity, for thermal effects (see third meaning of Id. 1) and other disturbing factors (by which we mean, in accordance with the second meaning of Id. 1, the exact compliance of the given analog device with another blueprint than the one intended for the computation) will change in unknown ways the successive measurement's outcomes. Since Id. 1, in senses 2-3, never obtains in real-world measurements, a series of such measurements will end up into a *population* of different measured values. Therefore, we only have an effective access to physical quantities through statistical populations.

Notice that the unavoidable occurrence of a final measurement in any analog computational process rules out the existence of pure A devices. Indeed the measurement operation maps the possibly infinitely precise value of the analog variable onto a discrete one. Therefore only AD devices are possible and, when an input must be additionally specified, only DAD devices are possible. All of this, and in particular the digitality of the outcome of a measurement process, is a consequence of the fact that intersubjective knowledge is symbolic, and therefore finite and discrete, whatever else 'symbolic' means.

Notice, however, that the digital character of intersubjective communication and information processing does not exclude that conscious understanding and inner processing depend in an ineffable way on the infinitely precise values of some physical quantity of the brain. For example feelings in Damasio's sense [2–p. 36/38], i.e. experience directly associated with the inner operation of the brain, might be expressed/supported/caused—however far-fetched this may appear—by analog physical quantities of the nervous system.

Turning to Id. 2 we notice that it cannot be asserted without assuming Id. 1: an exact zero detection instrument would not make sense in the absence of exact values of physical quantities. However, it might well be the case that physical

quantities assume exact values even though this fact is not physically detectable (and thus Id. 1 can be asserted without implying Id. 2).

The remarks above make it very implausible that one can build on Scarpellini's work toward an effective, analog, non-recursive computation, because the measurement process involved in Scarpellini's proposal needs infinite precision in evaluating analogically certain integrals and discriminating between the values $a = 0$, assumed by some continuous physical variable described by those integrals, and every other value $a > 0$, no matter how small is the difference between a and 0. Id. 1 and Id. 2 are necessary for such use of Scarpellini's work, and they are implausible to the point of physical impossibility.

A different framework which suggests the possibility of analog super-Turing sources is provided in [8]. Let us consider Siegelmann's work in the light of Id. 1-2. The focus of this work is on artificial neural networks (*ANN*s), even though other variants of analog computation are taken into account and carefully compared with results on *ANN*s. These include recurrent networks with rational or real states and weights. Activation functions considered are mostly piece-wise linear, and thus continuous, functions.

While comparing such *ANN*s with Turing machines, various equivalences and inclusions are proved along the way. In particular, Siegelmann defines a class of *ANN*s with real weights which is shown to include *all* discrete languages: the real weights are used there as information enabling one to decide every set of natural numbers. However, this capability depends, from a mathematical point of view, on trivial cardinality reasons.[4] Siegelmann admits that any physical model for those *ANN*s would still involve something like Id. 1 for the real weights to work as required. She writes:

> In nature, the fact that the constants are not known to us, or cannot even be measured, is irrelevant for the true evolution of the system. For example, the planets revolve according to the exact values of G, p, and their masses. [8–p. 59].

A quote that nicely upholds the first meaning of Id. 1. Indeed, no measurement is involved (no need for third meaning of Id. 1) nor the "true" blueprint is identified (no need for second meaning of Id. 1).

However, it seems that Siegelmann and Sontag [9] are able to bypass Id. 2 for some forms of analog neural computation through a very clever (4-Cantor) encoding scheme for neural weights and states that allows for finitely distinguishable, i.e. discrete, output values even with piecewise linear discrimination functions. The scheme certainly works for recursively computable processes, which need only the power of the rationals. Briefly, the precision required for a process is finite, but is allowed to increase unboundedly. This is on a par with the usual idealization for Turing machine resources mentioned above. As Siegelmann and Sontag follow a different route in their treatment of neural computations with real weights, it is not obvious that Id.2 is bypassed in that case too.

[4] Cp. [3–sect. 4].

In any case the epistemological and ontological problems emphasized in Myhill's reflections about the possibility of building analog super-Turing sources apply to Siegelmann's work, and are readily brought to bear on state-of-the-art work on analog super-Turing sources, for none of the approaches taken into account in recent comprehensive reviews (Copeland [1], Stannett [10]) seems to suggest a promising strategy to cope with both Id.1-2.

3 Analog Computation and Digital Virtuality

Conjectures about non-recursive behaviours allegedly implemented in the brain have provided broad epistemological motivations for investigating analog super-Turing sources. However, technical developments from Scarpellini's result up to the present day add no positive reason to support these speculations, while the philosophical reflections above suggest that these speculations give rise to vexing epistemological and ontological problems. Scarpellini's [7] retrospective comments in 2003 on the possible import of his 1963 result on the understanding of brain mechanisms parallel, although with a less pessimistic outlook, the remarks we made above on a far-fetched hypothesis about the experience of feelings according to Damasio:

> ... it does not seem unreasonable to suggest that the brain may rely on analogue processes for certain types of computation and decision-making. Possible candidates which may give rise to such processes are the axons of nerve cells... it is conceivable that the mathematics of a collection of axons may lead to undecidable propositions like those discussed in my paper.

Similar epistemological motivations (i.e. understanding brain information processing) can be adduced to investigate additional problems which (a) concern the relationship between analog and digital computation, (b) are expressed by means of basic concepts in computability theory, *but* (c) have nothing to do with the question whether analog machines can solve classes of problems that a universal Turing machine cannot solve. One such problem is suggested by the very notion of universal Turing machine in general, and by the effective coding of programs involved in Turing's definition of universal machine in particular. Universal Turing machines can emulate, when receiving in input numbers x and y, the computation on input x carried out by the Turing machine whose numerical code is y. Is there anything resembling *this* general purpose behaviour in analog or hybrid *AD* computation? Is this interpretation of program code in the capabilities of some analog computer? Notice that Myhill's idealizations concerning infinite precision will not be needed here, for the analog machines in question are not supposed to break the effective computability barrier, but to handle in a different way tasks that are well within the bounds of the effectively computable.

The effective coding and decoding of Turing machine descriptions enables one to prove fundamental computability theory results showing how a program can create, modify or simulate a(nother) program. And the needed coding machinery

is made available at low levels of subrecursive hierarchies: functions belonging to Grzegorczyk's class $\mathcal{E}_2$, which includes Smullyan's concatenation operation, are known to provide sufficient coding machinery [5]. Is the possibility of effectively operating on machine codes granted to analog computers of some complexity? And if this is not so, is "virtual machine behaviour" accessible to digital computers only, so that the *species* analog and digital of the *genus* computation fundamentally differ from each other in this respect?

Notice that these *procedurally* motivated questions may be independent of the extensional relationships between the classes of functions that are analog and digitally computable, respectively. Now, the fundamental results of partial recursive function theory, like the normal form theorem, the $S-m-n$ theorem, the recursion theorems, all depend on the possibility of effectively enumerating the *PRF*. Most biologically motivated computational systems—neural networks, p-systems, cellular automata, etc.—have been shown to be universal, usually by implementing in them a universal Turing machine. But the nature of the actual coding is usually neglected. Thus the possibility exists of an abstract extensional equivalence of that computational system's capabilities with the *PRF*, but with no attention paid to the management capabilities, the compiling, the linking, the interpreting that are used in everyday Computer Science and thus in algorithmic simulations of biological behaviour.

A deeper examination of these procedural relationships can be further motivated by the problem of understanding how brain information processing gives rise to intelligent behaviours. In particular, under the supposition that the human brain is able to work as an interpreter with respect to a wide variety of different algorithms, a thoroughly satisfactory explanatory account of these behavioural manifestations and related introspective evidence will have to comprise an explanation of how virtual machine behaviours arise in brain processing.[5]

Is it reasonable to suppose that the human brain works as an interpreter? Indeed it is difficult to understand how imagination, hallucinations, the processes supporting so called theory of mind (*TOM*) or mind reading behaviour, the occurrence of so called higher order thoughts (*HOT*s) or even simple planning in unstructured domains may take place in a system in which some form of virtuality would not be possible. Indeed, a natural suggestion towards implementing these forms of behaviour in a machine would involve calling different routines or the same routine in a different environment. However, no mechanism in the brain has been detected so far that one may interpret as a call to subroutine, let alone the definition of a virtual machine or simply the capability of running

[5] Alternatively, if brain processing does give rise to intelligent behaviour, including mind reading via *TOM* or introspective behaviour via *HOT*s, by analog processes which involve no virtuality, any duplication of those intelligent behaviours by digital processes is unlikely to pave the way to an explanation of how these behaviours take origin in the brain. Thus, a thorough rethinking of functionalist approaches to the explanation of intelligent behaviours is called for, if it turns out that analog and digital computation procedures fundamentally differ from each other in the way of virtual machine behaviour.

different programs. Despite the glib talk about brain, mind, hardware, and software, there is no hint of software in the brain. Everything seems to be handled by repetition. To identify a parameter value (e.g. the slope of a visually presented segment) a parallel battery of yes-no scanners, each one set to a specific slope, is found in the nervous system instead of a single device outputting a graded response (digital or otherwise). While this appears to be a reasonable biological solution to problems of early sensory processing, it is rather puzzling to try and understand how changes of behaviour in response to environmental or internal activity are in fact realized, without resorting to the programming scheme, i.e. to an effective enumeration of the available effective capabilities.

References

1. Copeland, B. J. "Hypercomputation", Minds and Machines 12 (2002), 461-502.
2. Damasio, A. R., *The Feeling of What Happens*, Harcourt Brace & Co., New York-San Diego-London, 1999.
3. Davis, M. "The myth of hypercomputation", in C. Teuscher (ed.), *Alan Turing: Life and Legacy of a Great Thinker*, Springer Verlag, Berlin-Heidelberg, 2004.
4. Myhill, J. "On the possibility of unpredictable machines, Parts I and II." Notes of the University of Michigan Intensive Short Course *Programming concepts, automata and adaptive systems*, mimeograph, Ann Arbor, MI, June 1966.
5. Ritchie, R. W. "Classes of predictably computable functions", Trans. Am. Math. Soc. 106 (1963), 139-173.
6. Scarpellini, B. "Zwei unentscheidbare Probleme der Analysis", Zeitschrift für mathematische Logik und Grundlagen der Mathematik 9 (1963), 265-289.
7. Scarpellini, B. "Comments on two undecidable problems of analysis", Minds and Machines 13 (2003), 79-85.
8. Siegelmann, H. T. *Neural Networks and Analog Computation*, Birkhäuser, Boston, 1999.
9. Siegelmann, H. T. and Sontag, E. D. "On the Computational Power of Neural Nets", J. of Comp. and Syst. Sci. 50 (1995), 132-150.
10. Stannett, M. "Hypercomputational models", in C. Teuscher (ed.), *Alan Turing: Life and Legacy of a Great Thinker*, Springer Verlag, Berlin-Heidelberg, 2004.
11. Turing, A. M. "Computing machinery and intelligence", Mind 59 (1950), 433-460.

A Network Model of Analogue Computation over Metric Algebras

John V. Tucker[1] and Jeffery I. Zucker[2,⋆]

[1] Department of Computer Science, University of Wales Swansea,
Singleton Park, Swansea, SA2 8PP, UK
J.V.Tucker@swansea.ac.uk
[2] Department of Computing & Software, McMaster University,
Hamilton, Ontario L8S 4K1, Canada
zucker@mcmaster.ca

Abstract. We define a general concept of a network of analogue modules connected by channels, processing data from a metric space A, and operating with respect to a global continuous clock $\mathbb{T}$. The inputs and outputs of the network are continuous streams $u : \mathbb{T} \to A$, and the input-output behaviour of the network with system parameters from A is modelled by a function $\Phi : \mathcal{C}[\mathbb{T}, A]^p \times A^r \to \mathcal{C}[\mathbb{T}, A]^q \ \ (p, q > 0, r \geq 0)$, where $\mathcal{C}[\mathbb{T}, A]$ is the set of all continuous streams equipped with the compact-open topology. We give an equational specification of the network, and a semantics which involves solving a fixed point equation over $\mathcal{C}[\mathbb{T}, A]$ using a contraction principle. We analyse a case study involving a mechanical system. Finally, we introduce a custom-made concrete computation theory over $\mathcal{C}[\mathbb{T}, A]$ and show that if the modules are concretely computable then so is the function Φ.

1 Introduction

Let us take analogue computation to be computation by the application of experimental procedures, notably measurement, to physical, chemical or biological systems. Analogue computation is based on continuous data, such as real numbers and data streams. The systems are networks of components or modules that operate in continuous time.

Historically, in analogue computation as conceived by Kelvin [12] and Bush [1], data are represented by measurable physical quantities such as length, voltage, etc., processed by networks of mechanical or electrical components. Currently, analogue computation can involve a much wider range of technologies, inspired, for example, by neural networks and cellular automata.

Digital computation, on the other hand, is fundamentally computation by algorithms on discrete data in discrete time. Starting in the 1930s, classical

⋆ The research of the second author was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 515–529, 2005.

computability theory has matured into a comprehensive and mathematically deep *theory of digital computation*. Turing computability and its equivalents have become the standard for what we mean by computation. The subject continues to develop in new directions [4]. Of particular relevance is Computable Analysis, where it is applied to computable functions on real numbers, Banach spaces, and, more generally, metric and topological spaces.

The *theory of analogue computation* is less developed. The general purpose analog computer (GPAC) was introduced by Shannon [11] to model Bush's Differential Analyzer. Shannon discovered that a function can be generated by a GPAC if, and only if, it is differentially algebraic, but his proof was incomplete. Marian Pour-El [10] gave a characterisation of the analogue computable functions, focusing on the classic analogue systems built from adders, scalar multipliers and integrators. This yielded a new proof of Shannon's equivalence and a proof that these analogue models do not compute all algorithmically (or "digitally") computable functions on the reals.

Cristopher Moore [8] defined a system of schemes rather like Kleene's [6], but with primitive recursion replaced by integration. Félix Costa and his colleagues [3, 7] have presented improved models extending GPAC.

We present two questions related to analogue technology:

1. What characteristics of data, physical components, transmissions, and system architecture, make up a suitable technology for analogue computation?
2. Given a technology that builds analogue systems from components, do these systems produce, by measurements, the same functions as those algorithmically computed?

Thanks to the work of Shannon, Pour-El, Moore and Costa, we have one possible precise formulation of question 1, and negative answer to question 2. Their models are based on the traditional components of analogue computing up to the 1960s (adders, integrators, etc.). However, even for the case of traditional analogue technologies, the conceptual basis is not sufficiently clear to answer (even) the first question fully.

We begin, in Section 2, with a definition of an analogue network, with modules connected by channels, processing data from a metric space A, with a global continuous clock $\mathbb{T}$ modelled by the set of non-negative reals. Let $\mathcal{C}[\mathbb{T}, A]$ be the set of all continuous streams $u : \mathbb{T} \to A$ with the compact-open topology. The input-output behaviour of a network N with p input channels, q output channels and r parameters from A is modelled by a function $\Phi : \mathcal{C}[\mathbb{T}, A]^p \times A^r \to \mathcal{C}[\mathbb{T}, A]^q$. The module functions must satisfy an important physically motivated condition: *causality*. We give an equational specification for N.

In Section 3 we give a semantics for the equational specification of a network satisfying causality. This involves solving a fixed point equation over $\mathcal{C}[\mathbb{T}, A]$ using a custom-made *contraction principle*, based on the fact that $\mathcal{C}[\mathbb{T}, A]$ can be locally approximated by metric spaces. This extends the well-known Banach fixed point theorem for metric spaces [2]. We also derive continuity of Φ, assuming continuity of the module functions. This gives a mathematical model of *computation by measurements on an analogue system*.

In Section 4 we analyse in detail a case study of analogue computation with a mechanical system in which data are represented by displacement, velocity and acceleration.

In Section 5 we compare analogue and digital computation. For this we introduce a custom-made concrete (digital) computation theory over $\mathcal{C}[\mathbb{T}, A]$. This is an extension to the non-metric space $\mathcal{C}[\mathbb{T}, A]$ of the theory of concrete computations on metric algebras [15]. We prove a soundness theorem for analogue, relative to concrete, computation:

Theorem. *If the functions defined by the components of an analogue network are concretely computable, then so is the function defined by the whole network.*

Settling a converse result, i.e. completeness of analogue with respect to digital computation, would be of great importance.

We have studied computation on discrete time streams in [14], and networks that process discrete time streams in [13].

2 Analogue Networks

An *analogue network* N consists of a number of *modules* and *channels* computing and communicating with data from a topological algebra A.

2.1 Data and Time

Assume we are working with data from a *complete metric space* (A, d). The network operates in continuous time $\mathbb{T}$, modelled by the non-negative reals with its usual topology. The channels carry signals in the form of *continuous streams* of data from A, represented as continuous functions $u : \mathbb{T} \to A$. Let $\mathcal{C}[\mathbb{T}, A]$ be the set of continuous streams on A, with the *compact-open topology* [2].

2.2 Modules

A *module* M has finitely many *input channels*, one *output channel*, and locations for some *parameters*. Associated with M is a function $\mathsf{F}_M : \mathcal{C}[\mathbb{T}, A]^{k_M} \times A^{l_M} \to \mathcal{C}[\mathbb{T}, A]$, with $k_M > 0$ *input streams*, $l_M \geq 0$ *parameters* and one *output stream*. We put $\mathsf{F}_M(\boldsymbol{u}, \boldsymbol{c}) = v$, where $\boldsymbol{u} = (u_1, \ldots, u_{k_M}) \in \mathcal{C}[\mathbb{T}, A]^{k_M}$ and $\boldsymbol{c} = (c_1, \ldots, c_{l_M}) \in A^{l_M}$.

Examples 2.2.1. Typical module operations (assuming $A = \mathbb{R}$) are the classical analogue processing units: (a) pointwise addition of two streams, (b) pointwise multiplication of a stream by a constant ("scalar"), (c) integration. There are parameters in (a) and (c), namely the scalar multiplier in (a), and the constant of integration in (c).

We will assume a *causality* property of the module functions, which states that the output is "causally" related to the inputs, in the sense that the output at any time depends only on the inputs up to that time. Precisely:

(*Caus*): *For* $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathcal{C}[\mathbb{T}, A]^{k_M}$, $\boldsymbol{c} \in A^{l_M}$ *and* $t \geq 0$*:*
$$\boldsymbol{u}_1 \restriction_{[0,t)} = \boldsymbol{u}_2 \restriction_{[0,t)} \Rightarrow \mathsf{F}_M(\boldsymbol{u}_1, \boldsymbol{c})(t) = \mathsf{F}_M(\boldsymbol{u}_2, \boldsymbol{c})(t).$$

Note that this condition depends on an assumption of *instantaneous response* of the modules. All the common module operations (including those listed in 2.2.1) satisfy **(*Caus*)**.

2.3 Network Architecture

Consider now (Figure 1) a network N with$m$ modules $M_1, \ldots, M_m$ and m channels $\alpha_1, \ldots, \alpha_m$. Each module $M_i (i = 1, \ldots, m)$ has some *input channels* $\alpha_{i_1}, \ldots, \alpha_{i_{k_i}}$ $(k_i > 0)$ (which are the outputs of modules $M_{i_1}, \ldots, M_{i_{k_i}}$ respectively), some (*local*) *parameter locations* $c_{i_1}, \ldots, c_{i_{l_i}}$ $(l_i \geq 0)$ and one *output channel* α_i. It computes the function $\mathsf{F}_i = \mathsf{F}_{M_i} : \mathcal{C}[\mathbb{T}, A]^{k_i} \times A^{l_i} \to \mathcal{C}[\mathbb{T}, A]$.

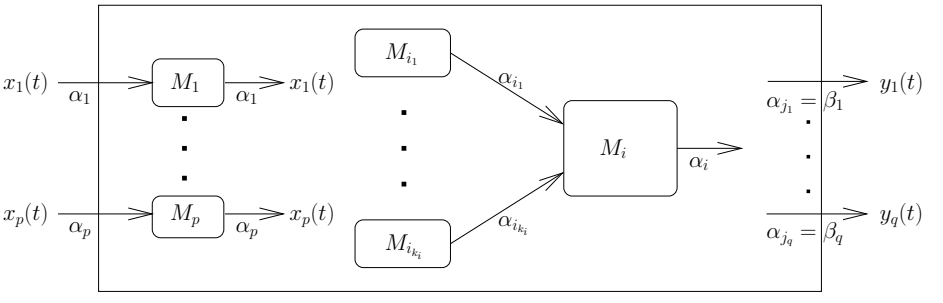


Fig. 1. A network

The network N itself has p *input channels* and q *output channels* $(p, q \leq m)$. We assume (for notational convenience) that the first p modules $M_1, \ldots, M_p$ are the identity module $M_{\mathbf{I}}$, and the p network input channels $\alpha_1, \ldots, \alpha_p$ are both the input and output channels for $M_{\mathbf{I}}$. The remaining (non-trivial) modules of the network are $M_{p+1}, \ldots, M_m$. For $i = 1, \ldots, m$, the channel α_i is the output channel for module M_i. The q network output channels are $\beta_1, \ldots, \beta_q$, where (say) $\beta_i = \alpha_{j_i}$ for $i = 1, \ldots, q$.

There are also locations for *global* or *network parameters* $\boldsymbol{c} = (c_1, \ldots, c_r)$ $(r \geq 0)$, which include the local parameters of all the modules in N. For each global parameter c_i and module M_j, it is specified which of the local parameters of M_j are to be identified with c_i.

We make an assumption of ***input determinacy***:

(*InDet*): *There is a well-determined value for the stream on each input channel at all times.*

2.4 Network Operation: The Model

Under the assumptions **(*InDet*)** and **(*Caus*)**, we want to prove a ***network determinacy*** condition:

(***NetDet***): *For certain input streams and parameter values, there is a well-determined value for the stream on each channel at all times.*

This means that, at least for a certain set $U \subseteq \mathcal{C}[\mathbb{T}, A]^p \times A^r$ of inputs and parameters, there is a well-determined tuple of total functions $u_i : \mathbb{T} \to A (i = 1, \ldots, m)$ that describes the data on every channel α_i.

Assuming (***NetDet***), there is associated with each module M_i $(i = 1, \ldots, m)$ a function $\Phi_i : \mathcal{C}[\mathbb{T}, A]^p \times A^r \rightharpoonup \mathcal{C}[\mathbb{T}, A]$ where $\Phi_i(\boldsymbol{x}, \boldsymbol{c}) = u_i$ for $(\boldsymbol{x}, \boldsymbol{c}) \in U$. From these module functions follows the existence of the *network function*

$$\begin{aligned} \Phi^N &: \mathcal{C}[\mathbb{T}, A]^p \times A^r \rightharpoonup \mathcal{C}[\mathbb{T}, A]^q, \\ \Phi^N(\boldsymbol{x}, \boldsymbol{c}) &= (\Phi_{j_1}(\boldsymbol{x}, \boldsymbol{c}), \ldots, \Phi_{j_q}(\boldsymbol{x}, \boldsymbol{c})) \qquad \text{for } (\boldsymbol{x}, \boldsymbol{c}) \in U. \end{aligned} \tag{2.1}$$

2.5 Network Operation: Algebraic Specification

Given the above assumptions, we can specify the model by the following *system equations*:

$$u_i(t) = \mathsf{F}_i(u_{i1}, \ldots, u_{ik_i}, c_{i1}, \ldots, c_{il_i})(t) \qquad (i = 1, \ldots, m,\ t \geq 0) \tag{2.2a}$$

$$u_i(t) = x_i(t) \qquad (i = 1, \ldots, p,\ t \geq 0), \tag{2.2b}$$

In the next section we will derive the existence and uniqueness of a solution of this specification as a fixed point of a certain function.

3 Solving Network Equations; Fixed Point Semantics

We are looking for an m-tuple of channel functions satisfying the equational specifications (2.2). First, we define some general concepts and give some results concerning stream spaces and stream transformations. Recall that (A, d) is a complete metric space.

3.1 Stream Spaces and Stream Transformations

Let $0 \leq a < b$, and let $\mathcal{C}[[a, b], A]$ be the set of continuous functions from $[a, b]$ to A. For $u, v \in \mathcal{C}[[a, b], A]$ (or $u, v \in \mathcal{C}[\mathbb{T}, A]$), define

$$\mathsf{d}_{a,b}(u, v) =_{df} \sup\{\mathsf{d}(u(t), v(t)) \mid t \in [a, b]\}.$$

This makes $\mathcal{C}[[a, b], A]$ a complete metric space, with the *uniform convergence* topology [2–§2.6]. The product space $\mathcal{C}[[a, b], A]^m$ $(m > 0)$ has the metric

$$\mathsf{d}^m_{a,b}(\boldsymbol{u}, \boldsymbol{v}) = \Big(\sum_{i=1}^{m} \big(\mathsf{d}_k(u_i, v_i)\big)^p\Big)^{\frac{1}{p}} \tag{3.1}$$

(where $\boldsymbol{u} = (u_1, \ldots, u_m)$ and $\boldsymbol{v} = (v_1, \ldots, v_m)$) for some fixed p $(1 \leq p \leq \infty)$. We will sometimes drop the superscript 'm' from $\mathsf{d}^m_{a,b}$. We also write d_k for $\mathsf{d}_{0,k}$ $(k = 1, 2, \ldots)$.

The *stream space* $\mathcal{C}[\mathbb{T}, A]$ is, in general, not a metric space, and $\mathsf{d}_{a,b}$ is only a pseudometric on $\mathcal{C}[\mathbb{T}, A]$. Nevertheless we can define a notion of convergence in $\mathcal{C}[\mathbb{T}, A]$ as follows. A sequence $(u_0, u_1, u_2, \dots)$ of elements of $\mathcal{C}[\mathbb{T}, A]$ is said to *converge locally uniformly* to the limit $u \in \mathcal{C}[\mathbb{T}, A]$ if for all k there exists N such that for all $n \geq N$, $\mathsf{d}_k(u_n, u_N) \leq 2^{-k}$. The space $\mathcal{C}[\mathbb{T}, A]$ is given the *compact open topology* [2–§3.4]. This is equivalent to the *topology of local uniform convergence*, which can be characterised as follows. Given a set $X \subseteq \mathcal{C}[\mathbb{T}, A]$ and a point $u \in \mathcal{C}[\mathbb{T}, A]$, u is in the *closure* of X if, and only if, there is a sequence of elements of X which converges locally uniformly to u.

This topology on $\mathcal{C}[\mathbb{T}, A]$ can also be characterised as the *inverse limit* [2] of the family of topological spaces $\mathcal{C}[[0, k], A]$ $(k = 0, 1, 2, \dots)$ with mappings $\pi_k : C[[0, k+1], A] \to \mathcal{C}[[0, k], A]$ defined by $\pi_k(u) = u\restriction_k$.

The space $\mathcal{C}[\mathbb{T}, A]$ is *complete* in the following sense. We must first define:

Definition 3.1.1 (Locally uniform Cauchy sequence). A sequence $(u_0, u_1, u_2, \dots)$ of elements of $\mathcal{C}[\mathbb{T}, A]$ is *locally uniform Cauchy* if $\forall k \, \exists N \, \forall m, n \geq N : \mathsf{d}_k(u_m, u_n) \leq 2^{-k}$.

Lemma 3.1.2 (Completeness of $\mathcal{C}[\mathbb{T}, A]$). *A locally uniform Cauchy sequence in $\mathcal{C}[\mathbb{T}, A]$ converges locally uniformly to a limit.*

We are interested in stream transformations $f : \mathcal{C}[\mathbb{T}, A]^m \to \mathcal{C}[\mathbb{T}, A]^m$.

Definition 3.1.3 (Contracting stream transformations). Let $0 < \kappa < 1$ and $\tau > 0$. A stream transformation $f : \mathcal{C}[\mathbb{T}, A]^m \to \mathcal{C}[\mathbb{T}, A]^m$ is *contracting w.r.t.* (κ, τ), or in $\boldsymbol{Contr}(\kappa, \tau)$, if for all $T \geq 0$ and all $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}[\mathbb{T}, A]^m$:

$$\mathsf{d}_{T,T+\tau}(f(\boldsymbol{u}), f(\boldsymbol{v})) \leq \kappa \cdot \mathsf{d}_{T,T+\tau}(\boldsymbol{u}, \boldsymbol{v}).$$

Lemma 3.1.4. *Suppose f satisfies* (***Caus***). *If $f \in \boldsymbol{Contr}(\kappa, \tau)$ for some $\tau > 0$, then $f \in \boldsymbol{Contr}(\kappa, \tau')$ for all $\tau' > 0$.*

Remark 3.1.5. Hence if $f \in \boldsymbol{Contr}(\kappa, \tau)$, we can choose τ freely. We will henceforth write $\boldsymbol{Contr}(\kappa)$ instead of $\boldsymbol{Contr}(\kappa, \tau)$, and generally take $\tau = 1$.

Theorem 1 (Fixed point of contracting stream transformation). *Suppose $f \in \boldsymbol{Contr}(\kappa)$ for some $\kappa < 1$. Then f has a unique fixed point, i.e., there is a unique $\boldsymbol{u} \in \mathcal{C}[\mathbb{T}, A]^m$ such that $f(\boldsymbol{u}) = \boldsymbol{u}$.*

Proof. Uniqueness is an easy exercise. We prove existence by constructing a fixed point $\boldsymbol{u}$ of f as a limit of a locally uniformly convergent Cauchy sequence of stream tuples:

$$\boldsymbol{u}_0, \ \boldsymbol{u}_1, \ \boldsymbol{u}_2, \ \dots \tag{3.2}$$

Define $\boldsymbol{u}_0$ arbitrarily, and $\boldsymbol{u}_{n+1} = f(\boldsymbol{u}_n)$. Then for all k, n, by induction on n:

$$\mathsf{d}_k(\boldsymbol{u}_{n+1}, \boldsymbol{u}_n) \leq \kappa^n \mathsf{d}_k(\boldsymbol{u}_1, \boldsymbol{u}_0).$$

The sequence (3.2) can then be seen to be a locally uniform Cauchy sequence, by choosing N (for a given k, in Definition 3.1.1) such that

$$\kappa^N < \frac{2^{-k}}{\mathsf{d}_k(\boldsymbol{u}_1, \boldsymbol{u}_0)}.$$

Thus by Lemma 3.1.2, the sequence (3.2) converges locally uniformly to a limit $\boldsymbol{u}$. Hence, also, the sequence

$$f(\boldsymbol{u}_0),\ f(\boldsymbol{u}_1),\ f(\boldsymbol{u}_2),\ \ldots \tag{3.3}$$

converges locally uniformly to $f(\boldsymbol{u})$, since by the contraction property of f,

$$\mathsf{d}(f(\boldsymbol{u}_n), f(\boldsymbol{u})) \ \leq\ \kappa \cdot \mathsf{d}(\boldsymbol{u}_n, \boldsymbol{u}).$$

Since (3.3) is the sequence (3.2) shifted by 1, it also converges to $\boldsymbol{u}$, and so $f(\boldsymbol{u}) = \boldsymbol{u}$. $\square$

In Section 5, where we consider the *computability* of the fixed point $\boldsymbol{u}$ as a function of the inputs $(\boldsymbol{x}, \boldsymbol{c})$, we will need a stronger property of the sequence (3.2) than local uniform convergence, namely *effective* local uniform convergence.

We turn to apply the above theory to the network N.

3.2 Network Function

Recall the network function Φ^N (2.1) and the specifications (2.2). Notice next that a stream tuple $(u_1, \ldots, u_m)$ satisfying the specifications (2.2) can be characterised as a *fixed point* of the function

$$\Psi^N_{\boldsymbol{c}} : \mathcal{C}[\mathbb{T}, A]^m \ \rightarrow\ \mathcal{C}[\mathbb{T}, A]^m$$

defined by
$$\Psi^N_{\boldsymbol{c}}(u_1, \ldots, u_m) \ =\ (\mathsf{F}_1(\boldsymbol{u}_1, \boldsymbol{c}_1), \ldots, \mathsf{F}_m(\boldsymbol{u}_m, \boldsymbol{c}_m)) \tag{3.4}$$

(where, on the r.h.s., $\boldsymbol{u}_i, \boldsymbol{c}_i$ are the lists of input streams and local parameters associated with F_i) subject to the constraint (2.2*b*). Now by equation (2.2*b*), the first p components $(u_1, \ldots, u_p)$ of the tuple $(u_1, \ldots, u_m)$ on the left hand side are identical to $\boldsymbol{x}$. Similarly, on the right hand side, for $i = 1, \ldots, p$, F_i is the identity function, with argument $u_i = x_i$, and so (3.4) can be rewritten as

$$\Psi^N_{\boldsymbol{c}}(\boldsymbol{x}, u_{p+1}, \ldots, u_m) \ =\ (\boldsymbol{x},\ \mathsf{F}_{p+1}(\boldsymbol{u}_{p+1}, \boldsymbol{c}_{p+1}), \ldots, \mathsf{F}_m(\boldsymbol{u}_m, \boldsymbol{c}_m)). \tag{3.5}$$

Therefore Ψ^N can be reformulated as a function only of the *non-input streams* $\boldsymbol{u} \ =\ (u_{p+1}, \ldots, u_m)$, with the *input streams* $\boldsymbol{x}$ as further parameters, thus:

$$\begin{aligned} \Psi^N_{\boldsymbol{c},\boldsymbol{x}} \ :\ & \mathcal{C}[\mathbb{T}, A]^{m-p} \ \rightarrow\ \mathcal{C}[\mathbb{T}, A]^{m-p} \\ & \Psi^N_{\boldsymbol{c},\boldsymbol{x}}(\boldsymbol{u}) \ =_{df}\ \Psi^N_{\boldsymbol{c}}(\boldsymbol{x}, \boldsymbol{u}). \end{aligned} \tag{3.6}$$

So *a fixed point for* $\Psi^N_{\boldsymbol{c},\boldsymbol{x}}$ *will be a solution to* (2.2). Thus the basic questions are:

- Under what conditions does $\Psi^N_{\boldsymbol{c},\boldsymbol{x}}$ have a fixed point?
- Under what conditions is it unique?

We will give at least a partial solution to this, namely a sufficient condition for a fixed point, by applying the theory of contracting stream transformations developed above.

3.3 Solution of Fixed Point Equation

Recall Def. 3.1.3 and Remark 3.1.5.

Definition 3.3.1 (Contracting condition for network). Given $\boldsymbol{c} \in A^r$, $\boldsymbol{x} \in \mathcal{C}[\mathbb{T}, A]^p$ and $0 < \kappa < 1$, the network N satisfies $\boldsymbol{Contr}_{\boldsymbol{c},\boldsymbol{x}}(\kappa)$ if the stream transformation $\Psi^N_{\boldsymbol{c},\boldsymbol{x}}$ is in $\boldsymbol{Contr}(\kappa)$. It is *contracting at* $(\boldsymbol{c}, \boldsymbol{x})$ if it satisfies $\boldsymbol{Contr}_{\boldsymbol{c},\boldsymbol{x}}(\kappa)$ for some $\kappa < 1$.

Theorem 2.

(a) *Suppose for all* $(\boldsymbol{x}, \boldsymbol{c}) \in U \subseteq \mathcal{C}[\mathbb{T}, A]^p \times A^r$, *there exists* $\kappa < 1$ *such that the network* N *satisfies* $\boldsymbol{Contr}_{\boldsymbol{c},\boldsymbol{x}}(\kappa)$. *Then for all* $(\boldsymbol{x}, \boldsymbol{c}) \in U$ *there is a unique* $\boldsymbol{u} = (u_1, \ldots, u_m) \in \mathcal{C}[\mathbb{T}, A]^m$ *satisfying* (2.2). *It is given by specifying* $u_i = x_i$ *for* $i = 1, \ldots, p$, *and* $\boldsymbol{u} = (u_{p+1}, \ldots, u_m)$ *as the unique fixed point of the function* $\Psi^N_{\boldsymbol{c},\boldsymbol{x}}$ *defined by equations* (3.5) *and* (3.6). *This defines the network function* Φ^N *as in* (2.1), *with* $\Phi^N(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{u}$ *for all* $(\boldsymbol{x}, \boldsymbol{c}) \in U$.

(b) *If, in addition, the module functions are continuous, then* Φ^N *is continuous at all points in* U *at which* κ *can be defined continuously.*

Part (a) is immediate from Theorem 1. We omit the proof of (b).

4 A Case Study

We apply the theory of Section 3 to an example from a standard text [5].

4.1 The Physical System

(See Figure 2.) A mass M is suspended by a spring with stiffness K and damping coefficient D. A force f (varying with time t) is applied to M. We want to compute its displacement x as a function of t.

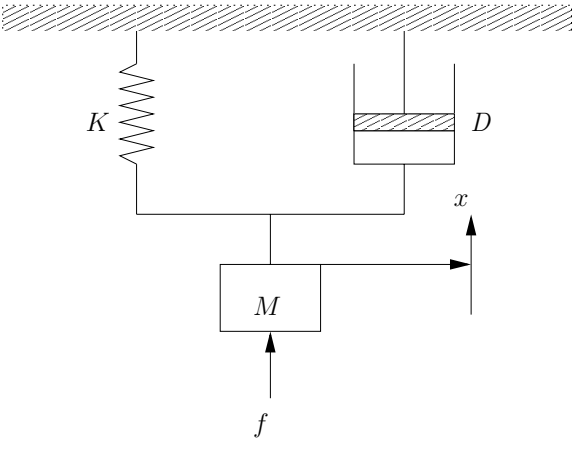


Fig. 2. Case study: The physical system

4.2 Equational Specification

Three forces act on the mass: the external force f, the spring force $-Kx$, and the damping force $-Ddx/dt$. By Newton's second law of motion, $Ma+Dv+Kx = f$, where $v = dx/dt$ is the velocity, and $a = dv/dt$ the acceleration.

4.3 Network

The analogue network N for this system is shown in Figure 3. It is simplified from the one in [5], by combining each scalar multiplier with the preceding or following module. There is also an extra "identity" module M_1 for the input stream f.

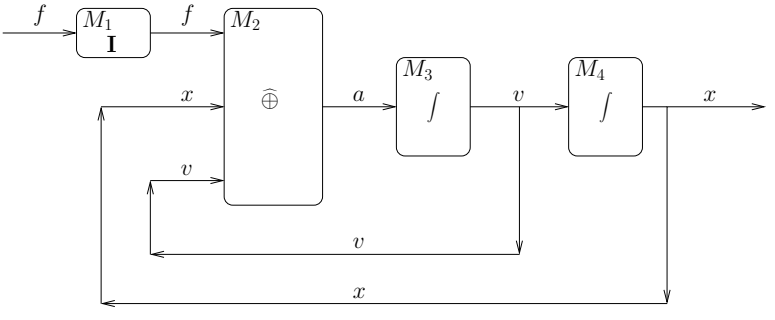


Fig. 3. Case study: The network

There are 3 other modules M_2, M_3, M_4, with associated functions F_i $(i = 2, 3, 4)$:

$$\begin{aligned} a(t) &= \mathsf{F}_2(f, x, v)(t) = (f(t) - Kx(t) - Dv(t))/M \\ v(t) &= \mathsf{F}_3(a)(t) = (\textstyle\int_0^t a) + v_0 \\ x(t) &= \mathsf{F}_4(a)(t) = (\textstyle\int_0^t v) + x_0 \end{aligned}$$

The integration constants v_0 and x_0 represent initial velocity and displacement.

4.4 Network Semantics

The parameter list is $\boldsymbol{c} = (M, K, D, v_0, x_0)$, the single *input stream* is f, and the list of *non-input streams* is $\boldsymbol{u} = (a, v, x)$. So we want a fixed point of the function $\Psi_{\boldsymbol{c},f} : \mathcal{C}[\mathbb{T}, \mathbb{R}]^3 \rightharpoonup \mathcal{C}[\mathbb{T}, \mathbb{R}]^3$, where $\Psi_{\boldsymbol{c},f}(a, v, x) = (a', v', x')$ with

$$\begin{aligned} a'(t) &= (f(t) - Kx(t) - Dv(t))/M \\ v'(t) &= (\textstyle\int_0^t a) + v_0 \\ x'(t) &= (\textstyle\int_0^t v) + x_0. \end{aligned} \tag{4.1}$$

For changes $\delta a, \delta v, \delta x$ in a, v, x, and corresponding changes $\delta a', \ldots$ in $a', \ldots$:

$$\Psi_{\boldsymbol{c},f}(a + \delta a,\, v + \delta v,\, x + \delta x) = (a' + \delta a',\, v' + \delta v',\, x' + \delta x').$$

Then (using the pseudonorm $\|u\| =_{df} \sup\{u(t) \mid T \leq t \leq T+\tau\}$) from (4.1):

$$\|\delta a'\| \leq (K\|\delta x\| + D\|\delta v\|)/M \quad (4.2a)$$
$$\|\delta v'\| \leq \tau\|\delta a\| \quad (4.2b)$$
$$\|\delta x'\| \leq \tau\|\delta v\|. \quad (4.2c)$$

Now assume $M > \max(K, 2D)$ (4.3)

and put $\kappa =_{df} \max(K, 2D)/M,$ (4.4)

$\tau =_{df} D/M.$ (4.5)

By (4.3), $\kappa < 1$. Define the product pseudonorm $\|(\delta a, \delta v, \delta x)\| =_{df} \|\delta a\| + \|\delta v\| + \|\delta x\|$. (This corresponds to taking $p = 1$ in (3.1).) Then

$$\begin{aligned} \|(\delta a', \delta v', \delta x')\| &= \|\delta a'\| + \|\delta v'\| + \|\delta x'\| \\ &\leq (K/M)\|\delta x\| + (D/M + \tau)\|\delta v\| + \tau\|\delta a\| \quad \text{by (4.2)} \\ &\leq \kappa\,\|(\delta a, \delta v, \delta x)\| \quad \text{by (4.4) \& (4.5)} \end{aligned}$$

which proves $\boldsymbol{Contr}_{\boldsymbol{c},f}(\kappa)$. Note that the only assumption needed to prove the contraction property was (4.3), i.e., that the mass M be sufficiently large relative to the stiffness K and damping coefficient D. No assumption was needed on either the initial values v_0 and x_0 of velocity and displacement, or the external force $f(t)$. Hence, from Theorem 2:

Theorem 3. *The network of Figure 3 is contracting, and hence satisfies* $(\boldsymbol{NetDet})$, *for any input stream $f(t)$, provided $M > \max(K, 2D)$.*

Corollary. *The system of Figure 2 has a well-determined solution $(a(t), v(t), x(t))$ for the acceleration, velocity and displacement as functions of time $t \geq 0$, for any input force $f(t)$ as a continuous function of time $t \geq 0$, and any initial conditions (v_0, x_0) for the velocity and displacement, provided only that $M > \max(K, 2D)$. Moreover, under this condition, the solution streams (a, v, x) depend continuously on the input stream f and the parameters (M, K, D, v_0, x_0).*

5 Computability of the Solution

We want to show that the network function which solves the network specification (2.2) according to Theorem 2 is computable relative to the module functions; in other words, the output streams are computable from the input streams, parameters, and module functions. Hence if the module functions are computable, then so is the network function.

By "computable" here we mean: computable according to some concrete model of computation on $\mathcal{C}[\mathbb{T}, A]$. We give a new model, inspired by the approximation of $\mathcal{C}[\mathbb{T}, A]$ by the metric spaces $\mathcal{C}[[0, k], A]$.

An alternative treatment of concrete computation on the space $\mathcal{C}[X, Y]$ with the compact-open topology is given in [16], with $X \subseteq \mathbb{R}^m$ and $Y = \mathbb{R}^n$.

5.1 Topological Algebra of Streams

Consider the *5-sorted topological algebra*

$$\mathcal{C} \;=\; (A,\, \mathbb{R},\, \mathbb{T},\, \mathcal{C}[\mathbb{T}, A],\, \mathbb{N};\; \mathsf{d},\, \mathsf{eval})$$

where $\mathsf{d} : A^2 \to \mathbb{R}$ and $\mathsf{eval} : \mathcal{C}[\mathbb{T}, A] \times \mathbb{T} \to A$ are, respectively, the distance function on A and the evaluation function: $\mathsf{eval}(u, t) \;=\; u(t)$. $\mathcal{C}$ is a topological algebra, because each of the five carriers has an associated topology with respect to which the basic functions (d and eval) are continuous. The carrier $\mathbb{R}$ is needed for the metric on A. The set of sorts of the signature Σ of $\mathcal{C}$ is $\boldsymbol{Sort} = \boldsymbol{Sort}(\Sigma) = \{\, \mathsf{A},\, \mathsf{R},\, \mathsf{T},\, \mathsf{C},\, \mathsf{N} \,\}$. For ease of notation, we also refer to the five carriers of $\mathcal{C}$ as C_s for $s \in \boldsymbol{Sort}$.

5.2 Enumerations of Subfamilies of $\mathcal{C}$

The following extends the concepts in [15] on concrete computation on metric algebras to the case of the topological (non-metric) algebra $\mathcal{C}[\mathbb{T}, A]$. We will fix an *enumeration* of certain subsets of the carriers, i.e., a family α of bijections $\alpha_s : \mathbb{N} \twoheadrightarrow X_s \subseteq C_s$ ($s \in \boldsymbol{Sort}$) of $\mathbb{N}$ with certain subsets X_s of C_s. The pair (X_s, α_s) is called an *enumerated subset* of C_s. The enumerations are as follows.

First, the mapping $\alpha_{\mathsf{A}} : \mathbb{N} \twoheadrightarrow X \subseteq A$ is an enumeration of some dense subset X of A. Here we need a separability assumption:

(*Sep*): *A is separable.*

From **(*Sep*)** follows that $\mathcal{C}[\mathbb{T}, A]$ is also separable. This enumeration α_{A} (or rather its "computational closure" $\overline{\alpha}_{\mathsf{A}}$, see below) must also satisfy a *Σ-effectivity* assumption, to be described below (5.4.3). The mapping $\alpha_{\mathsf{R}} : \mathbb{N} \twoheadrightarrow \mathbb{Q} \subset \mathbb{R}$ is a standard enumeration of the rationals. (In case $A = \mathbb{R}$, α_{A} is the same as α_{R}.) Similarly $\alpha_{\mathsf{T}} : \mathbb{N} \twoheadrightarrow \mathbb{Q}^+ \subset \mathbb{T}$ is a standard enumeration of the non-negative rationals. The mapping α_{N} is just the identity on $\mathbb{N}$. Finally, and most interestingly, the mapping $\alpha_{\mathsf{C}} : \mathbb{N} \twoheadrightarrow Z \subset \mathcal{C}[\mathbb{T}, A]$ is a "standard" enumeration of some countable dense subset Z of $\mathcal{C}[\mathbb{T}, A]$, which must satisfy a *$\Sigma$-effectivity* assumption (5.4.3 below), as well as the following:

Assumption 5.2.1 (Effective locally uniform continuity of (Z, α_{C})). *There is a recursive function $\mu : \mathbb{N}^3 \to \mathbb{N}$ such that for all k, ℓ, n, writing $z_n = \alpha_{\mathcal{C}}(n)$:*

$$\forall t_1, t_2 \in [0, k] : |t_1 - t_2| < 2^{-\mu(k,\ell,n)} \Rightarrow \mathsf{d}(z_n(t_1),\, z_n(t_2)) < 2^{-\ell}.$$

5.3 Computational Closure

For our model of *concrete computation* on $\mathcal{C}[\mathbb{T}, A]$, we construct the *computational closures* $\mathcal{C}_{\alpha_s}(X_s)$ of the enumerated subsets (X_s, α_s) of the spaces C_s, with enumerations $\overline{\alpha}_s : \Omega_{\overline{\alpha}_s} \twoheadrightarrow \mathcal{C}_{\alpha_s}(X_s)$, so that $X_s \subseteq \mathcal{C}_{\alpha_s}(X_s) \subseteq C_s$ for $s \in \boldsymbol{Sort}$, as we now describe.

First, for the metric space A, we define the set $\mathcal{C}_{\alpha_{\mathsf{A}}}(X)$ of *α-computable elements of A*, to be the limits in A of effectively convergent Cauchy sequences of elements of the enumerated subset X, with corresponding enumeration $\overline{\alpha}_{\mathsf{A}}$. Details of the construction of $\mathcal{C}_{\alpha_{\mathsf{A}}}(X)$ and $\overline{\alpha}_{\mathsf{A}}$ can be found in [15]. We omit them, since below, for the computational closure of $Z \in \mathcal{C}[\mathbb{T}, A]$, we describe a model of concrete computability for a more general situation — the non-metric topological space $\mathcal{C}[\mathbb{T}, A]$.

The computational closures $\mathcal{C}_{\alpha_{\mathsf{R}}}(\mathbb{Q})$ and $\mathcal{C}_{\alpha_{\mathsf{T}}}(\mathbb{Q}^+)$ in $\mathbb{R}$ and $\mathbb{T}$ respectively are defined in the same way. The computational closure of $\mathbb{N}$ is, trivially, $\mathbb{N}$, with (again) the identity enumeration. Finally, for the space $\mathcal{C}[\mathbb{T}, A]$ with its enumerated subset (Z, α_{C}) (where we henceforth usually drop the subscripts of α and $\overline{\alpha}$), let $\mathcal{C}_\alpha(Z) \subset \mathcal{C}[\mathbb{T}, A]$ be the set of all limits in $\mathcal{C}[\mathbb{T}, A]$ of *α-effectively locally uniform Cauchy sequences* of elements of Z — such limits always existing by the completeness of $\mathcal{C}[\mathbb{T}, A]$ (Lemma 3.1.2) — and let $\Omega_{\overline{\alpha}} \subset \mathbb{N}$ be the set of codes for $\mathcal{C}_\alpha(Z)$. More precisely, $\Omega_{\overline{\alpha}}$ consists of pairs of numbers $c = \langle e, m\rangle$ where (i) e is an index for a recursive function defining a sequence

$$z_0,\ z_1,\ z_2,\ \ldots \tag{5.1}$$

of elements of Z, where $z_n = \alpha(\{e\}(n))$; and (ii) m is an index for a modulus of *local uniform convergence*, i.e., $\forall k, \forall n, p \geq \{m\}(k) : \mathsf{d}_k(z_n\restriction_k, z_p\restriction_k) \leq 2^{-k}$. For any such code c, $\overline{\alpha}(c)$ is defined as the limit in $\mathcal{C}[\mathbb{T}, A]$ of the Cauchy sequence (5.1), and $\mathcal{C}_\alpha(Z)$ is the range of $\overline{\alpha}$.

5.4 Concrete Computation on $\mathcal{C}[\mathbb{T}, A]$

For a tuple of sorts $\sigma = (s_1, \ldots, s_m)$ we have the *product space* $C^\sigma =_{df} C_{s_1} \times \cdots \times C_{s_m}$, the *product domain* $\Omega^\sigma_{\overline{\alpha}} =_{df} \Omega_{\overline{\alpha}_{s_1}} \times \cdots \times \Omega_{\overline{\alpha}_{s_m}} \subseteq \mathbb{N}^m$, and the *product enumeration* $\overline{\alpha}^\sigma = (\alpha_{s_1}, \ldots, \alpha_{s_m}) : \Omega^\sigma_{\overline{\alpha}} \to C^\sigma$.

Definition 5.4.1 (Tracking function). Let $f : C^\sigma \rightharpoonup C_s$. A function $\varphi : \Omega^\sigma_{\overline{\alpha}} \rightharpoonup \Omega_{\overline{\alpha}_s}$ is an *$\overline{\alpha}$-tracking function* for f if the following diagram commutes:

$$\begin{array}{ccc} C^\sigma & \xrightarrow{\ f\ } & C_s \\ {\scriptstyle \overline{\alpha}^\sigma}\uparrow & & \uparrow{\scriptstyle \overline{\alpha}_s} \\ \Omega^\sigma_{\overline{\alpha}} & \xrightarrow[\ \varphi\ ]{} & \Omega_{\overline{\alpha}} \end{array}$$

Definition 5.4.2 (Concrete computability on $\mathcal{C}[\mathbb{T}, A]$). Suppose $f, g_1, \ldots, g_k$ are functions on $\mathcal{C}[\mathbb{T}, A]$ with $\overline{\alpha}$-tracking functions $\varphi, \psi_1, \ldots, \psi_k$ respectively. Then f is *$\overline{\alpha}$-computable in* (or *relative to*) $g_1, \ldots, g_k$ iff φ is partially recursive in $\psi_i, \ldots, \psi_k$.

We need one more assumption on the enumeration α.

Assumption 5.4.3 (Σ-effectivity of $\overline{\alpha}$). *The basic functions of the algebra $\mathcal{C}[\mathbb{T}, A]$, namely* d *and* eval*, are $\overline{\alpha}$-computable.*

Example 5.4.4 (Concrete computation on $\mathcal{C}[\mathbb{T}, \mathbb{R}]$). Consider, in particular, the case that the metric space A is $\mathbb{R}$. As stated above, for α_{A} we would take the same as α_{R}, i.e., a standard enumeration of the rationals.

As an example of a countable dense subset of $\mathcal{C}[\mathbb{T}, \mathbb{R}]$, take $Z = \mathsf{ZZ}$, the set of all continuous "zigzag functions" from $\mathbb{T}$ to $\mathbb{R}$ with finitely many turning points, all with rational coordinates. Clearly, the set ZZ, under any reasonable enumeration α_{C}, satisfies the effective locally uniform continuity assumption (5.2.1). Also, the enumeration $\overline{\alpha}$ derived from α is clearly Σ-effective (Assumption 5.4.3).

We could use instead, as our starting point, the set of polynomial functions of t with rational coefficients. This yields the same set $\mathcal{C}_\alpha(Z)$ of computable elements of $\mathcal{C}[\mathbb{T}, \mathbb{R}]$.

5.5 Relative Concrete Computability of Functions Defined by Analog Networks

Given a network N, we want to show the network function Φ^N is $\overline{\alpha}$-computable relative to the module functions, provided it is contracting at the parameter and stream inputs.

For this we need a constructive concept of contraction, namely that a contracting factor $\kappa < 1$ can be found *effectively* in the parameters and stream inputs $\boldsymbol{c}, \boldsymbol{x}$ over some domain.

Definition 5.5.1 (Effectively contracting network). Given $U \subseteq \mathcal{C}[\mathbb{T}, A]^p \times A^r$, the network N is $(\overline{\alpha})$-*effectively contracting* on U if a contracting factor $\kappa_{\boldsymbol{x},\boldsymbol{c}}$ can be found $\overline{\alpha}$-effectively in $(\boldsymbol{x}, \boldsymbol{c}) \in U$.

Note that this certainly holds with the case study in Section 4, where a value for κ can be found effectively in the parameters M, K, D (and independent of the input stream f), by equations (4.3) and (4.4), in the region $U =_{df} \{ (M, K, D) \in \mathbb{R}^3 \mid M > \max(K, 2D) \}$.

Theorem 4. *Suppose the network N satisfies* (***Caus***)*, (Z, α_{C}) satisfies effective local uniform continuity, and $\overline{\alpha}$ is Σ-effective. Suppose also N is $\overline{\alpha}$-effectively contracting on $U \subseteq \mathcal{C}[\mathbb{T}, A]^p \times A^r$. Then the network function Φ^N is defined (at least) on U, and is $\overline{\alpha}$-computable relative to the module functions of N. Hence if the module functions are $\overline{\alpha}$-computable, then so is Φ^N.*

Proof (outline). For an input $(\boldsymbol{x}, \boldsymbol{c}) \in U$, the output of Φ^N is a sub-tuple of the fixed point $\boldsymbol{u}$ of $\Psi^N_{\boldsymbol{c},\boldsymbol{x}}$ (§3.2). So it suffices to show that the function from $(\boldsymbol{x}, \boldsymbol{c}) \in U$ to this $\boldsymbol{u}$ is computable. (Here "computable" means $\overline{\alpha}$-computable relative to the module functions.)

Consider the sequence of stream tuples $\boldsymbol{u}_n$, defined in the proof of Theorem 1, with $f = \Psi^N_{\boldsymbol{c},\boldsymbol{x}}$. First, each $\boldsymbol{u}_n$ is computable in $(\boldsymbol{x}, \boldsymbol{c})$, by induction on n.

Further, $(\boldsymbol{u}_n)$ is an *effectively* locally uniform Cauchy sequence, i.e. (in the notation of Definition 3.1.1) N can be obtained effectively from k. From this it follows that the limit $\boldsymbol{u}$ of this sequence, which is the fixed point of $\Psi^N_{\boldsymbol{c},\boldsymbol{x}}$, is also computable in $(\boldsymbol{x}, \boldsymbol{c})$. □

5.6 Concrete Computability of Module Functions

The standard module functions on $\mathcal{C}[\mathbb{T}, \mathbb{R}]$ are $\overline{\alpha}$-computable. For (a) *pointwise addition* and (b) *scalar multiplication* this is obvious. The interesting case is (c) *integration*. Here, in taking the integral as the limit of a Cauchy sequence of Riemann sums, we use the *effective locally uniform continuity* assumption (5.2.1).

Thus all the module functions in the case study in Section 4 are concretely computable. Combining this with Theorem 4, we conclude that the function which solves the network equations in that example is concretely computable.

6 Concluding Remarks

Most current research on analogue systems is focused on computation on the reals with the traditional processing units (adders, integrators etc.). Our network models, involving arbitrary processing units on data from metric spaces in continuous time, are new.

Several questions and problems are left open:

1. For the modules, it turned out that we did not need the assumption of *time invariance* (satisfied by the standard module functions in §5.6) which, like *causality* (which we did need) is common in dynamical system theory [9]. What is the significance of this assumption — or its absence?
2. What if we allow *partial* or *many-valued* module functions or partial streams?
3. Find reasonable conditions, other than the contraction property, that guarantee "good behaviour" of these networks.
4. Characterise the networks that produce all (and only) computable functions on $\mathcal{C}[\mathbb{T}, A]$.

References

1. V. Bush. The differential analyzer: a new machine for solving differential equations. *Journal of the Franklin Institute*, **212** (1931) 447–488.
2. R. Engelking. *General Topology*. Heldermann Verlag, 1989.
3. D.S. Graça and J.F. Costa: Analog computers and recursive functions over the reals. *Journal of Complexity*, **19** (2003) 644–664.
4. E. Griffor, editor. *Handbook of Computability Theory*. North Holland, 1999.
5. D.E. Hyndman. *Analog and Hybrid Computing*. Pergamon Press, 1970.
6. S.C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.
7. J. Mycka and J.F. Costa. Real recursive functions and their hierarchy. *Journal of Complexity* **20** (2005) 835–857.
8. C. Moore. Recursion theory on the reals and continuous time computation. *Theoretical Computer Science* **162** (1996) 23–44.
9. A.V. Oppenheim and A.S. Willsky. *Signals and Systems (2nd edition)* Prentice Hall, 1997.

10. M.B. Pour-El. Abstract computability and its relation to the general-purpose analog computer. *J. American Mathematical Society* **199** (1974) 1–28.
11. C. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics* **20** (1941) 337–354.
12. W. Thompson and P.G. Tait. *Treatise on Natural Philosophy (2nd edition) Part I.* Cambridge University Press, 1880.
13. B.C. Thompson and J.V. Tucker. *Algebraic specification of synchronous concurrent algorithms and architectures.* Research Report CSR 9-91, Department of Computer Science, University College of Swansea, 1991.
14. J.V. Tucker and J.I. Zucker. Computable functions on stream algebras. In: *Proof and Computation: NATO Advanced Study Institute Summer School at Marktoberdorf, 1993*, ed. H. Schwichtenberg. Springer-Verlag (1994) 341–382.
15. J.V. Tucker and J.I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Trans. on Computational Logic*, **5** (2004) 611–668.
16. K. Weihrauch. *Computable Analysis: An Introduction.* Springer-Verlag, 2000.

Computable Analysis

Klaus Weihrauch

Lehrgebiet Theoretische Informatik I,
Berechenbarkeit und Logik,
Fachbereich Informatik,
FernUniversität in Hagen, Informatikzentrum,
58084 Hagen, Germany
klaus.weihrauch@fernuni-hagen.de
http://www.informatik.fernuni-hagen.de/thi1/thi1.html

Computable Analysis studies those functions on the real numbers and related sets which can be computed by machines such as digital computers. It connects two traditionally disjoint fields, namely Analysis/Numerical Analysis on the one hand and Computability/Computational Complexity on the other hand, combining concepts of approximation and of computation. In particular, Computable Analysis supplies an algorithmic foundation of numerical computation. While essentially all computablity models for number functions are equivalent (Church's Thesis) several non-equivalent mathematical models for real number computation are still being discussed.

In this tutorial we mainly concentrate on the representation model of computation (TTE) which is based on a definition of computable real functions introduced by A. Grzegorczyk and D. Lacombe in 1955. In TTE, infinite sequences of symbols (more generally and informally, of rational numbers, rational balls, finite rational step functions etc.) are used as "names" of the points of sets like real numbers, continuous real functions, open subsets of Euclidean space or L_2-functions. Computability is defined by Turing machines which read from infinite files and write to infinite files. We introduce and explain basic concepts and definitions, discuss many examples, and present a number of applications. The following topics will be treated:

- approaches to Computable Analysis (Banach/Mazur, Markov, Grzegorczyk/ Lacombe, Pour-El/Richards, real RAM, Scott domain, Interval Analysis, ...);
- computable real numbers, non-computable real numbers;
- computable real functions, examples and counter-examples;
- representations, reducibility, induced computability, induced continuity;
- admissible representations, main theorem for admissible representations;
- constructions of new representations;
- computability on subsets of Euclidean space (closed, open, compact);
- computability on the space of continuous real functions;
- zero-finding for continuous real functions;
- computational complexity of real functions;
- degrees of discontinuity (Wadge reducibility);

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 530–531, 2005.

– some interesting results (Baire category theorem, Riemann mapping theorem, solution operators of differential equations, computational complexity of Julia sets).

Despite remarkable progress during recent years Computable Analysis is still in it Infancy, still only the surface of what can be done has been scratched. Numerous secrets are waiting for becoming detected. The tutorial is addressed to participants who have some basic knowledge in Analysis as well as in Computability.

The Transfinite Action of 1 Tape Turing Machines

Philip D. Welch

School of Mathematics,
University of Bristol,
Clifton, Bristol, BS8 1TW, UK
p.welch@bristol.ac.uk

Abstract.

- We produce a classification of the pointclasses of sets of reals produced by infinite time turing machines with 1-tape. The reason for choosing this formalism is that it apparently yields a smoother classification of classes defined by algorithms that halt at limit ordinals.
- We consider some relations of such classes with other similar notions, such as arithmetical quasi-inductive definitions.
- It is noted that the action of ω many steps of such a machine can correspond to the double jump operator (in the usual Turing sense): $\boldsymbol{a} \longrightarrow \boldsymbol{a''}$.
- The ordinals beginning gaps in the "clockable" ordinals are admissible ordinals, and the length of such gaps corresponds to the degree of reflection those ordinals enjoy.

1 Introduction

This paper is concerned with exploring the actions of certain models of transfinite time Turing machines. The idea of formulating such a model is due to Hamkins and Kidder, and [1] is the standard reference here. We refer the reader to this article for basic description of these machines. There is some discussion there on relating these machines to other types of "supertasks" (thus computations involving Rapidly Accelerating Machines, Zeno machines, and computations done in Malament-Hogarth spacetimes). We take the view that the infinite time Turing machines are an idealised laboratory for discussing notions of computation involving the transfinite, much as ordinary Turing machines do for ordinary forms of algorithmic computation. The advantage of these infinite time Turing machines is that they may simulate these others, whilst coming uncluttered with any "Thompson-Lamp" like worries about what state the machine is in after a limit number of steps: we simply define a behaviour for them at limit stages of time.

There have been suggestions of other models: a 1-tape version ([1] had three tapes, one for input, one for scratch work, and one for output.) The one defining feature of such machines is that, of course, if they can take transfinite time,

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 532–539, 2005.

they can read and out put infinite strings of $\{0, 1\}$ bits. Such a string we shall identify with a subset of $\mathbb{N}$ or equivalently with a member of Cantor space ${}^{\omega}2$. Such devices work acording to an ordinary Turing program at "ordinary" successor stages of time, but of course one must specify what these machines do at limit stages of ordinal time. A *limit rule* is needed to specify what a cell on the tape contains at limit time λ if it has altered unboundedly in λ. The class of "computable" functions is surprisingly robust if one alters these limit rules within reason, but the pointclasses of what kinds of real numbers is on the tape at particular times can and does vary.

We can relate these "infinite time Turing Machines" (ITTM's) to some other notions that have appeared: the *arithmetical quasi-inductive definitions* (Burgess) [2]; distilled from the notion of sets of integers defined by *Herzberger revision sequences* ([3],[4]; the *partial fixed point semantics* of Kreutzer [5]. In one sense these definitions are all different facets of the same many sided coin: any one such notion can be replicated or simulated by another, and proofs and techniques formulated with one notion are usually translatable to another.

Pointclasses can be defined by functions delimiting the number of steps that an infinite time Turing machine was allowed to take before converging. We investigate these, so to speak, "TIME" classes a little here. The questions these pointclasses raise are really in turn unanswered questions about the action of such machines. [1] mentions the existence of "gaps" in the order types of halting times of machines on integer inputs. The issue of what those gaps were was not resolved. It was also at that time an open question as to whether a machine on an integer input could require longer time, meaning more ordinal stages, to run, than could be coded by any other nachine's output.

There is a further issue of what are these "machine" processes really? In [6] we looked at the "global" set-theoretical properties of these machines and analysed the relationships between halting times and ordinals produced by such machines, and determined exactly what were the decidable sets of integers *etc.* However the actual detailed analysis of what the machines were producing was passed over. What a machine can produce in ω many steps can be concretely given (see Theorem 10 below). One could thus view a universal such machine as a "double jump" operator, (Corollary 12), which can be iterated through the ordinals, with a specific non-monotone limit operation of "eventual value".

Instead of the 3-tape machines of [1], we shall use instead the 1-tape machine model that we proposed in [7]. We feel that the results here about classifying these classes support the use of this model.

There already has been an analysis of 1-tape machines (in [8]) where it was surprisingly shown that 1-tape machines could not replicate all the features of the standard 3-tape machines for functions $f : {}^{\omega}2 \longrightarrow {}^{\omega}2$ (although they could for $f : {}^{\omega}2 \longrightarrow \mathbb{N}$.)

The difference between the machine of [8] and that of [7] is the use of a third symbol besides $0, 1$ a blank (denoted "B") to be interpreted as "undetermined". We enumerate the cells of the tape by $\langle C_i | i < \omega \rangle$ with C_0 being the leftmost one. We let the contents of the i'th cell at time ν be denoted by $C_i(\nu)$. At successor

stages of time, the cells' contents are specified according to the usual Turing machine program finitary rules.

We need to introduce a limit rule to specify cell values at limit times. We declare that a cell of the machine C_i at a limit time μ should have contents $C_i(\mu)$ determined by the contents $C_i(\alpha)$ for $\alpha < \mu$, according to a scheme where $C_i(\mu)$ is a symbol (*i.e.* a 0 or 1 or B) if $C_i(\alpha)$ was 0 (or 1 or B) for all sufficiently large ordinals $\alpha < \mu$; if there has been cofinally in μ a value change, then $C_i(\mu)$ is set to the blank symbol B.

> *One Tape machine limit rule: If μ is a limit ordinal, then the contents of the i'th cell on the (single) tape at time μ, $C_i(\mu)$, is given by:* $(\exists \nu < \mu)(\forall \nu' < \mu)(\nu < \nu' \rightarrow C_i(\nu) = C_i(\nu')) \rightarrow C_i(\mu) = C_i(\nu)$*; otherwise set $C_i(\mu)$ to be a blank.*

Thus if a cell's value has varied cofinally often below μ, we set the value to the "non-determined" value of a blank. The formalism is otherwise similar to that of [1]: at limit times, by fiat, it is in a special limit state q_L viewing cell C_0. At successor steps of time, the action is just as for an ordinary Turing machine: it acts according to its finite program, reading/writing and moving one cell to the left or right.

If we identify the reals $\mathbb{R}$ with Cantor space 2^ω one then has:

Theorem 1. (cf. *[7] Theorem 1) Let $\mathcal{C}$ be the class of functions $F : \mathbb{R} \rightarrow \mathbb{R}$ computable by the Hamkins-Lewis machines of [1], and $\mathcal{C}'$ those of the one-tape machine just specified. Then $\mathcal{C} = \mathcal{C}'$.*

In general we feel that the 1-tape model has conceptual advantages, not just that it provides a smoother theory of the classes P^f as below. The model

- has a "physical" construction that of a normal Turing machine: namely one infinite tape
- it treats 0's and 1's symmetrically at limits;
- the use of a blank to assign truth values at a limit indicating "undetermined" accords with one's perceptions of a process that going through time vacillates cofinally in that limit ordinal;
- allows a "clean" halting process at limits: algorithms that produce an output only can use the "bit" in the single cell C_0 at the beginning of the tape.

The latter may seem somewhat obscure, but it is the feature of the standard model that there are 3 cells observable that creates an "odd" class of sets of reals computable in exactly certain limit times. It should be emphasised that for the vast majority of results, especially those of a more "global" nature, it makes no difference whatsoever which model one uses. It is only in the mechanics of halting, and the results pertaining thereto that can be affected. (See also a discussion in [8] as to which ordinals are "clockable"[1] on 1-tape machines: there

[1] They call an ordinal *clockable* if it is the length of a halting computation on 0 input.

are possibly minor variations here, but the overall picture of the machines, the functions they compute etc, is no different if one takes the [1] or the [7] model.)

We shall define these classes as follows:

Definition 2. *Let* $A \subseteq {}^{\omega}2$.
We say that A is semi-decidable *if there is an (infinite time) Turing machine computable functional* φ_e *so that*
(i) $x \in A$ *if and only if* $\varphi_e(x)\downarrow 1$
(ii) A is decidable *if both A and its complement are semi-decidable.*

By "$\varphi_e(x)\downarrow 1$" we mean that the machine has halted with the first cell of the tape containing a 1; similarly 0 *etc.*

We recall a definition from [1]:

Definition 3. $\lambda^x =_{df} \sup\{\alpha \mid \exists e \varphi_e(x)\downarrow y \wedge y \in WO \wedge rk(y) = \alpha\}$.

Equivalently (and the reader may take this as a definition):

Fact 4. *([9] Theorem 1.1)* λ^x *is the supremum of halting times of any Turing computable function on input x.*

Prior to the last Fact's proof it was thought possible that halting times might have outrun the ordinals producible by such machines. Without the last Fact one could not have proven:

Theorem 5. (Normal form theorem) $\forall e \exists e' \forall x \in {}^{\omega}2$:
$[\varphi_e(x)\downarrow \longrightarrow \varphi_{e'}(x)\downarrow y \in {}^{\omega}2$ *where y is a code for a wellordered computation sequence for* $\varphi_e(x)]$.
The map $e \mapsto e'$ *can be made effective (in the usual sense).*

Implicit in Fact 4 - when taken with the definition of decidable sets of reals [1] - (see the discussion in [7]) is the following characterisation of such sets.

Fact 6. $A \in {}^{\omega}2$ *is decidable if and only if there are* Σ_1 *formulae in the language of set theory* $\varphi_0(v_0), \varphi_1(v_0)$ *so that*

$$x \in A \Longleftrightarrow L_{\lambda^x}[x] \vDash \varphi_0[x] \Longleftrightarrow L_{\lambda^x}[x] \vDash \neg\varphi_1[x]$$

We shall be concerned with classifying certain pointclasses of sets of reals that fall strictly within $\boldsymbol{\Delta^1_2}$.

Suppose we are given any function $f : \mathcal{D} \to \omega_1$ of ordinary Turing degrees to countable ordinals that is definable *via* a Σ_1 formula $\psi(v_0, v_1)$, so that for any (ordinary) Turing degree $[y]_T$ we have $f([y]_T) = \alpha$ iff $L[y] \vDash \psi(y, \alpha)$; then we may define a slice through Δ^1_2 defining a lightface pointclass Γ_0 as follows: $A \in \Gamma_0$ if and only if for some formula $\theta(v_0)$ we have $x \in A \longleftrightarrow L_{f(x)}[x] \vDash \theta(x)$. (A boldface definition would add in a real parameter here to ψ and θ.) How high a rank f has in ${}^{\mathcal{D}}\aleph_1$ modulo the Martin measure (*cf* [10] p386), determines the complexity of the pointclass.

In [11] we were initially motivated by certain questions of Schindler [12] where certain pointclasses P^f were defined that can be seen to fit into the above

general scheme. The pointclasses are strictly within a proper initial segment of Δ^1_2, bounded by the function $f(x) = \lambda^x$ (recalled below.)

We shall define these classes as follows:

Definition 7. *Let $f : \mathcal{D} \longrightarrow \omega_1$ be (standard) Turing invariant. Let $A \subseteq {}^{\omega}2$. We say that $A \in P^f$ if there is a total (infinite time) Turing machine computable functional φ_e so that*
(i) A is decidable by φ_e, that is $x \in A$ if and only if $\varphi_e(x)\downarrow 1$
(ii) $\forall z \in {}^{\omega}2 \quad \varphi_e(z)\downarrow$ in $\leq f(z)$ steps.

By "Turing invariant" we mean that the value of $f(x)$ is the same irrespective of which choice of x from a degree $\boldsymbol{d} \in \mathcal{D}$ is made. Here, in this phrase, we mean the standard notion of Turing recursion, and degree; hereafter we shall use the notions of infinite time Turing recursion only, and shall always mean these, unless otherwise specified.

If the value of $f(x)$ is some constant α then the classes P^f lie strictly within the Borel hierarchy ([12] Lemma 2.7). Recall that ω_1^x denotes the first ordinal not recursive in x and (see [13]) that ω_1^x is also the first *x-admissible ordinal*, beyond ω, where α is x-admissible, if it is the ordinal height of a transitive model of Kripke-Platek set theory containing x. (It is admissible if it is $\varnothing$-admissible.) Thus the smallest transitive set model of $KP+$ Axiom of Infinity, containing x is $L_{\omega_1^x}[x]$. The ordinals λ^x are x-admissible, and enjoy strong reflecting properties (*cf.*[1]).

If we now define $f(x) = \omega_1^x$ then P^f coincides with hyperarithmetic (and so we are really still within the realms of Kleene recursion *e.g.* see [14]). When $f(x) > \omega_1^x$ for all x we then truly enter for the first time the world of sets that are essentially computed by infinite time Turing machines.

Clearly then:

Lemma 8. *If f dominates the function $x \longmapsto \lambda^x$ then P^f equals the class of decidable sets.*

Let f_k be defined as $f_k(x) = \omega_1^x + \omega + k$. As $[\omega_1^x, \omega_1^x + \omega)$ is a gap in the x-clockable ordinals (in either formulation of machine: *cf.* [1] 3.4 and [8] 3.3) thus f_0 is the first one to interest us beyond $g(x) = \omega_1^x$. (In [12] P^{f_0} is denoted P^{++}.)

We classify P^{f_0} as follows. We take Γ to be the pointclass of sets of reals A so that there are formulae $\varphi_0(v_0), \varphi_1(v_0)$, which are

Σ_4 in the language $\mathcal{L}_{\{\dot{\in},\dot{x}\}}$, with the property that

$$\forall x \quad x \in A \Longleftrightarrow L_{\omega_1^x}[x] \vDash \varphi_0(x) \longleftrightarrow \neg\varphi_1(x).$$

In other words A is in "$\Delta_4(L_{\omega_1^x}[x])$".

By the above comments then P^g is the pointclass of sets of reals that are "$\Delta_1(L_{\omega_1^x}[x])$". Here we have:

Theorem 9. $P^{f_0} = \Gamma$.

The theme from the above analysis is that ω many steps of the ITTM can add two levels of definability (in the arithmetical hierarchy) to the tape's contents, and hence the double Turing jump nature of this operation. As a corollary to the method of proof of $\Gamma \subseteq P^{f_0}$ in the above, we may state this as: that:

Theorem 10. *There is an infinite time program P_e so that on 0 input $(0, 0, 0, \ldots)$, after ω many steps the tape contains a code for $Fin =_{df} \{n | W_n$ is finite $\}$ (where $W_n = dom\, \varphi_n$ for φ_n a standard Turing computable function.)*

Fin is complete Σ_2 whence follows our remarks on the double jump of the abstract. To be more precise, as the tape works in 3^ω (rather than 2^ω), for the program P_e under discussion, $Fin = \{n | C_{2n}(\omega) = 1\}$. Fin is thus recursive in $\langle C_k(\omega) | k < \omega \rangle$.

This is result is best possible. More formally put:

Theorem 11. *Let $g = \langle C_k(\omega) | k < \omega \rangle$ code the contents of the tape after some program on 0 input has run for ω steps. Then $g \leq_T 0''$.*

Corollary 12. *If, then, g is the tape's contents after running the program P_e of Theorem 10, then $g \equiv_T Fin \equiv_T 0''$.*

It is also not hard to see from the form of Γ above that if $A \in \mathrm{Diff}(< \omega_1^{ck}, \Sigma_1^1)$ the Hausdorff difference hierarchy for levels below the first non-recursive ordinal, then $A \in P^{f_0}$. (Here ω_1^{ck} is the first non-recursive ordinal.)

Theorem 9 generalises:

Theorem 13. *Let $f(x)$ be an x-admissible ordinal which is uniformly not Π_3-reflecting. (That is, we suppose there is a Π_3 formula $\varphi(\dot{x})$ so that for all $x \in {}^\omega 2$ $L_{f(x)}[x] \vDash \varphi(\dot{x})$ whilst for all $\alpha < f(x)$ $L_{f(x)}[x] \vDash \neg\varphi(\dot{x})$.) Then $P^f = \Delta_4(L_{f(x)}[x])$.*

Let $Bool(\Gamma_1)$ be the class of sets of reals that are Boolean combinations of Γ_1 where Γ_1 is the class of similarly defined $\Sigma_4(L_{\omega_1^x}[x])$ sets of reals.

Our methods show:

Theorem 14. $\bigcup_{k<\omega} P^{f_k} = Bool(\Gamma_1)$.

Let Γ_2 be the $\Delta_6(L_{\omega_1^x}[x])$ definable sets.

Theorem 15. $\Gamma_2 = P^{f_\omega}$

The reader can imagine further extensions. It is to be emphasised that these results on the classes P^f hold only for the one tape model described in some more detail below. For the model from [1], one has that the class P^{f_0} would turn out to be the class of sets A that are differences of two sets in $\Sigma_4(L_{\omega_1^x}[x])$, and their complements. It is *inadequate* in the sense of Moschovakis [15] p.158, not being closed under finite unions or intersections. For the comments on the double jump operation, either model will do. In [8] a comparison of the halting times of 2-valued 1-tape machines and the 2-valued 3-tape machines was made. It was left

open (Question 3.3 of [8]) whether clockables that were not 1-tape clockable (in the 2-valued sense) were those of compound limit length. The following (using the methods here) provides a counterexample, and so the answer is negative (for either type of 1-tape machine).

Theorem 16. *Let γ be the least ordinal that is Π_3 reflecting. Then $\gamma + \omega$ is 3-tape clockable but not 1-tape clockable. However $\gamma + \omega + 1$ is 1-tape clockable.*

In [1] it is proven that no admissible ordinal is clockable: thus every admissible ordinal either starts a gap, or lies within a gap, in the clockable ordinals. We show:

Theorem 17. *If α is an ordinal starting a gap in the clockables, then α is admissible.*

Again, this is for any machine formulation.

References

1. Hamkins, J.D., Lewis, A.: Infinite time Turing machines. Journal of Symbolic Logic **65** (2000) 567–604
2. Burgess, J.: The truth is never simple. Journal for Symbolic Logic **51** (1986) 663–681
3. Herzberger, H.: Notes on naive semantics. Journal of Philosophical Logic **11** (1982) 61–102
4. Herzberger, H.: Naive semantics and the Liar paradox. Journal of Philosophy **79** (1982) 479–497
5. Kreutzer, S.: Partial fixed point logic on infinite structures. In: Annual Conference of the European Association for Computer Science Logic (CSL). Volume 2471 of Lecture Notes in Computer Science., Springer (2002)
6. Welch, P.: Eventually infinite time Turing degrees: infinite time decidable reals. Journal for Symbolic Logic **65** (2000) 1193–1203
7. Welch, P.: Post's and other problems in higher type supertasks. In Räsch, B.L.B.P.T., ed.: Classical and New Paradigms of Computation and their Complexity hierarchies, Papers of the Conference Foundations of the Formal Sciences III. Volume 23 of Trends in logic., Kluwer (2004)
8. Hamkins, J.D., Seabold, D.: Infinite time Turing machines with only one tape. Mathematical Logic Quarterly **47** (2001) 271–287
9. Welch, P.: The length of infinite time Turing machine computations. Bulletin of the London Mathematical Society **32** (2000) 129–136
10. A.Kanamori: The Higher Infinite. Omega Series in Logic. Springer Verlag, New York (1994)
11. Hamkins, J.D., Welch, P.D.: $P^f \neq NP^f$ almost everywhere. Mathematical Logic Quarterly **49** (2003) 536–540
12. Schindler, R.D.: $P \neq NP$ for infinite time Turing machines. Monatsheft für Mathematik **139** (2003) 335–340
13. Barwise, K.: Admissible Sets and Structures. Perspectives in Mathematical Logic. Springer Verlag (1975)

14. K.Hrbacek, S.Simpson: On Kleene degrees of analytic sets. In J.Barwise, H.J.Keisler, K.Kunen, eds.: Proceedings of the Kleene Symposium. Studies in Logic, North-Holland (1980) 347–352
15. Moschovakis, Y.: Descriptive Set theory. Studies in Logic series. North-Holland, Amsterdam (1980)

Complexity of Continuous Space Machine Operations*

Damien Woods[1,**] and J. Paul Gibson[2]

[1] Boole Centre for Research in Informatics and Department of Mathematics, University College Cork, Cork, Ireland
d.woods@cs.ucc.ie
[2] Theoretical Aspects of Software Systems Research Group, Department of Computer Science, National University of Ireland, Maynooth, Ireland

Abstract. We investigate the computational complexity of an optical model of computation called the continuous space machine (CSM). We characterise worst case resource growth over time for each of the CSM's ten operations with respect to seven resource measures. Many operations exhibit unreasonably large growth rates thus motivating restrictions on the CSM, in particular we give a restriction called the $\mathcal{C}_2$-CSM.

1 Introduction

The computational model we study is relatively new and is called the continuous space machine (CSM) [11, 12, 13, 18, 19]. The CSM is inspired by classical Fourier optics and uses complex-valued images, arranged in a grid structure, for data storage. The program also resides in images. The CSM has the ability to perform Fourier transformation, complex conjugation, multiplication, addition, thresholding and resizing of images. It has simple control flow operations and is deterministic. To analyse such a model we define a total of seven complexity measures inspired by real-world resources. For example, spatial resolution corresponds to number of pixels.

A variant of the model with real inputs was previously shown [19] to decide the membership problem for all recursively enumerable languages, and as such is unreasonable in terms of implementation. Here, we build on this work by showing the growth in resource usage for each CSM operation. This leads to a restriction of the CSM that is more suited to the standard tools from analysis of algorithms and complexity theory.

* We thank Tom Naughton for many fruitful discussions and in particular for his collaboration on the CSM definition.
** The first author is funded by the Irish Research Council for Science, Engineering and Technology.

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 540–551, 2005.

2 The CSM

We begin by informally describing the model, this brief overview is not intended to be complete: Detailed definitions and discussions can be found in [18, 19].

Definition 1 (complex-valued image). *A complex-valued image (or simply, image) is a function* $f : [0,1) \times [0,1) \to \mathbb{C}$*, where* $[0,1)$ *is the half-open real unit interval.*

We let $\mathcal{I}$ denote the set of all complex-valued images. $\mathbb{N}^+ = \{1, 2, 3, \ldots\}$ and $\mathbb{N} = \mathbb{N}^+ \cup \{0\}$. For a given CSM M we let $\mathcal{N}$ be a countable set of images that encode M's addresses. Also for a given M there is an address encoding function $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ such that $\mathfrak{E}$ is Turing machine decidable, under some *reasonable*[1] representation of images as words. An address is simply an element of $\mathbb{N} \times \mathbb{N}$.

Definition 2 (CSM). *A CSM is a quintuple* $M = (\mathfrak{E}, L, I, P, O)$*, where*
$\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ *is the address encoding function*
$L = ((s_\xi, s_\eta), (a_\xi, a_\eta), (b_\xi, b_\eta))$ *are the addresses:* sta*, a, and b,*
$I = ((\iota_{1,\xi}, \iota_{1,\eta}), \ldots, (\iota_{k,\xi}, \iota_{k,\eta}))$ *are the addresses of the* k *input images,*
$P = \{(\zeta_1, p_{1,\xi}, p_{1,\eta}), \ldots, (\zeta_r, p_{r,\xi}, p_{r,\eta})\}$ *are the* r *programming symbols and their addresses where* $\zeta_j \in (\{\mathrm{h}, \mathrm{v}, *, \cdot, +, \rho, \mathrm{st}, \mathrm{ld}, \mathrm{br}, \mathrm{hlt}\} \cup \mathcal{N}) \subset \mathcal{I}$*,*
$O = ((o_{1,\xi}, o_{1,\eta}), \ldots, (o_{\ell,\xi}, o_{\ell,\eta}))$ *are the addresses of the* ℓ *output images.*
Each address is an element from $\{0, 1, \ldots, \Xi - 1\} \times \{0, 1, \ldots, \Upsilon - 1\}$ *where* $\Xi, \Upsilon \in \mathbb{N}^+$*. Addresses a and b are distinct.*

Addresses whose contents are not specified by P in a CSM definition are assumed to contain the constant image $f(x, y) = 0$. We interpret this definition to mean that M is (initially) defined on a grid of images bounded by the constants Ξ and Υ, in the horizontal and vertical directions respectively.

In our grid notation the first and second elements of an address tuple refer to the horizontal and vertical axes of the grid respectively, and image $(0, 0)$ is at the bottom left-hand corner of the grid. The images in a grid have the same orientation as the grid. Figure 1 gives the CSM operations in this grid notation. Configurations are defined in a straightforward way as a tuple $\langle c, e \rangle$ where c is an address called the control and e represents the grid contents. In the sequel we write $\widehat{c}$ to mean the image (or instruction) at address c. It is beyond the scope of this paper to give CSM algorithms and so this informal description is sufficient for our analysis in Section 4. For a more thourough introduction, *cf.* [18, 19].

3 Complexity Measures

We want our complexity measures to be straightforward to analyse, while at the same time to be meaningful by reflecting the reality of optical computing. All finite resource bounding functions are from $\mathbb{N}$ into $\mathbb{N}$ and have the usual properties [1].

[1] Other authors have also raised this representation issue for different models, but with similar motivations. See [18] for further discussion.

h	: replace image in a with its horizontal 1D Fourier transform (FT).
v	: replace image in a with its vertical 1D FT.
$*$	: replace image in a with its complex conjugate.
$\cdot$	: multiply (point by point) the two images in a and b. Store result in a.
$+$	: perform a complex addition of a and b. Store result in a.
$\rho \mid z_\ell \mid z_\mathrm{u}$	: $z_\ell, z_\mathrm{u} \in \mathcal{I}$; filter the image in a by amplitude using z_l and z_u as lower and upper amplitude threshold images, respectively.
st $\mid \xi_1 \mid \xi_2 \mid \eta_1 \mid \eta_2$	: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy the image in a into the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and whose top right-hand corner address is (ξ_2, η_2).
ld $\mid \xi_1 \mid \xi_2 \mid \eta_1 \mid \eta_2$	: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy into a the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and top right-hand corner address is (ξ_2, η_2).
br $\mid \xi \mid \eta$	: $\xi, \eta \in \mathbb{N}$; unconditionally branch to the image at address (ξ, η).
hlt	: halt.

Fig. 1. The set of CSM operations, given in our informal grid notation

Definition 3. *The* TIME *complexity of a CSM M is the number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.*

Definition 4. *The* GRID *complexity of a CSM M is the minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.*

From the CSM definition GRID is at least $\Xi\Upsilon$. In previous work [11, 12, 13, 19] the number of grid images remained constant throughout a computation. Here we alter the CSM (by introducing the address encoding function $\mathfrak{E}$) so that GRID may grow over time.

Next we define SPATIALRES. Let a *pixel* λ be a constant complex function defined on a real-valued rectangle with rational endpoints, $\lambda : [0, W) \times [0, H) \to z$ where $z \in \mathbb{C}$; $W, H \in \mathbb{Q}$; $0 < W, H \leqslant 1$; and $[0, W), [0, H) \subset \mathbb{R}$. A *raster image* is an image composed entirely of nonoverlapping pixels, each pixel is of equal height H, equal width W, identical orientation, and arranged into Φ columns and Ψ rows where $\Phi W = 1 = \Psi H$. Let the *spatial resolution* of a raster image be $\Phi\Psi$. Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$. If we choose a reasonable and realistic S then the details of S are not important.

Definition 5. *The* SPATIALRES *complexity of a CSM M is the minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.*

If no such $\Phi\Psi$ exists then M has infinite SPATIALRES complexity.

For AMPLRES complexity consider the function $A : \mathcal{I} \times \mathbb{N}^+ \to \mathcal{I}$,

$$A(f(x, y), \mu) = \left\lfloor |f(x, y)|\mu + \frac{1}{2} \right\rfloor \frac{1}{\mu} \exp(\mathrm{i} \times \arg(f(x, y))), \tag{1}$$

where $|\cdot|$ gives the amplitudes of its image argument, $\arg(\cdot)$ gives the phase angles in the range $(0, 2\pi]$, and the floor operation operates separately on each image value. The value μ is the cardinality of the set of discrete nonzero amplitude values that each complex value in $A(f, \mu)$ can take, per half-open unit interval of amplitude. To aid in the understanding of Equation (1), recall that $f(x, y) = |f(x, y)| \exp(\mathrm{i} \times \arg(f(x, y)))$.

Definition 6. *The* AMPLRES *complexity of a CSM M is the minimum μ such that if each image $f(x, y)$ in the computation of M is replaced by $A(f(x, y), \mu)$ then M computes correctly on all inputs.*

If no such μ exists then M has infinite AMPLRES complexity.

Consider the function $P : \mathcal{I} \times \mathbb{N}^+ \to \mathcal{I}$ defined as

$$P(f(x, y), \mu) = |f(x, y)| \exp\left(\mathrm{i} \left\lfloor \arg(f(x, y)) \frac{\mu}{2\pi} + \frac{1}{2} \right\rfloor \frac{2\pi}{\mu}\right) . \tag{2}$$

The value μ is the cardinality of the set of discrete phase values that each complex value in $P(f, \mu)$ can take.

Definition 7. *The* PHASERES *complexity of a CSM M is the minimum μ such that if each image $f(x, y)$ in the computation of M is replaced by $P(f(x, y), \mu)$ then M computes correctly on all inputs.*

If no such μ exists then M has infinite PHASERES complexity.

Definition 8. *The* DYRANGE *complexity of a CSM M is the ceiling of the maximum of all the amplitude values stored in all of M's images during M's computation.*

Definition 9. *The* FREQ *complexity of a CSM M is the minimum optical frequency such that M computes correctly on all inputs.*

The concept of minimum optical frequency is explained in [19]. If approximations of a FT are sufficient for M, or if M does not execute h nor v, then M requires finite FREQ. If the original (unbounded) definitions of h and v must hold then M requires infinite FREQ. Using the traditional optical methods, any lower bound on SPATIALRES will impose a lower bound on FREQ [19], we should be aware of this in our complexity analysis.

Often we wish to make analogies between space on some well-known model and 'space-like' resources on the CSM. For this purpose we define the following convenient term.

Definition 10. *The* SPACE *complexity of a CSM M is the product of all of M's complexity measures except* TIME.

We argue that this definition is reasonable as it gives an upper bound on the information (e.g. number of bits) stored throughout a CSM computation.

We have defined the complexity of a computation (sequence of configurations) for each measure. We extend this definition to the complexity of a (possibly

Table 1. CSM resource usage after one timestep. For each operation and complexity measure pair, the table entry defines the worst case upper bound on CSM resource usage at TIME $T+1$, in terms of resources used at TIME T: GRID $= G_T$, SPATIALRES $= R_{\mathrm{S},T}$, AMPLRES $= R_{\mathrm{A},T}$, DYRANGE $= R_{\mathrm{D},T}$, PHASERES $= R_{\mathrm{P},T}$ and FREQ $= \nu_T$. "unb." stands for "unbounded". Theorems are cited in parentheses

	GRID	SPATIALRES	AMPLRES	DYRANGE	PHASERES	FREQ
h	G_T	∞ (11)	∞ (11)	∞ (12)	∞ (11)	∞ (11)
v	G_T	∞ (11)	∞ (11)	∞ (12)	∞ (11)	∞ (11)
$*$	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{D},T}$ (14)	ν_T
$\cdot$	G_T	$R_{\mathrm{S},T}$	$(R_{\mathrm{A},T})^2$ (15)	$(R_{\mathrm{D},T})^2$ (16)	$R_{\mathrm{P},T}$ (17)	ν_T
$+$	G_T	$R_{\mathrm{S},T}$	∞ (18)	$2R_{\mathrm{D},T}$ (19)	∞ (20)	ν_T
ρ	unb. (21)	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
st	unb. (21)	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
ld	unb. (21)	unb. (22)	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	unb. (22)
br	G_T (23)	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
hlt	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T

non-final) configuration in the obvious way. Also, we sometimes talk about the complexity of an image, this is simply the complexity of the configuration that the image is in. A more detailed explanation of the complexity measures can be found in [19], including a discussion on defining energy of computations in terms of the above measures.

4 Worst Case CSM Resource Usage

For the case of sequential computation it is usually obvious how the execution of a single operation will effect resource usage. In parallel models, execution of a single operation can lead to large growth in one timestep. For example a multiplication or shift operation in a unit cost parallel model (such as Pratt and Stockmeyer's unrestricted vector machines [16]) can double the length of a binary string in one step. When binary strings are interpreted as numbers, such multiplications and shifts quickly generate large values. Characterising resource growth is useful for proving upper bounds on power and setting model restrictions [1].

In this section we investigate the growth of complexity resources over TIME, with respect to CSM operations. We tackle this question for each operation and complexity measure pair. As expected, under certain operations some measures do not grow at all. Others grow at rates comparable to massively parallel models. By allowing operations like the FT we are mixing the continuous and discrete worlds, hence some measures grow to infinity in one timestep. This gives strong motivation for CSM restrictions and raises some interesting questions.

Table 1 summarises our results; the table defines the value of a complexity measure after execution of an operation (at TIME $T+1$). The complexity of a configuration at TIME $T+1$ is at least the value it was at TIME T, since complexity functions are nondecreasing. Our definition of TIME assigns unit time cost to each

operation, hence we do not have a TIME column. Many entries are immediate from the definitions, otherwise a theorem is cited to the right of the entry.

In the sequel $\langle c, e\rangle_T$ is an an arbitrary CSM configuration at TIME T and $\widehat{c}$ is the instruction pointed to by the program control c. Also, G_T, $R_{\mathrm{S},T}$, $R_{\mathrm{A},T}$, $R_{\mathrm{D},T}$, $R_{\mathrm{P},T}$ and ν_T are the GRID, SPATIALRES, AMPLRES, DYRANGE, PHASERES and FREQ respectively of configuration $\langle c, e\rangle_T$. Our resource growth analysis is worst case, hence we assume that at each computation step we want to preserve all information in each image (however for specific computations this may not be the case). Each of Theorems 11–23 trivially hold if the resources in question are infinite at TIME T, proofs are given only for the non-trivial finite case. We begin with resource usage after operations h or v for a number of complexity measures.

Theorem 11. *(*h/v *&* SPATIALRES*,* AMPLRES*,* PHASERES *and* FREQ*) Let either* $\widehat{c} = \mathrm{h}$ *or* $\widehat{c} = \mathrm{v}$*. Then* $R_{S,T+1} = R_{A,T+1} = R_{P,T+1} = \nu_{T+1} = \infty$*.*

Proof. We give a proof for the non-trivial case where each measure is finite at TIME T. The statement is proved for the measure in question if there is no finite minimum value for that measure at TIME $T + 1$. We use any rectangular step image, such as

$$a(x,y) = \begin{cases} \frac{1}{R_{\mathrm{A},T}}, & \text{if } \frac{1}{2} - \frac{1}{R_{\mathrm{S_{hor}},T}} \leqslant x < \frac{1}{2} \text{ and } \frac{1}{2} - \frac{1}{R_{\mathrm{S_{ver}},T}} \leqslant y < \frac{1}{2}\,, \\ 0, & \text{otherwise}\,. \end{cases}$$

$R_{\mathrm{S_{hor}},T}$ and $R_{\mathrm{S_{ver}},T}$ are the spatial resolutions in the horizontal and vertical directions, respectively. The image $a(x, y)$ is representable with finite SPATIALRES, AMPLRES, PHASERES and FREQ. However its (horizontal or vertical) Fourier spectrum is a sinc function containing an infinite number of spatially separated components and is therefore not representable by finite SPATIALRES nor FREQ. The amplitudes of the peaks in this Fourier spectrum monotonically decrease in value, never reaching zero, and hence are not representable by finite AMPLRES.

Goodman and Silvestri [9] discuss a method of phase quantisation that is equivalent to PHASERES. They prove that phase quantisation in the Fourier domain causes degradation in the resulting inverse FT, in general. In particular they show that the step function can not be perfectly reconstructed from its phase discretised FT, thus we need infinite PHASERES to represent its FT. □

All theorems in this section are a worst case analysis. The previous theorem tells us that applying standard complexity theory to analyse continuous FTs is pointless. Obviously the result is of little relevance to CSMs that do not use the FT, or only use approximations of the FT. Typically in optical setups it will be the case that at some point of the computation, discretisations are introduced.

Theorem 12. *(*h/v *&* DYRANGE*) Let either* $\widehat{c} = \mathrm{h}$ *or* $\widehat{c} = \mathrm{v}$*. Then* $R_{D,T+1} = \infty$*.*

Proof. Take the constant image $a = 1$. The horizontal FT $\mathrm{h}(a)$ has value 0 everywhere except at $x = 0$ where it is a δ function, for all y. Hence there is no finite minimum DYRANGE that bounds the value at $\mathrm{h}(a(0, y))$. A similar argument holds for v, the only difference is that we get the δ function at $\mathrm{v}(a(x, 0))$. □

It is worthwhile noting that restrictions on images (e.g. finite SPATIALRES) enable us to use Rayleigh's theorem [3–page 112] to specify a finite upper bound on the DYRANGE of $\mathrm{h}(a)$ in terms of the complexity of image a [18].

Lemma 13. *Let $z \in \mathbb{C}$, $\mu \in \mathbb{N}^+$ and given P from Equation* (2)*, then*

$$P(z,\mu) \in \left\{ z' \;\middle|\; z' = |z| \exp\left(\mathrm{i}\mu' \frac{2\pi}{\mu}\right), \mu' \in \{1,2,\ldots,\mu\} \right\}.$$

Proof. Let $j = \left\lfloor \arg(z)\frac{\mu}{2\pi} + \frac{1}{2} \right\rfloor$, hence $j \in \mathbb{Z}$. Let $\arg(z)$ have range $0 < \arg(z) \leqslant 2\pi$. By substituting for $\arg(z)$ in j it is clear that $j \in \mu' \cup \{0\}$. Since we are working in radians, $j = 0$ gives the same value in P as $j = \mu$, hence $j \in \mu'$. □

Theorem 14. *(∗ & PHASERES) Let $\widehat{c} = *$. Then $R_{P,T+1} = R_{P,T}$.*

Proof (Sketch). We give a proof for the non-trivial case of finite PHASERES. The $*$ operation affects only image a. By Lemma 13 the set of phase angles in range(a) at TIME T is of the form $\{\Theta \,|\, \Theta = \mu'(2\pi/\mu)\}$. Our notation is in radians hence $n\Theta$, for all $n \in \mathbb{Z}$, is in the above set of μ angles. For the case of complex conjugation, $n = -1$. Thus the set of possible phases in range(a) at TIME $T+1$ is a subset of the set of phases in range(a) at TIME T. For details see [18]. □

Theorem 15. *($\cdot$ & AMPLRES) Let $\widehat{c} = \cdot$, then $R_{A,T+1} = (R_{A,T})^2$.*

Proof. We give a proof for the non-trivial case of finite AMPLRES. The $\cdot$ operation affects only image a. For any $x, y \in [0,1)$, let $z_a = \mathrm{range}(a(x,y))$, $z_b = \mathrm{range}(b(x,y))$. At TIME $T+1$, $a(x,y)$ is replaced with $a'(x,y) = z_a z_b = |z_a||z_b| \exp(\mathrm{i}(\arg(a(x,y)) + \arg(b(x,y))))$. Let $R_{\mathrm{A},T} = \mu$ in Equation (1), the values in a and b at TIME T are of the form

$$A(z,\mu) = \left\lfloor |z|\,\mu + \frac{1}{2} \right\rfloor \frac{1}{\mu} \exp(\mathrm{i} \times \arg(z)).$$

We are interested only in AMPLRES so we ignore the phase term. At TIME $T+1$

$$|A(z_a,\mu)||A(z_b,\mu)| = \left\lfloor |z_a|\,\mu + \frac{1}{2} \right\rfloor \left\lfloor |z_b|\,\mu + \frac{1}{2} \right\rfloor \frac{1}{\mu^2}.$$

We are proving the theorem for the case that AMPLRES is finite, hence we know that at TIME T, $|z_a|$ and $|z_b|$ are rationals, moreover they are of the form $|z_a| = n/\mu$ and $|z_b| = m/\mu$ for some $n, m \in \mathbb{N}$. By substitution we simplify the above expression to get $|A(z_a,\mu)||A(z_b,\mu)| = nm/\mu^2$ In the worst case we require AMPLRES $\mu^2 = (R_{\mathrm{A},T})^2$ to represent the values in image a at TIME $T+1$. □

Theorem 16. *($\cdot$ & DYRANGE) Let $\widehat{c} = \cdot$, then $R_{D,T+1} = (R_{D,T})^2$.*

Proof. We give a proof for the non-trivial case of finite DYRANGE. The $\cdot$ operation affects only image a. Let z_a and z_b be defined as above. If $|z_a| = |z_b| = R_{\mathrm{D},T}$ then at TIME $T+1$ we get the worst case of $R_{\mathrm{D},T+1} = |a(x,y)| = (R_{\mathrm{D},T})^2$. It is easy to see that for all other values of $|z_a|$ and $|z_b|$, $R_{\mathrm{D},T+1} < (R_{\mathrm{D},T})^2$. □

Unlike AMPLRES and DYRANGE, PHASERES is unaffected by multiplication:

Theorem 17. *($\cdot$ & PHASERES) Let $\widehat{c} = \cdot$, then $R_{P,T+1} = R_{P,T}$.*

Proof. We give a proof for the non-trivial case of finite PHASERES. The operation $\cdot$ affects only image a. We will show that the set of possible phases in range(a) at time $T+1$ is a subset of the possible phases in range$(a) \cup$ range(b) at time T. For some (x, y) let $z_a = a(x, y)$ and $z_b = b(x, y)$. By definition $\cdot(z_a, z_b) = |z_a||z_b| \exp(\mathrm{i}(\arg z_a + \arg z_b))$. At TIME T (from Equation (2)), $\arg z_a = \frac{n}{R_{\mathrm{P},T}} 2\pi$ and $\arg z_b = \frac{m}{R_{\mathrm{P},T}} 2\pi$, where $n, m \in \mathbb{N}$. Thus $\cdot(z_a, z_b) = |z_a||z_b| \exp\left(\mathrm{i}\left(\frac{n+m}{R_{\mathrm{P},T}}\right) 2\pi\right)$ which is in the set of possible phase values in range$(a) \cup$ range(b) at TIME T. □

Theorem 18. *(+ & AMPLRES) Let $\widehat{c} = +$. Then $R_{\mathrm{A}_{T+1}} = \infty$.*

Proof. Suppose $R_{\mathrm{A},T} = 1$ and $R_{\mathrm{P},T} = 4$, then let $a(x, y) = \mathrm{i}$ and $b(x, y) = 1$. After the + operation, at time $T+1$, image a has value $a(x, y) = 1 + i = \sqrt{2} e^{\mathrm{i}\frac{1}{4}\pi}$. The CSM requires ∞ AMPLRES to represent the amplitude value $\sqrt{2}$. □

If we restrict PHASERES to be 1 or 2 then we don't meet the worst case scenario described in the previous theorem, we introduce this restriction in Section 5.

Theorem 19. *(+ & DYRANGE) Let $\widehat{c} = +$. Then $R_{D,T+1} = 2R_{D,T}$.*

Proof. The operation + has no effect on SPATIALRES hence without loss of generality we assume that a and b are everywhere constant, $a(x, y) = z_a = r_a e^{\mathrm{i}\Theta_a 2\pi}$ and $b(x, y) = z_b = r_b e^{\mathrm{i}\Theta_b 2\pi}$. Let $z_a = z_b$ and $|z_a| = R_{\mathrm{D},T}$, in this case $z_a + z_b = 2z_a = 2r_a e^{\mathrm{i}\Theta_a 2\pi}$ and hence $R_{\mathrm{D},T+1} = 2R_{\mathrm{D},T}$. In fact this is the worst case since adding any pair of complex values that lie on the origin-centred disk of radius $R_{\mathrm{D},T}$ gives a new complex value on the origin-centred disk of radius $2R_{\mathrm{D},T}$. □

Theorem 20. *(+ & PHASERES) Let $\widehat{c} = +$. Then $R_{P,T+1} = \infty$.*

Proof. We give a proof for the non-trivial case of finite PHASERES. The operation + has no effect on SPATIALRES hence without loss of generality we assume that a and b are everywhere constant. Let $a = 2$ and $b = \mathrm{i} = e^{\mathrm{i}\frac{1}{2}\pi}$. At time $T+1$, $a = \sqrt{5} \exp(\mathrm{i} \tan^{-1}(1/2)\,\pi)$. Niven [14], Corollary 3.12, shows that $(\tan^{-1}(1/2))/\pi$ is irrational, thus we require infinite PHASERES for addition. □

Theorem 21. *(st/ld/ρ & GRID) Let either $\widehat{c} = \mathrm{st}$, $\widehat{c} = \mathrm{ld}$ or $\widehat{c} = \rho$. Then there is no upper bound on the value of G_{T+1}.*

Proof. The address *decoding* function $\mathfrak{E}^{-1} : \mathcal{N} \to \mathbb{N}$ is Turing machine decidable. This is the only specific restriction on $\mathfrak{E}^{-1}$. Thus there is no upper bound on the natural number that an address parameter of *st* maps to. After a st operation we cannot bound GRID in terms of G_T, or any other complexity measure. The same argument holds for ld and ρ. □

The previous theorem highlights the caveat of *reasonableness* in the definition of $\mathfrak{E}$ in Section 2. When we defined $\mathfrak{E}$ we did not wish to restrict the CSM programmer from coming up with a novel $\mathfrak{E}$ suited to her needs. However, for reasonable addressing functions we should expect the growth rate of $\mathfrak{E}^{-1}$, with respect to the ordering on $\mathcal{N}$, to be reasonable. For example, in Section 5 we restrict $\mathfrak{E}$ to being logspace Turing machine computable, which is an agreed notion of reasonableness in parallel complexity theory. As one can imagine, a complicated $\mathfrak{E}$ will leave lots of headaches for the optical engineer who has to implement it. Not only that, we would also have an incomplete complexity analysis of the CSM in question (unless of course we work the growth rate of $\mathfrak{E}$ into our analysis). The same remark applies to the next theorem.

Theorem 22. *(*ld *&* spatialRes/freq*) Let* $\widehat{c} =$ ld*. Then there is no upper bound on the value of* $R_{S,T+1}$ *nor* ν_T*.*

Proof. After a ld operation with parameters $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$, image a has spatialRes $R_{\mathrm{S},T+1} = R_{\mathrm{S},T}(\xi_2 - \xi_1 + 1)(\eta_2 - \eta_1 + 1)$. From Theorem 21 there is no upper bound on the growth of $\mathfrak{E}^{-1}$. Thus there is no upper bound on the *ld* parameters. After a *st* operation there is no upper bound on spatialRes in terms of $R_{\mathrm{S},T}$, or any other complexity measure. Analogously, for freq the upper bound is in terms of $\mathfrak{E}^{-1}$ rather than any of the complexity measures. □

If we have agreed upon a reasonable (bound on) $\mathfrak{E}$, then it is straightforward to derive an upper bound on spatialRes and freq at time $T + 1$.

Even though br has address parameters, the previous arguments do not apply.

Theorem 23. *(*br *&* grid/freq*) Let* $\widehat{c} =$ br*. Then* $G_{T+1} = G_T$*.*

Proof. From the definition of a CSM configuration [19] the control must always be inside the initial (time 1) grid. Branching outside the current grid will always result in an undefined computation, hence *br* does not increase grid. □

5 $\mathcal{C}_2$-CSM

Motivated by a desire to apply standard complexity theory tools to the model, we define a restricted class of CSM.

Definition 24 ($\mathcal{C}_2$-CSM). *A* $\mathcal{C}_2$*-CSM is a CSM whose computation* time *is defined for* $t \in \{1, 2, \ldots, T(n)\}$ *and has the following restrictions:*

- *For all* time t *both* amplRes *and* phaseRes *have constant value of 2.*
- *For all* time t*,* space *is bounded above by* $c_1 2^{3t} + c_2$ *for constants* c_1 *and* c_2 *that depend on the program and the input, and* space *is redefined to be the product of all complexity measures except* time *and* freq*.*
- *Operations* h *and* v *compute the discrete FT (DFT) in the horizontal and vertical directions respectively.*
- *Given some* reasonable *binary word representation of the set of addresses* $\mathcal{N}$*, the address encoding function* $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ *is decidable by a logspace Turing machine.*

We have replaced the FT with the DFT [3]. FREQ is now solely dependent on SPATIALRES (rescaling the Fourier spectrum by changing FREQ is no longer necessary); hence FREQ is not an interesting complexity measure for $\mathcal{C}_2$-CSMs. The DFT is defined over a ring. Since DYRANGE is bounded, and AMPLRES and PHASERES are constant, we satisfy the definition of the DFT. The SPACE restriction is not unique to our model, such restrictions can be found in [15, 7].

In Section 2 we stated that address encoding functions should be Turing machine computable, here we strengthen this condition. At first glance sequential logspace computability may perhaps seem like a strong restriction, but in fact it is quite weak. From an optical implementation point of view it should be the case that $\mathfrak{E}$ is not complicated, otherwise we cannot assume fast addressing. Other (sequential/parallel) models usually have a very restricted 'addressing function': in most cases it is simply the identity function on $\mathbb{N}$. Without an explicit or implicit restriction on the computational complexity of $\mathfrak{E}$, finding non-trivial upper bounds on the power of $\mathcal{C}_2$-CSMs is impossible as $\mathfrak{E}$ could encode an arbitrarily complex halting Turing machine. As a weaker restriction we could give a specific $\mathfrak{E}$. However, this restricts the generality of the model and prohibits the programmer from developing novel, reasonable, addressing schemes.

A $\mathcal{C}_2$-CSM resource usage table would contain no "∞" or "unbounded" entries. In further work we will give exact characterisations of the power of this model.

6 Discussion

We have analysed the growth of CSM complexity measures with respect to its operations over TIME. Table 1 shows that many variations on the CSM can not be analysed if we restrict ourselves to the standard tools from complexity theory.

The results in this paper are independent of any particular data representations or program restrictions. If we restrict ourselves to certain (continuous or discrete) data representations then clearly we change the properties of computations and can reduce the upper bounds on resource growth. Another way to restrict the model is to place restrictions on the syntactic structure of programs.

Earlier CSM versions [11, 12, 13, 19] used constant GRID. The function $\mathfrak{E}$ allows GRID to be a more useful complexity resource (see [18] for further remarks).

Table 1 describes growth in complexity if inputs are finite. The irrational values that give rise to the infinities in Table 1 are computable reals (say, in the sense of [17]). It would be interesting to analyse this aspect of the model by making use of results from the framework of real recursive function theory [10, 4, 5] or other approaches to analog or real computation [2, 17]. There has been little work towards a parallel complexity theory for analog computation, this would be interesting future work.

The results from this paper are not only interesting from a computational complexity viewpoint, but from a physical viewpoint also. For example Goodman [9] studies PHASERES in the same way we do, and is motivated by practical concerns (reconstructing digital holograms). The $\mathcal{C}_2$-CSM is more realistic than

the CSM in terms of optical implementation; many optical information processing devices are pixellated (e.g. liquid-crystal displays and digital cameras) and operate over a finite set of grey levels [8]. Positive and negative rationals are routinely represented in optical architectures [6]. Clearly, the SPACE limitation decreases the difficulty of implementation.

This work is a starting point for developing CSM restrictions; in particular we defined the $\mathcal{C}_2$-CSM. Any restriction will exhibit resource growth less than or equal to that given by Table 1. Interesting future work would be to characterise the power of such restrictions. In further publications we will exactly characterise standard sequential and parallel complexity classes in terms of the $\mathcal{C}_2$-CSM. For example, we will show that the $\mathcal{C}_2$-CSM satisfies the parallel computation thesis [1, 7, 15] and that the class NC is characterised in terms of the $\mathcal{C}_2$-CSM [18].

References

1. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity, volumes I and II.* EATCS Monographs on Theoretical Computer Science. Springer, Berlin, 1988.
2. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and real computation.* Springer, New York, 1997.
3. R. N. Bracewell. *The Fourier transform and its applications.* Electrical and electronic engineering series. McGraw-Hill, second edition, 1978.
4. M. L. Campagnolo. *Computational Complexity of Real Valued Recursive Functions and Analog Circuits.* PhD thesis, Universidade Técnica de Lisboa, 2001.
5. D. da Silva Graça. The general purpose analog computer and recursive functions over the reals. Master's thesis, IST, Universidade Técnica de Lisboa, 2002.
6. D. G. Feitelson. *Optical Computing.* MIT Press, 1988.
7. L. M. Goldschlager. A universal interconnection pattern for parallel computers. *Journal of the ACM*, 29(4):1073–1086, Oct. 1982.
8. J. W. Goodman. *Introduction to Fourier optics.* McGraw-Hill, New York, second edition, 1996.
9. J. W. Goodman and A. M. Silvestri. Some effects of Fourier domain phase quantization. *IBM Journal of research and development*, 14:478–484, Sept. 1970.
10. C. Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44, Aug. 1996.
11. T. J. Naughton. Continuous-space model of computation is Turing universal. In S. Bains and L. J. Irakliotis, editors, *Critical Technologies for the Future of Computing*, Proceedings of SPIE vol. 4109, San Diego, California, Aug. 2000.
12. T. J. Naughton. A model of computation for Fourier optical processors. In R. A. Lessard and T. Galstian, editors, *Optics in Computing 2000*, Proc. SPIE vol. 4089, pages 24–34, Quebec, Canada, June 2000.
13. T. J. Naughton and D. Woods. On the computational power of a continuous-space optical model of computation. In M. Margenstern and Y. Rogozhin, editors, *Machines, Computations and Universality: Third International Conference*, volume 2055 of *LNCS*, pages 288–299, Chişinău, Moldova, May 2001. Springer.
14. I. Niven. *Irrational numbers*, volume 11 of *The Carus Mathematical Monographs.* The Mathematical Association of America, Wiley, 1956.
15. I. Parberry. *Parallel complexity theory.* Wiley, 1987.

16. V. R. Pratt and L. J. Stockmeyer. A characterisation of the power of vector machines. *Journal of Computer and Systems Sciences*, 12:198–221, 1976.
17. K. Weihrauch. *Computable Analysis: An Introduction*. Springer, Berlin, 2000.
18. D. Woods. *Computational complexity of an optical model of computation*. PhD thesis, National University of Ireland, Maynooth, 2004. Submitted.
19. D. Woods and T. J. Naughton. An optical model of computation. *Theoretical Computer Science*, 2005. In print.

Computable Analysis of a Non-homogeneous Boundary-Value Problem for the Korteweg-de Vries Equation

Ning Zhong

Department of Mathematical Sciences,
University of Cincinnati,
Cincinnati, OH 45221, USA

This note is concerned with computability of the solution operator associated with the wave maker problem for the classical Korteweg-de Vries equation. The Korteweg-de Vries equation is one of several non-linear partial differential equations which have been most intensively studied in the past fifty years due to its physical and mathematical importance.

The problem is usually posted in the form of the following initial-boundary-value problem (henceforth, the abbreviation "IBVP" will stand for "initial-boundary-value problem")

$$\begin{cases} u_t + u_x + uu_x + u_{xxx} = 0, \quad t, x \geq 0, \\ u(x,0) = \varphi(x), \ u(0,t) = h(t). \end{cases} \tag{1}$$

The IBVP arises as a model whenever waves determined at an entry point propagate into a patch of a medium for which disturbances are governed approximately by the Korteweg-de Vries equation, for example, in modeling near-shore zone motions generated by waves propagating from deep water, or when modeling the effect in a channel of a wave maker mounted at one end. In this conception, the wave profile in the channel is imagined to be determined everywhere at a given instant of time with a wave maker mounted at one end of the channel and the corresponding solution models the further wave motion.

Classically the IBVP (1) is known to be well-posed for initial- and boundary-data in certain Sobolev spaces (see, for example, [2]). Because of well-posedness, the problem (1) defines a nonlinear continuous map from the spaces where the auxiliary data are drawn to the space of solutions. A natural question arises: Is this nonlinear map computable? This is one of the open problems raised by Pour-El/Richards in their 1989 book "Computability in Analysis and Physics". The pure initial value problem posed on the whole line $\mathbb{R}$ was shown to be computable in the space $H^s(\mathbb{R})$ for any integer $s \geq 3$ by Weihrauch-Zhong [9]. The present note will provide an affirmative answer to the initial-boundary-value problem (1) by making use of modern methods for the study of nonlinear dispersive wave equations. Speaking technically, for any $T > 0$ the solution map: $H^3(\mathbb{R}^+) \times H^{\frac{4}{3}}(0,T) \to C([0,T]; H^3(\mathbb{R}^+))$ is computable for initial data φ in the space $H^3(\mathbb{R}^+)$ and boundary data h in the space $H^{\frac{4}{3}}(0,T)$ satisfying the

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 552–561, 2005.

compatibility condition $\varphi(0) = h(0)$ with respect to canonical representations of $H^3(\mathbb{R}^+)$, $H^{\frac{4}{3}}(0,T)$, and $C([0,T];H^3(\mathbb{R}^+))$.

The crux of the modern analysis of nonlinear dispersive equations is the linear estimates. For example, the estimates established for the linear problem

$$\begin{cases} u_t + u_x + u_{xxx} = 0, \quad t, x \geq 0, \\ u(x,0) = \varphi(x),\ u(0,t) = h(t), \end{cases} \tag{2}$$

associated to the problem (1) in the work of Bourgain, Kenig-Ponce-Vega, and Bona-Sun-Zhang [3, 4, 6, 2] made it possible to apply the contraction mapping principle to establish directly the well-posedness of the IBVP (1).

To study computability of the solution operator of the IBVP (1), the linear estimates play the same central role as they do in the study of classical well-posedness. The problem (1) is nonlinear and has no explicit solution formula. This implies that the best possible way to compute future wave motions, if the computation is possible, is to compute their approximations with arbitrary precision on computers, provided sufficiently precise information is given on the initial- and boundary-data. Two preconditions are necessary for such computations: The solutions can be approximated by functions computable on digital machines (called finite-objects) and it is possible to preassign the precision of approximations. The linear estimates make it possible to establish a predefined rate of approximation.

The plan of the present note is as follows. Section 1 is devoted to the introduction of certain codings on Sobolev functions. Computations on Sobolev functions are performed on their codes on Type-two Turing machines. Section 2 begins with establishing computability of the solution operators of the associated linear problems. Armed with these computable linear estimates, the main result is then set down. Due to page limit, proofs are either omitted or sketchy.

1 Encoding Sobolev Functions

The computational model used in this note is the Type-Two-Theory of Effectivity, or TTE for abbreviation, developed by Weihrauch and others [8]. In TTE model, computations on functions are performed on digital machines via their names (codes) which are finite or infinite sequences of finite words. In this section, a coding for Sobolev functions is introduced.

Speaking roughly, in TTE model, a real number or a real function is computable if it can be approximated with arbitrary precision at a predefined rate by a sequence of finite-objects produced by a machine. Such a sequence is called a name or a code of the real number or the real function. Examples of finite-objects include rational numbers and rational polynomials. A map $T : X \to Y$ from one function space to another is computable if there is a machine which computes a sequence of finite-object approximations to $T(x)$ arbitrarily well on a given sequence of finite-object approximations to x.

More precisely, let Σ be a sufficiently large finite alphabet, Σ^* the set of finite words over Σ with the discrete topology, and Σ^ω the set of infinite words over Σ

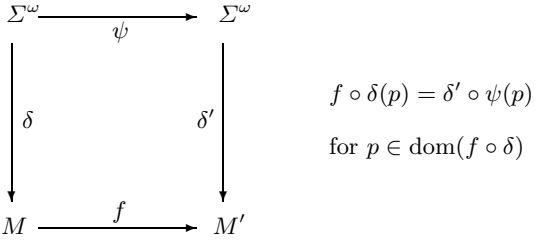


Fig. 1. ψ is a (δ, δ')-realization of f

with the Cantor topology. The symbol $\langle\ \rangle$ denotes various tupling functions. In particular, the infinite tupling on Σ^ω is defined by $\langle p_0, p_1, p_2, \cdots\rangle(\langle k, n\rangle) = p_k(n)$ $(p_k \in \Sigma^\omega)$ for all $k, n \in \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers, including 0.

In TTE, Turing computability is extended from finite words $w \in \Sigma^*$ to infinite words $p = (a_0 a_1 a_2 \ldots) \in \Sigma^\omega$. A Type-2 Turing machine T computes a function $f_T : \Sigma^\omega \to \Sigma^\omega$ (f_T might be partial) in the following way: $f_T(p) = q$ if and only if the machine T reads the input sequence $p = (a_0 a_1 a_2 \ldots)$ symbol by symbol from the left to the right, computes (in the same way as ordinary Turing machines), and writes the output sequence $q = (b_0 b_1 b_2 \ldots)$ symbol by symbol from the left to the right. A function $f : \Sigma^\omega \to \Sigma^\omega$ is computable if it is the input-output function of a Type-2 Turing machine.

A notation (representation) of a set M is a surjective map $\delta : \Sigma^* \to M$ $(\delta : \Sigma^\omega \to M)$. Via δ the elements of M are encoded by finite or infinite words. For any $x \in M$, if $\delta(p) = x$, then p is called a δ-name or δ-code of x; x is called δ-computable if p is computable in Σ^ω.

For a set M with a notation or a representation computations on M are defined by means of computations on names from Σ^* or Σ^ω which can be performed by ordinary or Type-2 Turing machines. If δ and δ' are representations of M and M', respectively, then a function $f : M \to M'$ is called (δ, δ')-continuous (-computable) if there exists a continuous (computable) function $\psi : \Sigma^\omega \to \Sigma^\omega$ such that the diagram in Fig. 1 commutes. In other words, f is (δ, δ')-computable if there is a machine which computes a δ'-name of $f(x)$ when given a δ-name of x as input. The concepts can be generalized straightforwardly to $(\delta_1, \ldots, \delta_n, \delta_0)$-realizations of functions $f :\subseteq M_1 \times \ldots \times M_n \to M_0$.

Two representations of a set M are called equivalent if and only if they induce the same kind of computability on M.

For any representations δ of M and δ' of M', there is a canonical representation $[\delta \to \delta']$ of the set of all (δ, δ')-continuous functions $f : M \to M'$. If both M and M' are topological T_0-spaces with countable bases and both δ and δ' are *admissible representations*, then $f : M \to M'$ is continuous if and only if it is (δ, δ')-continuous. In this case, $[\delta \to \delta']$ is a representation of $C(M; M')$, the set of all continuous functions from M to M' (see [8] for more details).

Next attention is turned to Sobolev functions. We start by laying out definition for the Sobolev functions defined on an interval $I \subseteq \mathbb{R}$, where I stands for (a, b), $\mathbb{R}^+$ or $\mathbb{R}$.

Definition 1. *(a) Assume m is a non-negative integer. The Sobolev space $H^m(I)$ consists of all locally finitely integrable functions $f : I \to \mathbb{R}$ such that the kth order weak derivative $D^k f$ exists and $D^k f \in L^2(I)$ (i.e. $D^k f$ is square integrable over I) for all integers $0 \le k \le m$. $H^m(I)$ is a Banach space with the norm*

$$||f||_{H^m(I)} = \left(\sum_{k \le m} \int_I |D^k f|^2 dx \right)^{1/2}.$$

(b) For $s \ge 0$, write $s = m + \theta$ where $0 \le \theta < 1$ and m is a non-negative integer. Thus $m = [s]$, the greatest integer in s. For $f \in C^\infty(I) \cap H^m(I)$, where $C^\infty(I)$ is the set of all infinitely differentiable functions defined on I, define a function $J_x^s f$ by

$$J_x^s f(x) = \begin{cases} |f^{(m)}(x)| & \text{if } \theta = 0, \\ \left(\int_I \tau^{-(2\theta+1)} |f^{(m)}(x+\tau) - f^{(m)}(x)|^2 d\tau \right)^{1/2} & \text{if } \theta > 0 \end{cases}$$

for any $x \in I$. Because $f^{(m)}$ is smooth and an $L^2(I)$-function and $\theta < 1$, $J_x^s f(x)$ is finite for all x. The quantity

$$||f||^2_{H^s(I)} = ||f||^2_{L^2(I)} + ||J_x^s f||^2_{L^2(I)} \tag{3}$$

defines a norm on $C^\infty(I) \cap H^m(I)$ and the completion of this space in the norm (3) is denoted by $H^s(I)$.

There are several natural ways, different yet equivalent, to encode Sobolev functions by sequences of finite or infinite words. Among them the coding $\delta_{H^s(I)}$ defined below is particularly user-friendly for the purpose of this note. This coding is based on a representation of the Schwartz space $\mathcal{S}(\mathbb{R})$. The following is a brief review on $\mathcal{S}(\mathbb{R})$.

Definition 2. *Let $\mathcal{S}(\mathbb{R})$ be the Schwartz space defined by*

$$\mathcal{S}(\mathbb{R}) = \{\phi \in C^\infty(\mathbb{R}) : \forall \alpha, \beta \in \mathbb{N}, \sup_{x \in \mathbb{R}} |x^\alpha \phi^{(\beta)}(x)| < \infty\}.$$

Define the metric d_S by

$$d_S(\phi, \varphi) = \sum_{\alpha,\beta=0}^{\infty} 2^{-\langle \alpha,\beta \rangle} \frac{||\phi - \varphi||_{\alpha,\beta}}{1 + ||\phi - \varphi||_{\alpha,\beta}}, \quad \forall \phi, \varphi \in \mathcal{S}(\mathbb{R}),$$

where $||\phi||_{\alpha,\beta} := \sup_{x \in \mathbb{R}} |x^\alpha \phi^{(\beta)}(x)|$. The space $(\mathcal{S}(\mathbb{R}), d_S)$ is a complete separable metric space. Moreover, the set $\mathcal{P}^*$ of "smoothly rationally truncated" polynomials with rational coefficients is dense in $\mathcal{S}(\mathbb{R})$ (see, for example, Weihrauch-Zhong [9]). The functions in $\mathcal{P}^*$ are used as basic computational objects for computations on Schwartz functions.

Let $I \subseteq \mathbb{R}$ be an interval and let $\mathcal{P}^*|_I$ denote the set of restrictions to I of functions in $\mathcal{P}^*$ and $\mathcal{S}(\mathbb{R})|_I$ the set of restrictions to I of functions in $\mathcal{S}(\mathbb{R})$. Then $\mathcal{P}^*|_I$ is dense in $\mathcal{S}(\mathbb{R})|_I$.

Definition 3. *Let $\delta_I : \Sigma^\omega \to \mathcal{S}(\mathbb{R})|_I$ be the representation of $\mathcal{S}(\mathbb{R})|_I$ defined as follows: for any $\phi \in \mathcal{S}(\mathbb{R})|_I$ and any infinite word $p = (p_0 p_1 \ldots) \in \Sigma^\omega$, $\delta_I(p) = \phi$ if each p_j encodes an element $P_j \in \mathcal{P}^*|_I$ such that $d_S(\phi, P_j) \leq 2^{-j}$ for all $j \in \mathbb{N}$.*

The following definition is sound, for the set of restrictions to $\mathbb{R}^+$ or to $(0,T)$ of functions in $\mathcal{S}(\mathbb{R})$ is dense in $H^s(\mathbb{R}^+)$ or $H^s(0,T)$, respectively (see [1]).

Definition 4. *For $s \geq 0$ and $T > 0$, let I denote either $\mathbb{R}^+$ or $(0,T)$. The representation $\delta_{H^s(I)} : \Sigma^\omega \to H^s(I)$ is defined as follows: for any $f \in H^s(I)$ and any $p \in \Sigma^\omega$, p is a $\delta_{H^s(I)}$-name of f, i.e. $\delta_{H^s(I)}(p) = f$, if p is an infinite tupling $p = \langle p_0, p_1, p_2, \ldots \rangle$ such that each p_j encodes a Schwartz function $\delta_I(p_j)$ in $\mathcal{S}(\mathbb{R})|_I$ and $||\delta_I(p_j) - f||_{H^s(I)} \leq 2^{-j}$ for all $j \in \mathbb{N}$.*

Let ρ be the standard Cauchy representation of real numbers. That is, for any real number x and any infinite word $p = (p_0 p_1 p_2 \ldots) \in \Sigma^\omega$, $\rho(p) = x$ if each p_j encodes a rational number r_j and $|x - r_j| \leq 2^{-j}$ for all $j \in \mathbb{N}$. Since ρ and $\delta_{H^s(\mathbb{R}^+)}$ are admissible representations, $[\rho \to \delta_{H^s(\mathbb{R}^+)}]$ is a representation of $C(I; H^s(\mathbb{R}^+))$, where $C(I; H^s(\mathbb{R}^+))$ is the set of all continuous functions from I to $H^s(\mathbb{R}^+)$.

2 Computing Solutions of Initial-Boundary Value Problems

We make use of a solution formula for the non-homogeneous linear equation associated with the IBVP (1) and linear estimates established in the work of Bona-Sun-Zhang [2]. In this section, propositions will be used for classical results while lemmas are reserved for results related to computability.

Proposition 5. *The solution, $S_L(\varphi, h, f) = u$, of the non-homogeneous linear initial-boundary value problem*

$$\begin{cases} u_t + u_x + u_{xxx} = f, \quad t, x \geq 0, \\ u(x,0) = \varphi(x), \ u(0,t) = h(t), \end{cases} \tag{4}$$

where φ and h are assumed to satisfy the compatibility condition $\varphi(0) = h(0)$, may be written as

$$\begin{aligned} u(x,t) = W_c(t)(\varphi(x) - e^{-x}\varphi(0)) + \int_0^t W_c(t-\tau)\left(f(x,\tau) + 2e^{-x-\tau}h(0)\right)d\tau \\ + \left[W_b(t)(h(t) - e^{-t}h(0)\right](x) + e^{-x-t}h(0), \end{aligned} \tag{5}$$

where $W_c(t)\varphi(x) = \sum_{j=0}^{2}\left(U_j^+(t)\varphi(x) + \overline{U_j^+(t)\varphi(x)}\right)$ with

$$U_0^+(t)\varphi(x) = \frac{1}{2\pi}\int_1^\infty e^{i\mu^3 t - i\mu t}\int_0^\infty e^{i\mu(x-\xi)}\varphi(\xi)d\xi d\mu,$$

$$U_1^+(t)\varphi(x) = \frac{1}{2\pi}\int_1^\infty e^{i\mu^3 t - i\mu t}e^{-(\frac{i\mu+\sqrt{3\mu^2-4}}{2})x}\int_0^\infty e^{-i\mu\xi}\varphi(\xi)d\xi d\mu,$$

$$U_2^+(t)\varphi(x) = \frac{1}{2\pi i}\int_1^\infty e^{-\mu^3 t - \mu t}e^{-(\frac{\mu-i\sqrt{3\mu^2+4}}{2})x}\int_0^\infty e^{-\mu\xi}\varphi(\xi)d\xi d\mu,$$

and $[W_b(t)h](x) = [U_b(t)h](x) + \overline{[U_b(t)h](x)}$ *with*

$$[U_b(t)h](x) = \frac{1}{2\pi}\int_1^\infty e^{i\mu^3 t - i\mu t} e^{-(\frac{\sqrt{3\mu^2-4}+i\mu}{2})x}(3\mu^2-1)\int_0^\infty e^{-(\mu^3 i - i\mu)\xi}h(\xi)d\xi d\mu.$$

for $x, t \geq 0$. *Here* $\overline{U}$ *denotes the complex conjugate of* U. □

Lemma 6. *For any computable real number* $T > 0$, *the solution map* $S_L : \mathcal{S}(\mathbb{R})|_{\mathbb{R}^+} \times \mathcal{S}(\mathbb{R})|_{(0,T)} \times C([0,T];\mathcal{S}(\mathbb{R})|_{\mathbb{R}^+}) \to C([0,T];\mathcal{S}(\mathbb{R})|_{\mathbb{R}^+})$, $\varphi, h, f \mapsto u$, *of the linear problem (4) is* $(\delta_{\mathbb{R}^+}, \delta_{(0,T)}, [\rho \to \delta_{\mathbb{R}^+}], [\rho \to \delta_{\mathbb{R}^+}])$*-computable.* □

Considered next is the fully nonlinear initial-boundary-value problem (1)

$$\begin{cases} u_t + u_x + uu_x + u_{xxx} = 0, \quad t, x \geq 0, \\ u(x,0) = \varphi(x),\ u(0,t) = h(t) \end{cases}$$

for the KdV equation. Here is the main theorem of this note.

Theorem 7. *For any computable real number* $T > 0$, *the nonlinear solution map* $S : H^3(\mathbb{R}^+) \times H^{4/3}(0,T) \to C([0,T];H^3(\mathbb{R}^+))$, $(\varphi, h) \mapsto u$, *where* $\varphi(0) = h(0)$ *and* u *is the unique solution of the problem (1), is* $(\delta_{H^3(\mathbb{R}^+)}, \delta_{H^{4/3}(0,T)}, [\rho \to \delta_{H^3(\mathbb{R}^+)}])$*-computable.*

Apparently, by the solution formula (5) of the non-homogeneous linear problem (4), if set $f = -uu_x$ in (4), then the IBVP (1) is equivalent to the following integral equation

$$u = L_I(\varphi) + L_B(h) - N(u,u), \tag{6}$$

where

$$L_I(\varphi)(t)(x) = W_c(t)(\varphi(x) - e^{-x}\varphi(0)) + \varphi(0)\left[e^{-x-t} + 2\int_0^t W_c(t-\tau)e^{-x-\tau}d\tau\right],$$
$$L_B(h)(t)(x) = W_b(t)\left[h(t) - e^{-t}h(0)\right](x),\ \textit{and}$$
$$N(u,v)(t)(x) = \int_0^t W_c(t-\tau)\left(uv_x\right)(x,\tau)d\tau.$$

The local well-posedness of the problem (1) may be proved by showing that the iteration

$$(*) \qquad \begin{cases} u_0(t) = L_I(\varphi)(t) + L_B(h)(t), \\ u_{j+1}(t) = u_0(t) - N(u_j, u_j)(t) \end{cases}$$

is a contraction near $t = 0$, based on several linear estimates. The following seminorms are used in presenting these linear estimates. For any $s \geq 0$, $T > 0$ and any function $w \equiv w(x,t) : \mathbb{R}^+ \times [0,T] \to \mathbb{R}$, define

$$\Lambda_{1,s}^T(w) \equiv \sup_{0\le t\le T} ||w(\cdot,t)||_{H^s(\mathbb{R}^+)},$$

$$\Lambda_{2,s}^T(w) \equiv \left(\sup_{x\in\mathbb{R}^+}\int_0^T |J_x^{s+1}w(x,t)|^2 dt\right)^{1/2},$$

$$\Lambda_{3,s}^T(w) \equiv \sup_{x\in\mathbb{R}^+} ||w(x,\cdot)||_{H^{(s+1)/3}(0,T)} + \sup_{x\in\mathbb{R}^+} ||D_x w(x,\cdot)||_{H^{s/3}(0,T)},$$

$$\Lambda_4^T(w) \equiv \left(\int_0^T \sup_{x\in\mathbb{R}^+} |D_x w(x,t)|^4 dt\right)^{1/4},$$

$$\Lambda_5^T(w) \equiv \left(\int_0^T \sup_{t\in[0,T]} |w(x,t)|^2 dx\right)^{1/2}.$$

In addition, let

$$\lambda_{T,s}(w) = \max\{\Lambda_{1,s}^T(w), \Lambda_{2,s}^T(w), \Lambda_{3,s}^T(w)\} + \Lambda_4^T(w) + \Lambda_5^T(w).$$

For any $T > 0$ and $s \in [3/4, 3]$, let $X_{T,s}$ be the collection of all functions $u \in C([0,T]; H^s(\mathbb{R}^+))$ satisfying $\lambda_{T,s}(u) < \infty$. For $u \in X_{T,s}$, define its norm $||u||_{X_{T,s}}$ as $||u||_{X_{T,s}} = \lambda_{T,s}(u)$.

The following proposition summarizes the linear estimates established in the work of Bona-Sun-Zhang. These linear estimates are central in showing that the iteration $(*)$ is a contraction.

Proposition 8. *Let $T > 0$ and $s \in [1,3]$ be given. There exists a constant C depending only on T and s such that for any $0 \le \theta \le T$,*

(1) $||L_I(\varphi)||_{X_{\theta,s}} \le C||\varphi||_{H^s(\mathbb{R}^+)}$ for any $\varphi \in H^s(\mathbb{R}^+)$;
(2) $||L_B(h)||_{X_{\theta,s}} \le C||h||_{H^{(s+1)/3}(0,T)}$ for any $h \in H^{(s+1)/3}(0,T)$;
(3) $||N(u,v)||_{X_{\theta,s}} \le C(\theta^{1/2}+\theta^{1/4})||u||_{X_{\theta,s}}||v||_{X_{\theta,s}}$ for any $u, v \in X_{\theta,s}$. □

Lemma 9. *If both T and s are computable real numbers, then the constant C in Proposition 8 is also computable and it can be computed from (codes of) T and s.* □

Proposition 10. *Let $T > 0$ be given. Then there exists a $T^* \in (0,T]$ depending only on $||\varphi||_{H^3(\mathbb{R}^+)} + ||h||_{H^{4/3}(0,T)}$ such that the iteration $(*)$ is a contraction on $X_{T^*,3}$.*

Proof. Let $r = 2C(||\varphi||_{H^3(\mathbb{R}^+)} + ||h||_{H^{4/3}(0,T)})$ and choose $T^* \in (0,T]$ such that

$$C((T^*)^{1/4} + (T^*)^{1/2})r \le \frac{1}{2}.$$

By the linear estimates given in Proposition 8, it is clear that

$$||u_j||_{X_{T^*,3}} \le r \text{ and } ||u_{j+1}-u_j||_{X_{T^*,3}} \le \tfrac{1}{2}||u_j - u_{j-1}||_{X_{T^*,3}} \text{ for all } j \in \mathbb{N}\setminus\{0\}.$$

Thus $(*)$ is a contraction. Its unique fixed point is the solution of the IBVP (1) defined on the temporal interval $[0,T^*]$. □

Lemma 11. *Let $T > 0$ be given. Assume that T is a computable real number. Then the function $V : \mathcal{S}(\mathbb{R})|_{\mathbb{R}^+} \times \mathcal{S}(\mathbb{R})|_{[0,T]} \times \mathbb{N} \to C([0,T]; \mathcal{S}(\mathbb{R})|_{\mathbb{R}^+})$,*

$$\begin{cases} V(\varphi, h, 0) = L_I(\varphi) + L_B(h) \\ V(\varphi, h, j+1) = V(\varphi, h, 0) - N(V(\varphi, h, j), V(\varphi, h, j)),\ j \geq 0 \end{cases}$$

is $(\delta_{\mathbb{R}^+}, \delta_{[0,T]}, \nu_{\mathbb{N}}, [\rho \to \delta_{\mathbb{R}^+}])$-computable, where $\nu_{\mathbb{N}} : \Sigma^* \to \mathbb{N}$ is a standard notation of $\mathbb{N}$.

Proof. Combine Lemma 6 and the fact that V is defined by primitive recursion. □

The rest of this section is devoted to the proof of Theorem 7. Let $T > 0$ be given. Assume that T is a computable real number. Let p be a $\delta_{H^3(\mathbb{R}^+)}$-name of the initial data $\varphi \in H^3(\mathbb{R}^+)$ and q a $\delta_{H^{4/3}(0,T)}$-name of the boundary data $h \in H^{4/3}(0,T)$. Then p encodes a sequence $\{\varphi_k\}$ of restrictions to $\mathbb{R}^+$ of Schwartz functions such that $||\varphi_k - \varphi||_{H^3(\mathbb{R}^+)} \leq 2^{-k}$ and q encodes a sequence $\{h_k\}$ of restrictions to $[0,T]$ of Schwartz functions such that $||h_k - h||_{H^{4/3}(0,T)} \leq 2^{-k}$.

Solving the problem (1) computationally will be shown that, for any $t \in (0,T]$, a sequence of Schwartz functions can be computed from p, q, a ρ-name of T, and a ρ-name of t such that it is effectively convergent to $u(t)$ in $H^3(\mathbb{R}^+)$-norm. Since

$$||\varphi_k||_{H^3(\mathbb{R}^+)} \leq ||\varphi_k - \varphi||_{H^3(\mathbb{R}^+)} + ||\varphi - \varphi_0||_{H^3(\mathbb{R}^+)} + ||\varphi_0||_{H^3(\mathbb{R}^+)} \leq ||\varphi_0||_{H^3(\mathbb{R}^+)} + 2$$

and

$$||\varphi||_{H^3(\mathbb{R}^+)} \leq ||\varphi - \varphi_0||_{H^3(\mathbb{R}^+)} + ||\varphi_0||_{H^3(\mathbb{R}^+)} \leq ||\varphi_0||_{H^3(\mathbb{R}^+)} + 1,$$

if set $r_1 = ||\varphi_0||_{H^3(\mathbb{R}^+)} + 2$, then $||\varphi||_{H^3(\mathbb{R}^+)} \leq r_1$ and $||\varphi_k||_{H^3(\mathbb{R}^+)} \leq r_1$ for all $k \in \mathbb{N}$. A similar argument shows that for $r_2 = ||h_0||_{H^{4/3}(0,T)} + 2$, $||h||_{H^{4/3}(0,T)} \leq r_2$ and $||h_k||_{H^{4/3}(0,T)} \leq r_2$ for all $k \in \mathbb{N}$. r_1 and r_2 are computable form p and q, respectively.

Let $r = 2C(r_1 + r_2)$ and choose $T^* \in (0,T]$ such that $C((T^*)^{1/2} + (T^*)^{1/4})r \leq 1/2$. Then a similar argument as of the proof of Proposition 10 implies that $||u||_{X_{T^*,3}} \leq r$, where u is the solution of the original IBVP (1); in addition, the following iterative sequence

$$v_0^k = L_I(\varphi_k) + L_B(h_k), \quad v_{j+1}^k = v_0^k - N(v_j^k, v_j^k), \quad j \in \mathbb{N} \tag{7}$$

satisfies the inequalities

$$||v_j^k||_{X_{T^*,3}} \leq r \text{ and } ||v_{j+1}^k - v_j^k||_{X_{T^*,3}} \leq (\frac{1}{2})^j ||v_1^k - v_0^k||_{X_{T^*,3}} \tag{8}$$

for all $j, k \in \mathbb{N}$. Let v^k be the unique fixed point of the iteration (7), then v^k satisfies the equation $v^k = L_I(\varphi_k) + L_B(h_k) - N(v^k, v^k)$ and $||v^k||_{X_{T^*,3}} \leq r$. Recall that, by (6), u is the solution of the IBVP (1) if and only if it satisfies

the integral equation $u = L_I(\varphi) + L_B(h) - N(u,u)$. A simple calculation reveals that the difference $u - v^k$ may be written as

$$u - v^k = L_I(\varphi - \varphi_k) + L_B(h - h_k) - N(u + v^k, u - v^k).$$

Let $\tilde{T} \in (0, T^*]$ be a number to be determined. Since

$$\begin{aligned} & ||u - v^k||_{X_{\tilde{T},3}} \\ \leq\ & C(||\varphi - \varphi_k||_{H^3(\mathbb{R}^+)} + ||h - h_k||_{H^{4/3}(0,T)} + \\ & +(\tilde{T}^{1/2} + \tilde{T}^{1/4})||u + v^k||_{X_{\tilde{T},3}}||u - v^k||_{X_{\tilde{T},3}}) \\ \leq\ & C(||\varphi - \varphi_k||_{H^3(\mathbb{R}^+)} + ||h - h_k||_{H^{4/3}(0,T)} + \\ & +(\tilde{T}^{1/2} + \tilde{T}^{1/4}) \cdot 2r \cdot ||u - v^k||_{X_{\tilde{T},3}}), \end{aligned} \tag{9}$$

if choose $\tilde{T}$ such that $C(\tilde{T}^{1/2} + \tilde{T}^{1/4}) \cdot 2r < 1/2$, then (9) implies

$$||u - v^k||_{X_{\tilde{T},3}} \leq 2C(||\varphi - \varphi_k||_{H^3(\mathbb{R}^+)} + ||h - h_k||_{H^{4/3}(0,T)}). \tag{10}$$

The inequality (10) establishes the fact that v^k effectively converges to u in $||\cdot||_{X_{\tilde{T},3}}$-norm as $k \to \infty$ for $||\varphi - \varphi_k||_{H^3(\mathbb{R}^+)} \leq 2^{-k}$ and $||h - h_k||_{H^{4/3}(0,T)} \leq 2^{-k}$ for all $k \in \mathbb{N}$. Since $\tilde{T} \leq T^*$, v_j^k converges to v^k in $||\cdot||_{X_{\tilde{T},3}}$-norm effectively in k and uniformly in j by (8). Combine Lemma 11, estimates (8) and (10), appropriate numbers k_n and j_n can be computed from p (a $\delta_{H^3(\mathbb{R}^+)}$-code of φ), q (a $\delta_{H^{4/3}(0,T)}$-code of h) and any ρ-code of T such that $||u - v_{j_n}^{k_n}||_{X_{\tilde{T},3}} \leq 2^{-n}$ for all n. Since $||u(t) - v_{j_n}^{k_n}(t)||_{H^3(\mathbb{R}^+)} \leq ||u - v_{j_n}^{k_n}||_{X_{\tilde{T},3}}$ for any $t \in (0, \tilde{T}]$, the sequence $\{v_{j_n}^{k_n}(t)\}$ of Schwartz functions is effectively convergent to $u(t)$ in $H^3(\mathbb{R}^+)$-norm for any $t \in (0, \tilde{T}]$. This is the desired sequence for t, $t \in (0, \tilde{T}]$.

It remains to show that the construction for the desired sequences can be extended from $[0, \tilde{T}]$ to $[0, T]$. The above construction shows that the map $\mathbb{R}^+ \to \mathbb{R}^+$, $||\varphi||_{H^3(\mathbb{R}^+)} + ||h||_{H^{4/3}(0,T)} \mapsto \tilde{T}$ is computable and non-increasing. Thus for any $\tilde{r} > 0$, if $||\varphi||_{H^3(\mathbb{R}^+)} + ||h||_{H^{4/3}(0,T)} \leq \tilde{r}$, then the solution of the problem (1) exists on the temporal interval $[0, \tilde{T}(\tilde{r})]$. Make use of a priori estimate in [2], a constant $R > 0$ can be computed from φ, h and T such that

$$\sup_{0 \leq t \leq T} ||u(\cdot, t)||_{H^3(\mathbb{R}^+)} + ||h||_{H^{4/3}(0,T)} \leq R.$$

Since $u(x,0) = \varphi(x)$, $||\varphi||_{H^3(\mathbb{R}^+)} + ||h||_{H^{4/3}(0,T)} \leq R$. Choose $\tilde{T} = \tilde{T}(R)$. Then the solution exists on the temporal interval $[0, \tilde{T}]$. In addition, when taking $\tilde{T}$ as the initial time and $u(x, \tilde{T})$ as the initial data, since $||u(\cdot, \tilde{T})||_{H^3(\mathbb{R}^+)} + ||h||_{H^{4/3}(0,T)} \leq R$, the solution of the problem

$$w_t + w_x + w_{xxx} = -ww_x, \quad x \geq 0,\ t \geq \tilde{T}, \quad w(x, \tilde{T}) = u(x, \tilde{T}),\ w(0,t) = h(t)$$

exists on $[\tilde{T}, 2\tilde{T}]$. This solution is also the solution of the IBVP (1) over the temporal interval $[\tilde{T}, 2\tilde{T}]$. Repeat the process finitely many times until the solution is extended to the interval $[0, T]$. □

References

1. Robert A. Adams: Sobolev spaces. Academic Press, New York, 1975.
2. J. L. Bona, S. Sun and B.-Y. Zhang: A non-homogeneous boundary-value problem for the Korteweg-de Vries equation in a quarter plane. *Trans. American Math. Soc.* **354** (2001), 427 – 490.
3. J. Bourgain: Fourier transform restriction phenomena for certain lattice subsets and applications to non-linear evolution equations, part I: Schrödinger equations. *Geom. & Funct. Anal.* **3** (1993), 107 – 156.
4. J. Bourgain: Fourier transform restriction phenomena for certain lattice subsets and applications to non-linear evolution equations, part II: the KdV equation. *Geom. & Funct. Anal.* **3** (1993), 209 – 262.
5. D.J. Korteweg and G. de Vries: On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *Philosophy Magazine*, 35(5):422–443, 1895.
6. Carlos E. Kenig, Gustavo Ponce, and Luis Vega: Well-posedness of the initial value problem for the Korteweg-de Vries equation. *J. Amer. Math. Soc.*, 4:323–347, 1991.
7. Marian B. Pour-El and J. Ian Richards: Computability in Analysis and Physics. Perspectives in Mathematical Logic. Springer, Berlin, 1989.
8. Klaus Weihrauch: Computable Analysis. Springer, Berlin, 2000.
9. Klaus Weihrauch and Ning Zhong: Computing the solution of the Korteweg-de Vries Equation with arbitrary precision on Turing machines. *Theoretical Computer Science*, to appear.
10. Ning Zhong and Klaus Weihrauch: Computability theory of generalized functions. *Journal of ACM*, 50(4):469–505, 2003.

Computability and Continuity on the Real Arithmetic Hierarchy and the Power of Type-2 Nondeterminism

Martin Ziegler*

Institut for Matematik og Datalogi,
Syddansk Universitet, 5230 Odense M, Denmark
ziegler@imada.sdu.dk

Abstract. The sometimes so-called Main Theorem of Recursive Analysis implies that any computable real function is necessarily continuous. We consider three relaxations of this common notion of real computability for the purpose of treating also discontinuous functions $f : \mathbb{R} \to \mathbb{R}$:
- non-deterministic computation;
- relativized computation, specifically given access to oracles like $\emptyset'$ or $\emptyset''$;
- encoding input $x \in \mathbb{R}$ and/or output $y = f(x)$ in weaker ways according to the Real Arithmetic Hierarchy.

It turns out that, among these approaches, only the first one provides the required power.

1 Motivation

What does it mean for a Turing Machine, capable of operating only on discrete objects, to compute a real number x:

$\rho_{\mathbf{b,2}}$: To decide its binary expansion?

$\rho_{\mathbf{Cn}}$: To compute a sequence (q_n) of rational numbers eventually converging to x?

ρ: To compute a *fast* convergent sequence $(q_n) \subseteq \mathbb{Q}$ for x, i.e. with $|x - q_n| \leq 2^{-n}$ (in other words: to approximate x with error bounds)?

$\rho_{<}$: To approximate x from below, i.e., to compute (q_n) such that $x = \sup_n q_n$?

All these notions make sense in being closed under arithmetic operations like addition and multiplication. In fact they are well (known equivalent to variants) studied in literature[1]; e.g. [11], [2], [12], [13] in order.

Now what does it mean for a Turing Machine M to compute a real function $f : \mathbb{R} \to \mathbb{R}$? Most naturally it says that, upon input of $x \in \mathbb{R}$ given in one of

* Supported by project 21-04-0303 of Statens Naturvidenskabelige Forskningsråd SNF.
[1] Their above names by Greek letters are taken from [13, Section 4.1].

S.B. Cooper, B. Löwe, and L. Torenvliet (Eds.): CiE 2005, LNCS 3526, pp. 562–571, 2005.

the above ways, M outputs $y = f(x)$ also in one (not necessarily the same) of the above ways. And, again, many possible combinations have already been investigated. For instance the standard notion of real function computation in Recursive Analysis [3, 6, 5, 13] refers (or is equivalent) to input and output given according to ρ. Here, the Main Theorem of Computable Analysis implies that any computable f will necessarily be continuous [13, Theorem 4.3.1].

We are interested in ways of lifting this restriction.

A first approach might equip the Turing Machines under consideration with access to an oracle, say, for the Halting Problem $\emptyset'$ or its iterated jumps $\emptyset^{(d)}$ in Kleene's Arithmetic Hierarchy. However closer inspection in Section 3.1 reveals that this Main Theorem relies solely on information rather than recursion theoretic arguments and therefore requires continuity also for oracle-computable real functions with respect to input and output of form ρ. (For the special case of an $\emptyset'$–oracle, this had been observed in [4, Theorem 16].)

A second idea changes the input and output representation for x and $y = f(x)$ from ρ to a weaker form like, say, ρ_{Cn}. This relates to the Arithmetic Hierarchy, too, however in a completely different way: Computing x in the sense of ρ_{Cn} is equivalent to computing x in the sense of ρ [4, Theorem 9] *relative* (i.e., given access) to the Halting Problem $\emptyset'$. And, most promisingly, the Main Theorem [13, Corollary 3.2.12] which previously required continuity of computable real functions now does not apply any more since ρ_{Cn}, in contrast to ρ, lacks the technical property of *admissibility*. It therefore came to quite a surprise when Brattka and Hertling established that any $(\rho_{\mathrm{Cn}} \to \rho_{\mathrm{Cn}})$–computable f (that is, with respect to input x and output $f(x)$ encoded according to ρ_{Cn}) still satisfies continuity; see [13, Exercise 4.1.13d] or [2, Section 6].

In Section 3.2, we extend this result. Specifically it is proven that continuity is necessary for $(\rho'' \to \rho'')$–computability of f; here, $\rho \precneqq \rho' \equiv \rho_{\mathrm{Cn}} \precneqq \rho'' \precneqq \ldots$ denote the first levels of an entire hierarchy of real number representations emerging naturally from the Real Arithmetic Hierarchy of Weihrauch and Zheng [14].

The class of $(\rho \to \rho)$–computable functions $f : \mathbb{R} \to \mathbb{R}$ is well-known to admit an alternative characterization based on Caldwell's and Pour-El's famous Effective Weierstraß Theorem. Its extension based on relativization gives rise to a hierarchy of continuous functions $f : \mathbb{R} \to \mathbb{R}$. The $(\rho \to \rho)'$–computable ones for example have been investigated in [4, Section 4]. Section 4 of the present work locates the class of $(\rho' \to \rho')$–computable functions among this hierarchy.

Rather than endowing deterministic Turing Machines with oracles, in Section 5 we finally consider *non*deterministic computation. Remarkably and in contrast to the classical (Type-1) theory, this significantly increases the principal capabilities. For example, discontinuous real functions now do become computable and so does conversion among the aforementioned representations ρ_{Cn} and $\rho_{\mathrm{b},2}$.

2 Arithmetic Hierarchy of Reals

In [4], Ho observed an interesting relation between computability of a real number x in the respective senses of ρ and ρ_{Cn} in terms of oracles: $x = \lim_n q_n$ for an (eventually convergent) computable rational sequence (q_n) iff x admits a $\emptyset'$–computable *fast* convergent rational sequence, that is, a sequence $(p_m) \subseteq \mathbb{Q}$ recursive in $\emptyset'$ with $|x - p_m| \leq 2^{-m}$. This suggests to write ρ' for ρ_{Cn}; and denoting by $\Delta_1\mathbb{R} = \mathbb{R}_c$ the set of reals computable in the sense of Recursive Analysis (that is with respect to ρ), it is therefore natural to write, in analogy to Kleenes classical Arithmetic Hierarchy, $\Delta_2\mathbb{R}$ for the set of all $x \in \mathbb{R}$ computable with respect to ρ'. Weihrauch and Zheng extended these considerations and obtained for instance [14, Corollary 7.3] the following characterization of $\Delta_3\mathbb{R}$: A real $x \in \mathbb{R}$ admits a fast convergent rational sequence recursive in $\emptyset''$ iff x is computable in the sense of ρ'' defined as follows:

$\boldsymbol{\rho''}$**:** $x = \lim_i \lim_j q_{\langle i,j\rangle}$ for some computable rational sequence (q_n).

Similarly, $\Sigma_1\mathbb{R}$ contains of all $x \in \mathbb{R}$ computable with respect to $\rho_<$ whereas $\Sigma_2\mathbb{R}$ includes all x computable in the sense of $\rho'_<$ defined as follows:

$\boldsymbol{\rho'_<}$**:** $x = \sup_i \inf_j q_{\langle i,j\rangle}$ for some computable rational sequence (q_n).

To $\Sigma_2\mathbb{R}$ belongs for instance the radius of convergence $r = 1/\limsup_{n\to\infty} \sqrt[n]{a_n}$ of a computable power series $\sum_{n=0}^{\infty} a_n x^n$ [14, Theorem 6.2]. More generally we take from [14, Definition 7.1 and Corollary 7.3] the following

Definition 1 (Real Arithmetic Hierarchy). *Let* $d = 0, 1, 2, \ldots$

$\boldsymbol{\rho^{(d)}_<}$*:* $\Sigma_{d+1}\mathbb{R}$ *consists of all* $x \in \mathbb{R}$ *of the form* $x = \sup_{n_1} \inf_{n_2} \ldots \Theta_{n_{d+1}} q_{\langle n_1,\ldots,n_{d+1}\rangle}$ *for a computable rational sequence* (q_n)*, where* $\Theta{=}\sup$ *or* $\Theta{=}\inf$ *depending on* d*'s parity;*

$\boldsymbol{\rho^{(d)}_>}$*:* $\Pi_{d+1}\mathbb{R}$ *similarly for* $x = \inf_{n_1} \sup_{n_2} \ldots$

$\boldsymbol{\rho^{(d)}}$*:* $\Delta_{d+1}\mathbb{R}$ *contains all* $x \in \mathbb{R}$ *of the form* $x = \lim_{n_1} \lim_{n_2} \ldots \lim_{n_d} q_{\langle n_1,\ldots,n_d\rangle}$ *for a computable rational sequence* (q_n).

(For levels beyond ω see [1]...)

The close analogy between the discrete and this real variant of the Arithmetic Hierarchy is expressed in [14] by a variety of elegant results like, e.g.,

Fact 2.

a) $x \in \Delta_d\mathbb{R}$ *iff deciding its binary expansion is in* Δ_d.
b) x *is computable with respect to* $\rho^{(d)}$ *iff there is a* $\emptyset^{(d)}$*–computable fast convergent rational sequence for* x.
c) x *is computable with respect to* $\rho^{(d)}_<$ *iff* x *is the supremum of a* $\emptyset^{(d)}$*–computable rational sequence.*
d) $\Delta_d\mathbb{R} = \Sigma_d\mathbb{R} \cap \Pi_d\mathbb{R}$.
e) $\Sigma_d\mathbb{R} \cup \Pi_d\mathbb{R} \subsetneq \Delta_{d+1}\mathbb{R}$.

2.1 Type-2 Theory of Effectivity

Specifying an encoding formalizes how to feed some general form of input like graphs or integers into a Turing Machine with fixed alphabet Σ. Already in the

discrete case, the complexity of a problem usually depends heavily on the chosen encoding; e.g., numbers in unary versus binary. This dependence becomes even more important when dealing with objects from an continuum like the set of reals and their computability. While Recursive Analysis usually considers one particular encoding for $\mathbb{R}$, the Type-2 Theory of Effectivity (TTE) due to Weihrauch provides (a convenient formal framework for studying and comparing) a variety of encodings for different universes. Formally speaking, a *representation* α for $\mathbb{R}$ is a partial surjective mapping $\alpha :\subseteq \Sigma^\omega \to \mathbb{R}$; and an infinite string $\bar\sigma \in \mathrm{dom}(\alpha)$ is regarded as a name for the real number $x = \alpha(\bar\sigma)$. In this way, $(\alpha \to \beta)$–computing a real function $f : \mathbb{R} \to \mathbb{R}$ means to compute a transformation on infinite strings $F :\subseteq \Sigma^\omega \to \Sigma^\omega$ such that any α–name $\bar\sigma$ for $x = \alpha(\bar\sigma)$ gets transformed to a β–name $\bar\tau = F(\bar\sigma)$ for $f(x) = y$, that is, satisfying $\beta(\bar\tau) = y$; cf. [13, Section 3].

Now observe that (the above characterization of) each level of the Real Arithmetic Hierarchy gives rise not only to a notion of computability for real numbers but also canonically to a representation for $\mathbb{R}$; for instance let

$\boldsymbol{\rho}$: encode (arbitrary!) $x \in \mathbb{R}$ as a fast convergent rational sequence (q_n);
$\boldsymbol{\rho_<}$: encode $x \in \mathbb{R}$ as supremum of a rational sequence: $x = \sup_n q_n$;
$\boldsymbol{\rho'}$: encode $x \in \mathbb{R}$ as limit of a rational sequence: $x = \lim_n q_n$;
$\boldsymbol{\rho'_<}$: encode $x \in \mathbb{R}$ as $(q_n) \subseteq \mathbb{Q}$ with $x = \sup_i \inf_j q_{\langle i,j\rangle}$;
$\boldsymbol{\rho''}$: encode $x \in \mathbb{R}$ as $(q_n) \subseteq \mathbb{Q}$ with $x = \lim_i \lim_j q_{\langle i,j\rangle}$.

In fact the first three of them are known in TTE as ρ, $\rho_<$, and ρ_{Cn}, respectively [13, Section 4.1]. In general one obtains, similar to Definition 1, a hierarchy of real representations as follows:

Definition 3. *Let* $\rho^{(0)} := \rho$, $\rho_<^{(0)} := \rho_<$, $\rho_>^{(0)} := \rho_>$. *Now fix* $1 \le d \in \mathbb{N}$*:*
A $\rho^{(d)}$*–name for* $x \in \mathbb{R}$ *is a (*$\nu_{\mathbb{Q}}^\omega$*–name for a) rational sequence* (q_n) *such that*

$$x \;=\; \lim_{n_1}\lim_{n_2}\ldots\lim_{n_d} q_{\langle n_1,\ldots,n_d\rangle} \;.$$

A $\rho_<^{(d)}$*–name for* $x \in \mathbb{R}$ *is a (name for a) sequence* $(q_n) \subseteq \mathbb{Q}$ *such that*

$$x \;=\; \sup_{n_1}\inf_{n_2}\ldots\Theta_{n_{d+1}} q_{\langle n_1,\ldots,n_{d+1}\rangle} \;.$$

A $\rho_>^{(d)}$*–name for* $x \in \mathbb{R}$ *is a sequence* $(q_n) \subseteq \mathbb{Q}$ *such that* $x = \inf_{n_1}\sup_{n_2}\ldots$

Results from [14] about the Real Arithmetic Hierarchy are easily re-phrased in terms of these representations. However this leads only to non-uniform claims. Fact 2d) for example translates as follows:

$$x \text{ is } \rho^{(d)}\text{–computable} \;\;\text{iff}\;\; \text{it is both } \rho_<^{(d)}\text{–computable and } \rho_>^{(d)}\text{–computable.}$$

Closer inspection of the proofs in particular of Lemma 3.2 and Lemma 3.3 in [14] reveals them to hold even uniformly. More precisely we can prove

Lemma 4. $\rho \equiv \rho_< \sqcap \rho_> \not\preceq \rho_< \not\preceq \rho' \equiv \rho'_< \sqcap \rho'_> \not\preceq \rho'_< \not\preceq \rho'' \equiv \ldots$, *where "*$\preceq$*" and "*$\equiv$*" denote computable reducibility and equivalence, respectively, of representations and "*$\sqcap$*" their* join*, i.e., the least upper bound w.r.t. "*$\preceq$*"; see* [13, Definition 2.3.2].

3 Computability and Continuity

Recursive Analysis has established as folklore that any computable real function is continuous. More precisely, computability of a partial function from/to infinite strings $f :\subseteq \Sigma^\omega \to \Sigma^\omega$ requires continuity with respect to the Cantor Topology τ_C [13, Theorem 2.2.3]; and this requirement carries over to functions $f :\subseteq A \to B$ on other topological spaces (A, τ_A) and (B, τ_B) where input $a \in A$ and output $b = f(a)$ are encoded by respective *admissible* representations α and β. Roughly speaking, this property expresses that the mappings $\alpha :\subseteq \Sigma^\omega \to A$ and $\beta :\subseteq \Sigma^\omega \to B$ satisfy a certain compatibility condition with respect to the topologies τ_A/τ_B and τ_C involved. For $A = B = \mathbb{R}$, the (standard) representation ρ for example is admissible [13, Lemma 4.1.4.1], thus recovering the folklore claim.

Now in order to treat and non-trivially investigate computability of discontinuous real functions $f : \mathbb{R} \to \mathbb{R}$ as well, there are basically two ways out: Either enhance the underlying Type-2 Machine model or resort to non-admissible representations. It turns out that for either choice, at least the straight-forward approaches fail:

- extending Turing Machines with oracles as well as
- considering weakened representations for $\mathbb{R}$.

3.1 Type-2 Oracle Computation

Specifically concerning the first approach we make the following

Observation 5. *Let $\mathcal{O} \subseteq \Sigma^*$ be arbitrary. Replace in* [13, Definition 2.1.1] *the Turing Machine M by $M^{\mathcal{O}}$, that is, one with oracle access to $\mathcal{O}$. Then* Lemma 2.1.11, Theorem 2.1.12, Theorem 2.2.3, Corollary 3.2.12, *and* Theorem 4.3.1 *in* [13] *still remain valid.*

In particular, the Main Theorem of Computable Analysis relativizes.

Corollary 6. *A partial function on infinite strings $f :\subseteq \Sigma^\omega \to \Sigma^\omega$ is Cantor-continuous iff it is computable relative to some oracle $\mathcal{O}$.*

This is to be compared with Type-1 Theory (that is, computability on finite strings) where any function $f :\subseteq \Sigma^* \to \Sigma^*$ becomes recursive in some appropriate $\mathcal{O} \subseteq \Sigma^*$.

Proof. If f is recursive in $\mathcal{O}$, then it is also continuous by the relativized version of [13, Theorem 2.2.3]. Conversely let f be continuous; then a monotone total function $h : \Sigma^* \to \Sigma^*$ with $h_\omega = f$ according to [13, Definition 2.1.10.2] can be seen to exist. But being a classical Type-1 function, h is recursive in a certain oracle $\mathcal{O} \subseteq \Sigma^*$. The relativization of [13, Lemma 2.1.11.2] then asserts also f to be computable in $\mathcal{O}$. □

While oracles thus do not increase the computational power of a Type-2 Machine sufficiently in order to handle also discontinuous functions, Section 5 reveals that nondeterminism does.

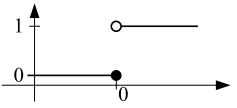
3.2 Weaker Representations for Reals

Concerning the second approach, we are interested in relaxations of the standard representation ρ for single reals and their effect on the computability of function evaluation $x \mapsto f(x)$. Since, with exception of ρ, none of the ones introduced in Definition 3 is admissible [13, Lemma 4.1.4, Example 4.1.14.1], chances are good for this problem to become computable even for discontinuous $f : \mathbb{R} \to \mathbb{R}$.

Example 7. Heaviside's Function

$H : \mathbb{R} \to \mathbb{R}, \qquad x \mapsto 0 \;\text{ for }\; x \le 0, \quad x \mapsto 1 \;\text{ for }\; x > 0$

is both $(\rho_< \to \rho_<)$–computable and $(\rho'_< \to \rho'_<)$–computable.



Proof. Given $(q_n) \subseteq \mathbb{Q}$ with $x = \sup_n q_n$, compute $p_n := H(q_n)$. Then indeed, $(p_n) \subseteq \mathbb{Q}$ has $\sup_n p_n = H(x)$: In case $x \le 0$, $q_n \le 0$ and hence $p_n = 0$ for all n; whereas in case $x > 0$, $q_n > 0$ and hence $p_n = 1$ for some n.

Let $x \in \mathbb{R}$ be given by a rational double sequence $(q_{i,j})$ with $x = \sup_i \inf_j q_{i,j}$. Proceeding from $q_{i,j}$ to $\tilde{q}_{i,j} := \max\{q_{0,j}, \ldots, q_{i,j}\}$, we assert $\inf_j \tilde{q}_{i+1,j} \ge \inf_j \tilde{q}_{i,j}$. Now compute $p_{i,j} := H(\tilde{q}_{i,j} - 2^{-i})$. Then in case $x \le 0$, it holds $\forall i \exists j : \tilde{q}_{i,j} \le 2^{-i}$, i.e., $p_{i,j} = 0$ and thus $\sup_i \inf_j p_{i,j} = 0 = H(x)$. Similarly in case $x > 0$, there is some i_0 such that $\inf_j \tilde{q}_{i_0,j} > x/2$ and thus $\inf_j \tilde{q}_{i,j} > x/2$ for all $i \ge i_0$. For $i \ge i_0$ with $2^{-i} \le x/2$, it follows $p_{i,j} = 1\ \forall j$ and therefore $\sup_i \inf_j p_{i,j} = 1 = H(x)$. □

It turns out that the implication "$(\rho_< \to \rho_<) \Rightarrow (\rho'_< \to \rho'_<)$" is not specific to H:

Theorem 8. *Consider $f : \mathbb{R} \to \mathbb{R}$.*
a) If f is $(\rho \to \rho)$–computable, then it is also $(\rho' \to \rho')$–computable.
b) If f is $(\rho \to \rho_<)$–computable, then it is also $(\rho' \to \rho'_<)$–computable.
c) If f is $(\rho_< \to \rho_<)$–computable, then it is also $(\rho'_< \to \rho'_<)$–computable.
d) If f is $(\rho' \to \rho')$–computable, then it is also $(\rho'' \to \rho'')$–computable.
Our proof exploits Fact 9 and Theorem 10 below.

By the Main Theorem of Recursive Analysis, any $(\rho \to \rho)$–computable real function is continuous. This claim has been extended in various ways.

Fact 9. *Consider $f : \mathbb{R} \to \mathbb{R}$.*
a) If f is $(\rho \to \rho_<)$–computable, then it is lower semi continuous.
b) If f is $(\rho_< \to \rho_<)$–computable, then it is monotonically increasing.
c) If f is $(\rho' \to \rho')$–computable, then it is continuous.
The claims remain valid under oracle-supported computation.

Proof. For a) see e.g. [7, Chapter 6.7]; c) is established in [2, Section 6].

These claims also generalize to the above hierarchy of representations:

Theorem 10. *Consider $f : \mathbb{R} \to \mathbb{R}$.*
a) If f is $(\rho'' \to \rho'')$–computable, then it is continuous.
b) If f is $(\rho' \to \rho'_<)$–computable, then it is lower semi-continuous.
c) If f is $(\rho'_< \to \rho'_<)$–computable, then it is monotonically increasing.
The claims remain valid under oracle-supported computation.

This allows us to strengthen Lemma 4:

Corollary 11. $\rho \equiv \rho_< \sqcap \rho_> \not\preceq_t \rho_< \not\succeq_t \rho' \equiv \rho'_< \sqcap \rho'_> \not\succeq_t \rho'_< \not\succeq_t \rho'' \equiv \ldots$ *where "*$\preceq_t$*" denotes* continuous *reducibility of representations* [13, Def. 2.3.2].

Proof. The positive claims follow from Lemma 4 with Corollary 6. For a negative claim like for example "$\rho''_< \not\preceq_t \rho''$" suppose the contrary. Then by Corollary 6, with the help of some appropriate oracle $\mathcal{O}$, one can convert $\rho''_<$–names to ρ''–names. As Heaviside's Function H is $(\rho'' \to \rho''_<)$–computable by Example 7 and Theorem 8b), composition with this conversion implies $(\rho'' \to \rho'')$–computability of H relative to $\mathcal{O}$ — contradicting Theorem 10a). □

4 Arithmetic Weierstraß Hierarchy

Recall that Weierstrass Approximation Theorem asserts any continuous real function $f : [0,1] \to \mathbb{R}$ to be the uniform limit $f = \mathrm{ulim}_n P_n$ of a sequence of rational polynomials $(P_n) \subseteq \mathbb{Q}[X]$. Here, 'ulim' suggestively denotes uniform limits of continuous functions, that is, with $\sup_{0\le x\le 1} |f(x) - P_n(x)| =: \|f - P_n\| \to 0$ as $n \to \infty$.

For $(\rho \to \rho)$–computable (and thus continuous) f, the well-known Effective Weierstraß Theorem yields even a *computable* and fast convergent such sequence (P_n). Furthermore by allowing this sequence (P_n) to be computable relative to $\emptyset^{(d)}$, one naturally obtains, in analogy to the Real Arithmetic Hierarchy, a hierarchy of real functions $f : [0,1] \to \mathbb{R}$ with the effective case as lowest level. In fact its second level, too, has already been characterized namely by Ho.

Lemma 12.

a) A real function $f : [0,1] \mapsto \mathbb{R}$ is $(\rho \mapsto \rho)$–computable if and only if it holds

$(\rho \to \rho)$: *There exists a computable sequence of (degrees and coefficients of) rational polynomials $(P_n) \subseteq \mathbb{Q}[X]$ such that* $\|f - P_n\| \le 2^{-n}$ (1)

b) To a real function $f : [0,1] \to \mathbb{R}$, there exists a $\emptyset'$–computable sequence of polynomials (P_n) satisfying Equation (1) if and only if it holds

$(\rho \to \rho)'$: *There is a computable sequence $(Q_m) \subseteq \mathbb{Q}[X]$ converging uniformly (although not necessarily 'fast') to f, that is, with $f = \mathrm{ulim}_{m\to\infty} Q_m$.*

c) To a real function $f : [0,1] \to \mathbb{R}$, there exists a $\emptyset''$–computable sequence of polynomials (P_n) satisfying Equation (1) if and only if it holds

$(\rho \to \rho)''$: *There is a computable sequence $(Q_m) \subseteq \mathbb{Q}[X]$ s.t. $f = \mathrm{ulim}_i \mathrm{ulim}_j Q_{\langle i,j\rangle}$.*

We emphasize the similarity to Fact 2b).

Proof. a) See, e.g., [6, Section 0.7]. b) See [4, Theorem 16]. c) follows similarly from Shoenfield's Limit Theorem.

Recall that the $(\rho' \to \rho')$–computable functions are continuous by [2, Section 6] and thus applicable to the Weierstraß Theorem. The following result relates this class to relativized computation of Weierstraß approximations.

Theorem 13. *a) Let $f : [0,1] \to \mathbb{R}$ be $(\rho \to \rho)'$–computable in the sense of Lemma 12b). Then, f is $(\rho' \to \rho')$–computable.*
b) Let $f : [0,1] \to \mathbb{R}$ be $(\rho' \to \rho')$–computable. Then, f is $(\rho \to \rho)''$–computable.
c) There is a $(\rho' \to \rho')$–computable but not $(\rho \to \rho)'$–computable $f : [0,1] \to \mathbb{R}$.

As opposed to $d = 0$, the hierarchies of $(\rho^{(d)} \to \rho^{(d)})$–computable functions on the one hand and of $(\rho \to \rho)^{(d)}$–computable functions on the other hand thus lie skewly to each other for $d \geq 1$.

5 Type-2 Nondeterminism

Observing that the proofs of the Main Theorem in Recursive Analysis as well as its generalizations to oracle- and weakened real computation in Theorem 10 crucially exploit the underlying Turing Machines to be deterministic, the question becomes evident whether nondeterminism might yield the additional power necessary for computing discontinuous real functions like Heaviside's.

In the discrete (i.e., Type-1) setting where any computation is required to terminate, the finitely many possible choices of a nondeterministic machine can of course be simulated by a deterministic one — however already here subject to the important condition that all paths of the nondeterministic computation indeed terminate, cf. [9].

In contrast, a Type-2 computation realizes a transformation from/to infinite strings and is therefore a generally non-terminating process. Therefore, nondeterminism here involves an infinite number of guesses which turns out cannot be simulated by a deterministic Type-2 machine. We also point out that nondeterminism has already before been revealed not only a useful but indeed the most natural concept of computation on infinite strings. More precisely Büchi extended Finite Automata from finite to infinite strings and proved that, here, nondeterministic ones are closed under complement [10] as opposed to deterministic ones. Since automata and Turing Machines constitute the bottom and top levels, respectively, of Chomsky's Hierarchy of classical languages (Type-1 setting), we suggest that over infinite strings (Type-2 setting) both their respective counterparts, that is Büchi Automata as well as Type-2 Machines, be considered nondeterministically.

Definition 14. *Let A and B be uncountable sets with respective representations $\alpha :\subseteq \Sigma^\omega \to A$ and $\beta :\subseteq \Sigma^\omega \to B$. A function $f :\subseteq A \to B$ is called* nondeterministically $(\alpha \to \beta)$*–computable if some nondeterministic one-way Turing Machine,*
- *upon input of any α–name for some $a \in \mathrm{dom}(f)$,*
- *has a computation which outputs a β–name for $b = f(a)$* *and*
- *any infinite string output by some computation is a β–name for $b = f(a)$.*

This definition is sensible insofar as it leads to closure under composition:

Lemma 15. *Let $f :\subseteq A \to B$ be nondeterministically $(\alpha \to \beta)$–computable and $g :\subseteq B \to C$ be nondeterministically $(\beta \to \gamma)$–computable. Then, $g \circ f :\subseteq A \to C$ is nondeterministically $(\alpha \to \gamma)$–computable.*

A subtle point in Definition 14, computations leading only to finite output are semantically ignored. This enables algorithms to 'withdraw' a nondeterministic choice once its unfavourability is detected. In particular, one can nondeterministically convert forth and back among representations on the Real Arithmetic Hierarchy from Definition 1:

Theorem 16. *For each $d = 0, 1, 2, \ldots$, the identity $\mathbb{R} \ni x \mapsto x$ is nondeterministically $(\rho^{(d+1)} \to \rho^{(d)})$–computable. It is furthermore $(\rho \to \rho_{b,2})$–computable.*

In particular and in striking contrast to usual Type-2 Theory, nondeterministic computability of real functions is largely independent of the representation under consideration. This notion includes discontinuous functions as, by Example 7, the Heaviside Function is nondeterministically computable in any reasonable sense.

Proof (Theorem 16). Consider for simplicity the case $d = 0$. Let $x \in \mathbb{R}$ be given by a sequence $(q_n) \subseteq \mathbb{Q}$ eventually converging to x. There exists a strictly increasing sequence $(n_k) \subseteq \mathbb{N}$ such that (q_{n_k}) converges fast to x, that is, satisfying

$$\forall \ell \leq k : \quad q_{n_k} \in [q_{n_\ell} - 2^{-\ell}, q_{n_\ell} + 2^{-\ell}] \tag{2}$$

For each $k \in \mathbb{N}$, the algorithm iteratively guesses $n_k > n_{k-1}$, prints q_{n_k}, and checks Equation (2): If violated, abort having output only a finite string.

For $(\rho \to \rho_{\mathrm{b},2})$–computability, let $x \in (0, 2)$ be given by a fast convergent sequence $(q_n) \subseteq \mathbb{Q}$. We guess the leading digit $b \in \{0, 1\}$ for x's binary expansion $b.*$; in case $b = 0$, check whether $x > 1$ — a ρ–semi decidable property — and if so, abort; similarly in case $b = 1$, abort if it turns out that $x < 1$. Otherwise (that is, proceeding while simultaneously continuing the above semi-decision process via dove-tailing) replace x by $2(x - b)$ and repeat guessing the next bit. □

We point out that for non-unique binary expansion (i.e., for dyadic x), nondeterminism in the above $(\rho \to \rho_{\mathrm{b},2})$–computation, in accordance with the third requirement of Definition 14, generates both possible expansions.

6 Conclusion

Recursive Analysis is often criticized for being unable to (non-trivially) treat discontinuous functions.

Strictly speaking, this reproach does not apply since for instance Zhong and Weihrauch do investigate the computability of generalized (and in particular of discontinuous) functions by suitable representations in [16]. However here, evaluating $x \mapsto f(x)$ an L^2 function or a distribution f at a point $x \in \mathbb{R}$ does not make sense already mathematically and therefore is not supported.

Another reply to the aforementioned critics, Heaviside's function — although discontinuous — *is* $(\rho \to \rho_<)$–computable; and this notion of function computability does allow for (lower) evaluation. On the other hand, it lacks closure under composition. Closure under composition holds for $(\rho_< \to \rho_<)$–computability, but here closure under inversion $f \mapsto -f$ fails.

The present work investigates on sufficient and necessary conditions for some $f : \mathbb{R} \to \mathbb{R}$ to be $(\rho^{(d)} \to \rho^{(d)})$–computable. Being closed both under composition and inversion, respectively, these notions emerging from the Real Arithmetic Hierarchy are not applicable to the Main Theorem for $d = 1, 2, \ldots$ and might therefore include discontinuous functions.

We extend the surprising result [2, Section 6] that $(\rho' \to \rho')$–computability does imply continuity to the case $d \geq 2$.

Due to the purely information-theoretic nature of the arguments employed by Brattka and Hertling, their result immediately relativizes, that is, even oracle support does not lift the continuity condition. So we looked for other ways of enhancing the underlying model. Replacing, in analogy to Büchi Automata, deterministic Turing machines by nondeterministic ones turned out to do the job. While their practical realizability is admittedly questionable, the obtained notion of function computability benefits from its elegance and invariance under encodings of real numbers.

References

1. G. Barmpalias: "A Transfinite Hierarchy of Reals", pp.163–172 in *Mathematical Logic Quarterly* vol.**49(2)** (2003).
2. V. Brattka, P. Hertling: "Topological Properties of Real Number Representations", pp.241–257 in *Theoretical Computer Science* vol.**284** (2002).
3. A. Grzegorczyk: "On the Definitions of Computable Real Continuous Functions", pp.61–77 in *Fundamenta Mathematicae* **44** (1957).
4. C.-K. Ho: "Relatively Recursive Real Numbers and Real Functions", pp.99–120 in *Theoretical Computer Science* vol.**210** (1999).
5. Ker-I Ko: "*Complexity Theory of Real Functions*", Birkhäuser (1991).
6. M.B. Pour-El, J.I. Richards: "*Computability in Analysis and Physics*", Springer (1989).
7. J.F. Randolph: "*Basic Real and Abstract Analysis*", Academic Press (1968).
8. R.I. Soare: "*Recursively Enumerable Sets and Degrees*", Springer (1987).
9. E. Spaan, L. Torenvliet, P. van Emde Boas: "Nondeterminism, Fairness and a Fundamental Analogy", pp.186–193 in *The Bulletin of the European Association for Theoretical Computer Science* (EATCS Bulletin) vol.**37** (1989).
10. Thomas, W.: "Automata on Infinite Objects", pp.133–191 in *Handbook of Theoretical Computer Science*, vol.**B** (Formal Models and Semantics), Elsevier (1990).
11. Turing, A.M.: "On Computable Numbers, with an Application to the Entscheidungsproblem", pp.230–265 in *Proc. London Math. Soc.* vol.**42(2)** (1936).
12. Turing, A.M.: "On Computable Numbers, with an Application to the Entscheidungsproblem. A correction", pp.544–546 in *Proc. London Math. Soc.* vol.**43(2)** (1937).
13. K. Weihrauch: "*Computable Analysis*", Springer (2001).
14. X. Zheng, K. Weihrauch: "The Arithmetical Hierarchy of Real Numbers", pp.51–65 in *Mathematical Logic Quarterly* vol.**47** (2001).
15. X. Zheng: "Recursive Approximability of Real Numbers", pp.131–156 in *Mathematical Logic Quarterly* vol.**48** Supplement **1** (2002).
16. N. Zhong, K. Weihrauch: "Computability Theory of Generalized Functions", pp.469–505 in *J. ACM* vol.**50** (2003).

Author Index