ISSN 0252–9742

Bulletin

of the

European Association for Theoretical Computer Science

EATCS

Number 89 June 2006

Council of the European Association for Theoretical Computer Science

President:	Mogens Nielsen	Denmark
Vice Presidents:	Jan van Leeuwen	The Netherlands
	Paul Spirakis	Greece
Treasurer:	Dirk Janssens	Belgium
Secretary:	Branislav Rovan	Slovakia
Bulletin Editor:	Vladimiro Sassone	United Kingdom

Luca Aceto	Iceland	David Peleg	Israel
Giorgio Ausiello	Italy	Grzegorz Rozenberg	The Netherlands
Wilfried Brauer	Germany	Arto Salomaa	Finland
Josep Díaz	Spain	Don Sannella	United Kingdom
Zoltán Ésik	Hungary	Jiří Sgall	Czech Republic
Alan Gibbons	United Kingdom	Andrzej Tarlecki	Poland
Giuseppe F. Italiano	Italy	Wolfgang Thomas	Germany
Jean-Pierre Jouannaud	France	Emo Welzl	Switzerland
Juhani Karhumäki	Finland	Ingo Wegener	Germany
Eugenio Moggi	Italy	Gerhard Wöeginger	The Netherlands
Catuscia Palamidessi	France	Uri Zwick	Israel

Past Presidents:

Maurice Nivat	(1972–1977)	Mike Paterson	(1977–1979)
Arto Salomaa	(1979–1985)	Grzegorz Rozenberg	(1985–1994)
Wilfred Brauer	(1994–1997)	Josep Díaz	(1997–2002)

EATCS Council Members

EMAIL ADDRESSES

Luca Aceto	luca@ru.is
Giorgio Ausiello	ausiello@dis.uniroma1.it
Wilfried Brauer	brauer@informatik.tu-muenchen.de
Josep Díaz	diaz@lsi.upc.es
Zoltán Ésik	ze@inf.u-szeged.hu
Alan Gibbons	amg@dcs.kcl.ac.uk
Giuseppe F. Italiano	italiano@disp.uniroma2.it
Dirk Janssens	Dirk.Janssens@ua.ac.be
Jean-Pierre Jouannaud	jouannaud@lix.polytechnique.fr
Juhani Karhumäki	karhumak@cs.utu.fi
Jan van Leeuwen	jan@cs.uu.nl
Eugenio Moggi	moggi@disi.unige.it
Mogens Nielsen	mn@brics.dk
Catuscia Palamidessi	catuscia@lix.polytechnique.fr
David Peleg	peleg@wisdom.weizmann.ac.il
Jiří Sgall	sgall@math.cas.cz
Branislav Rovan	rovan@fmph.uniba.sk
Grzegorz Rozenberg	rozenber@liacs.nl
Arto Salomaa	asalomaa@utu.fi
Don Sannella	dts@dcs.ed.ac.uk
Vladimiro Sassone	vs@ecs.soton.ac.uk
Paul Spirakis	spirakis@cti.gr
Andrzej Tarlecki	tarlecki@mimuw.edu.pl
Wolfgang Thomas	thomas@informatik.rwth-aachen.de
Ingo Wegener	ingo.wegener@uni-dortmund.de
Emo Welzl	emo@inf.ethz.ch
Gerhard Wöeginger	g.j.woeginger@math.utwente.nl
Uri Zwick	zwick@post.tau.ac.il

Bulletin Editor: Vladimiro Sassone, Southampton, United Kingdom
Cartoons: DADARA, Amsterdam, The Netherlands

The bulletin is entirely typeset by pdfTEX and ConTEXt in TXfonts. The Editor is grateful to Uffe H. Engberg, Hans Hagen, Marloes van der Nat, and Grzegorz Rozenberg for their support.

All contributions are to be sent electronically to

`bulletin@eatcs.org`

and must be prepared in LaTeX 2$_\varepsilon$ using the class `beatcs.cls` (a version of the standard LaTeX 2$_\varepsilon$ `article` class). All sources, including figures, and a reference PDF version must be bundled in a ZIP file.

Pictures are accepted in EPS, JPG, PNG, TIFF, MOV or, preferably, in PDF. Photographic reports from conferences must be arranged in ZIP files layed out according to the format described at the Bulletin's web site. Please, consult `http://www.eatcs.org/bulletin/howToSubmit.html`.

We regret we are unfortunately not able to accept submissions in other formats, or indeed submission not *strictly* adhering to the page and font layout set out in `beatcs.cls`. We shall also not be able to include contributions not typeset at camera-ready quality.

The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

The Editor welcomes proposals for surveys, tutorials, and thematic issues of the Bulletin dedicated to currently hot topics, as well as suggestions for new regular sections.

The EATCS home page is `http://www.eatcs.org`

Table of Contents

EATCS MATTERS

Letter from the President 3

Letter from the Editor 4

Foreign Chapters

The Japanese Chapter 6

INSTITUTIONAL SPONSORS

EATCS NEWS

News from Australia, *by C.J. Fidge* 13

News from India, *by M. Mukund* 15

News from Latin America, *by A. Viola* 17

News from New Zealand, *by C.S. Calude* 20

THE EATCS COLUMNS

The Algorithmics Column, *by G. Woeginger*

Sublinear-Time Algorithms, *by A. Czumaj and C. Sohler* 23

The Complexity Column, *by J. Torán*

Learning Boolean Functions under the Uniform Distribution via the Fourier Transform, *by J. Köbler and W. Lindner* 48

The Concurrency Column, *by L. Aceto*

On the Control of Timed and Hybrid Systems, *by P. Bouyer and F. Chevalier* 79

The Distributed Computing Column, *by M. Mavronicolas*

A Game on a Distributed Network, *by V. Papadopoulou* 97

The Formal Language Theory Column, *by A. Salomaa*

Enumeration of Formal Languages, *by M. Domaratzki* 117

The Formal Specification Column, *by H. Ehrig*

Report on ACCAT Workshop at ETAPS 2006, *by H. Ehrig* 134

The Logics in Computer Science Column, *by Y. Gurevich*

Henkin Quantifiers: Logic, Games, and Computation, *by M. Sevenster* 136

TECHNICAL CONTRIBUTIONS

Relations over Words and Logic: a Chronology, *by C. Choffrut* 159

Binary Words with Few Squares, *by T. Harju and D. Nowotka* 164

Fine and Wilf's Theorem for Abelian Periods, *by S. Constantinescu and L. Ilie* 167

Recursion Theorems and Self-Replication via Text Register Machine Programs, *by L. Moss* 171

THE PUZZLE CORNER, *by L. Rosaz* 183

REPORTS FROM CONFERENCES

BCTCS 2006 187

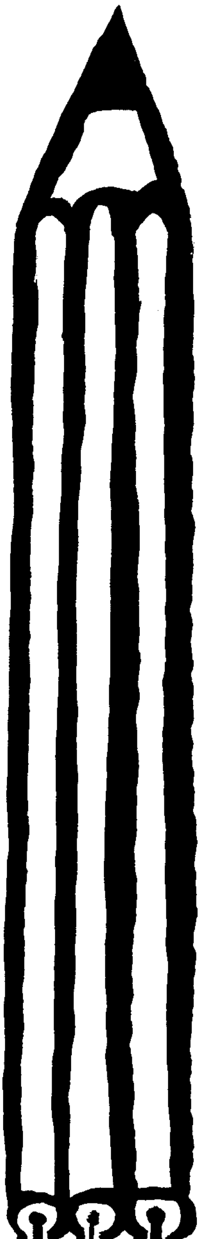
UC 2005 212

OPODIS 2005 214

ABSTRACTS OF PHD THESES 217

EATCS LEAFLET 231

EATCS Matters



Letter from the President

Dear EATCS members,

Time is approaching for ICALP 2006, and we look forward to seeing many of you in Venice this summer. ICALP again this year attracted a very high number of submissions, and as you can see from the official ICALP'06 website, icalp06.dsi.unive.it, we shall have a high-quality scientific event with lots of interesting events, including three co-located conferences and nine workshops. Let me also mention the EATCS Award Ceremony, during which this year's Gödel Prize will be awarded to Manindra Agrawal, Neeraj Kayal, and Nitin Saxena for their paper "PRIMES is in P", Annals of Mathematics 160(2), 781-793, 2004, and the EATCS Distinguished Award will be presented to Professor Mike Paterson, University of Warwick, in recognition of his outstanding scientific contributions to theoretical computer science.

Also, we shall have the annual EATCS General Assembly during ICALP in Venice. As usual, the EATCS annual report will be made available before the General Assembly on our web pages, www.eatcs.org. In this report you will find information on our activities during the past year, as well as figures on EATCS membership and finances. We hope to have a lively discussion during the General Assembly. If you suggestions for specific topics to be put on the agenda, don't hesitate to contact me or any other memebr of the Council.

Mogens Nielsen, Aarhus
May 2006

Letter from the Bulletin Editor

Dear Reader,

Welcome to the June 2006 issue of the Bulletin of the EATCS, which among the usual variety of contributions hosts the first installment of Gerhard Woeginger's Algorithmics Column. *Gerhard is a professor at Eidhoven University of Technology, where he heads the Combinatorial Optimization group. After completing his PhD in 1991 at Graz University of Technology, he held academic positions in Graz, Twente and Eindhoven. His research interests focus on optimisation problems, including polyhedral techniques, local search, online routing, performance guarantees for approximation algorithms. As Gerhard takes over the column, we wish him a long and fruitful collaboration with the Bulletin.*

Regarding the rest of the regular columns, several surveys stand out. These include Bouyer *and* Chevalier*'s paper on the control of timed and hybrid systems,* Michael Domaratzki*'s 'Enumeration of Formal Languages,'* Köbler *and* Lindner*'s survey on learning via the Fourier transform, and* Merlijn Sevenster*'s work on Henkin quantifiers.*

The volume presents four refereed contributions, including Larry Moss*'s on self-replication, and is closed by a set of three reports from international meetings. I am particularly glad to draw your attention to Faron Moller's thorough report from the annual* British Colloquium on Theoretical Computer Science, *BCTCS 2006. Reports from conferences also feature in Alfredo Viola's* News from Latin America, *and in and Hartmut Ehrig's* Formal Specification column.

Enjoy

Vladimiro Sassone, Southampton
June 2006

Gilles Kahn

IN MEMORIAM

17.04.1946 – 9.02.2006

An impassioned scientist, an exceptional researcher, and a visionary director, Gilles Kahn profoundly marked the French and international scientific communities. He was a great man of science.

REPORT FROM THE JAPANESE CHAPTER

K. Makino (Tokyo Univ.)

Nippon from South to North

The Last Article!!

Quite a while ago, early 1992, I thought of a new series for this column — *Nippon from South to North*. The idea is to ask someone in each area of Japan to introduce TCS researchers around him/her with some personal comments about them. This was a nice idea because I do not have to worry about an article at least for the June issue ;-)

I decided to start from *Kyushu*, the major southmost island, simply because I knew that Professor *Satoru Miyano* is the best person to ask the first article of the series. Since then I have been asking several researchers to write a report on local TCS researchers. Then finally, we came up to the major northmost island, *Hokkaido*, and with this 15th article, I can happily close this series.

I would like to express my sincere thanks to the contributors of this series, and thank the readers to their interest to this series!

Osamu Watanabe
Tokyo Inst. of Tech.

TCS Researchers in Hokkaido University

Prof. A. Namamura

Hokkaido is the northernmost and second-largest island in Japan. It is only 1.5 hour by air from Tokyo to Hokkaido, but the climate is quite different to that of the mainland. Colder and drier, it is a popular destination during both the hot summer months and the winter ski season. Transparent lakes, a large purple carpet of lavender, vast ice floes, illuminated large snow statues in Sapporo snow festival . . . there are many attractions for travelers.

Sapporo, the capital of Hokkaido, is the fifth largest city in Japan, and the population is about 1.9 million. Sapporo is one of the most comfortable cities to live in Japan. Living cost here is not so high comparing to that in other large cities. It is a planned city developed in 19th century, and is organized in a grid pattern. Since the address is specified as like North 12 West 5, which means the 12th north and the 5th west block from the center of the city, it is easy to reach a destination without losing your way. Sea food here is delicious and you can eat fresh Sushi with reasonable price. In an hour from the center of the city, you can reach several ski slopes, Jozankei Spa and Chitose international airport.

Hokkaido University, established in 1876 as Sapporo Agricultural College, was the first college in Japan to award bachelor degrees. The first Vice President Dr. Clark's words, "Boys, be ambitious!", are one of the most well-known phrases among Japanese people.

In Hokkaido University, now one of main Japanese universities, researches in almost all academic disciplines are conducted. As for TCS researchers, the number of them had increased twice when Graduate School of Information Science and Technology was established in April, 2004. Now, let's introduce TCS researchers in Hokkaido University.

Yuzuru Tanaka is working on database and VLSI algorithms. He is the founder of Meme Media Laboratory and well-known as the developer of *IntelligentPad*, a new-era software architecture.

Thomas Zeugmann joined from University of Lübeck is working on computational learning theory, especially, inductive inference. Since he was also at Kyusyu University from 1997 to 2000, he is very familiar with Japanese culture. Now, he is the steering committee chair of the Algorithmic Learning Theory workshop. He is a good-looking guy!

Makoto Haraguchi is working on analogical reasoning. This year he is the head of Department of Electronics and Information Engineering, and having very busy days.

Hiroki Arimura joined from Kyusyu University is working on computational learning theory and semi-structured data mining. He is a very nice guy and the first person who takes on any troublesome job.

Shin-ichi Minato joined from NTT laboratories is a specialist on binary decision diagrams. His doctor thesis was published by Kluwer Academic Publishers. He is a good violinist and a local orchestra member.

Tetsuya Yoshida joined from Osaka University is a specialist of machine learning and data mining. He is working on efficient algorithms for extracting interesting substructures from collections of graphs.

Takuya Kida joined from Kyusyu University is working on string pattern matching, especially, that in a compressed text. Last year, he was the youngest associate professor in Faculty of Engineering.

Kimihito Ito is working on logic programming and web application. He is now belonging to Research Center for Zoonosis Control and trying to predict the next spreading influenza virus.

Yoshiaki Okubo is working on data abstraction. He is belonging to the same laboratory as Prof. Haraguchi, and is busy for supporting the busy professor.

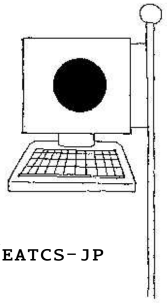
Jan Poland joined from Swiss institute for Artificial Intelligence is a post doctor fellow in Zeugmann research group. He is working on learning theory.

Atsuyoshi Nakamura, who is writing this article, was joined from NEC Company four years ago. He is working on computational learning theory and its application on WWW. He thought Hokkaido is too cold for human being to live before, but now he thinks Hokkaido is the best place to live.

Hokkaido university is huge and the author's experience here is only four years, so there might be many other TCS researchers the author has not yet been aware of. Members introduced above belong to CS division, and researches by the other members in CS division are also based on mathematics such as statistics, functional analysis, fuzzy logic and so on. Members with different mathematical bases in CS division have been progressing by positively affecting each other.

Finally, all CS division members welcome TCS researchers' visits to Hokkaido University. Please keep in mind that Hokkaido is the best place to visit in Japan!

Atsuyoshi Nakamura
Graduate School of Information Science and Technology
Hokkaido University
atsu@main.ist.hokudai.ac.jp



The Japanese Chapter

Institutional Sponsors

BRICS, Basic Research in Computer Science,
Aarhus, Denmark

Elsevier Science
Amsterdam, The Netherlands

IPA, Institute for Programming Research and Algorithms,
Eindhoven, The Netherlands

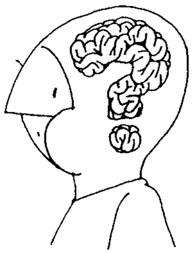
Microsoft Research,
Cambridge, United Kingdom

PWS, Publishing Company,
Boston, USA

TUCS, Turku Center for Computer Science,
Turku, Finland

UNU/IIST, UN University, Int. Inst. for Software Technology,
Macau, China

EATCS News



News from Australia

BY

C.J. FIDGE

School of Engineering and Data Communications
Queensland University of Technology, Brisbane, Australia
http://www.fit.qut.edu.au/~fidgec

Over the past twelve months, researchers in Australian universities have been obsessed by three letters: RQF. This abbreviation stands for the Research Quality Framework, the Australian Government's newly-proposed scheme for allocating research funds to the tertiary education sector.

For as long as most of us care to remember, the Australian Government's Department of Education, Science and Training has run the annual Higher Education Research Data Collection process. Each year Australian university administrators have been required to prepare a report on the number and type of publications produced by their academic and research staff, and the number and value of research grants awarded to academic staff. These figures are used to decide how much money the government will give to each university as part of its annual operating grant. See `http://www.dest.gov.au/sectors/research_sector/online_forms_services/higher_education_research_data_collection.htm`.

Not surprisingly, this method of allocating funds has attracted considerable criticism, especially because it emphasises quantity over quality. In particular, it encourages Australian researchers to publish large numbers of low quality papers. Studies have shown that since the system was introduced there has been a steady increase in the number of research papers published by Australian academics and a corresponding decrease in the 'impact' of these papers as measured by citation indices.

Early last year, in response to these criticisms, the Minister for Education, Science and Training, Dr Brendan Nelson, announced that a new funding allocation scheme would be developed, the Research Quality Framework. An Expert Advisory Group was established to examine best practice in this area and report back to the government on the 'Preferred Model' for its implementation.

While this was being done, the 38 Australian universities went into a frenzy, trying to second guess what the model would be and jostling to position themselves in the best possible way with respect to it. The Australian Technology Network universities, of which the Queensland University of Technology is a member, even conducted a full-scale trial of the anticipated reporting process, to assess how much it would cost to implement and how each university would fare.

A document describing the Advisory Group's proposal appeared in September 2005 (`http://www.dest.gov.au/sectors/research_sector/policies_issues_reviews/key_issues/research_quality_framework/rqf_preferred_model.htm`). The approach was influenced by the scheme currently used in the UK. It describes a system in which each university nominates 'Research Groupings' of academics and researchers in various disciplines. For each such group several outstanding researchers are selected and their four best research outputs are described in an 'Evidence Portfolio'. This includes proof of the 'impact' of the research output in terms of citation counts, industrial uptake, etc. An RQF Moderation Panel will then produce a report based on this evidence which determines the amount of funding made available to each university.

Since then, however, the whole process has been thrown into confusion. Firstly, the timeline for the RQF's introduction has slipped from 'early 2006' to 'mid 2006' and now to '2007'. Secondly, news from Britain indicates that that country intends to abandon its own RQF-like process in favour of a simpler model just like the one Australia currently uses, making the wisdom of the whole approach questionable. Thirdly, the costs of administering the proposed model have been disclosed to be extremely high. *The Australian* newspaper recently reported that the government has set aside \$3 million just for establishment of the RQF, while the Australian universities are estimating that it will cost them \$50 million to prepare for it. Finally, the impetus for the reform has been diminished by a cabinet reshuffle which has seen the minister pushing to change the existing funding model replaced. The latest word is that the new minister will continue the policies of her predecessor, but for now the universities will just have to wait and see.

Lastly, to end on a more positive note, this is my annual reminder to consider escaping the northern hemisphere winter and take a summer holiday in Australia to attend *Computing: The Australasian Theory Symposium* (CATS) in January/February 2007. Research paper submissions are due on the 11th of August. Full details are available at `http://www-staff.it.uts.edu.au/~cbj/cats07/CATS07.html`.

News from India

by

Madhavan Mukund
Chennai Mathematical Institute
Chennai, India
madhavan@cmi.ac.in

In this edition of News from India, we look ahead to some of the schools, workshops and conferences coming up during the second half of 2006.

Summer School on Algorithms, Complexity, Cryptology. A summer school on *Algorithms, Complexity and Cryptology* is being organized in Bangalore from May 22 to June 9, 2006 by Microsoft Research India and the IISc Mathematics Initiative, Indian Institute of Science, Bangalore.

The school is aimed at senior undergraduate students, graduate students, research scholars and faculty members and will focus on the following topics: Factoring problem and Linear Algebra, Pairing Based Cryptosystems, A Cryptographers view of System Security, Recent problems in Linear Algebra, Auctions and Game Theory, Signal Processing for Applications with a security twist, Authentication, hashing and Watermarking, Elliptic Curves for Undergraduates, and Elliptic Curves and Cryptography.

The list of speakers includes Dan Boneh (Stanford, USA), Kamal Jain (Microsoft Research, USA), David Jao (Microsoft Research, USA), Ravi Kannan (Yale University, USA), Kivanc Mihcak (Microsoft Research, USA), A. Shamir (Weizmann Institute, Israel), and Eran Tromer (Weizmann Institute, Israel).

More information about the school available at http://math.iisc.ernet.in/~imi/sacc.htm.

Formal Methods Update Meeting. During the past few years, the Indian Association for Research in Computing Science (IARCS) has organized regular "update" meetings in the area of formal methods. The meetings are intended as a forum for Indian researchers and students in theoretical computer science to update themselves on current trends and to explore new research areas.

This year's meeting will be held at IIT Guwahati from 3–6, July 2006. As in previous years, the meeting will be devoted to talks, tutorials, discussions and presentations by researchers on recent trends and advances in the field. The broad theme this year is *Verification of Infinite State Systems*. The last meeting was held in IIT Bombay in July, 2005, with the theme *Advances in Concurrency, Logic and Verification*. Earlier meetings were at IMSc, Chennai with themes *Models for Programs*, *Timed Systems* and *Automata and Verification*.

More information about the workshop is available at `http://www.iitg.ernet.in/cse/fac/pbhaduri/update06`.

SEFM 2006. SEFM 2006, the 4th IEEE International Conference on Software Engineering and Formal Methods, will be held in Pune, India during the period September 11–15, 2006. The aim of the conference is to bring together practitioners and researchers from academia, industry and government to advance the state-of-the-art in formal methods, to scale up their application in software industry and to encourage their integration with practical engineering methods.

The Program Committee for SEFM 2006 is jointly chaired by Paritosh Pandya (TIFR, Mumbai, India) and Dang Van Hung (UNU-IIST, Macao, China). This year's invited speakers are Sriram Rajamani (Microsoft Research India, India), John Rusby (SRI International, USA), Joseph Sifakis (CNRS and VERIMAG, France) and Bertrand Meyer (ETH Zurich, Switzerland).

The website for SEFM 2006 is at `http://www.iist.unu.edu/SEFM06`.

FSTTCS 2006. The 26th edition of FSTTCS will take place from December 13–15, 2006 at the Indian Statistical Institute, Kolkata. It will be preceded by two pre-conference workshops on December 11–12, 2006.

The Program Committee is co-chaired by S. Arun-Kumar and Naveen Garg from IIT, Delhi. The list of invited speakers for FSTTCS 2006 includes Gordon Plotkin (Edinburgh, UK), Emo Welzl (ETH Zurich, Switzerland), Gérard Boudol (INRIA, Sophia Antipolis, France), David Shmoys (Cornell, USA), and Eugene Asarin (LIAFA, Paris 7, France).

The conference website is at `http://www.fsttcs.org`.

ISAAC 2006. The 17th International Symposium on Algorithms and Computation (ISAAC 2006) will take place in Kolkata, India. The Program Committee is chaird by Tetsuo Asano (JAIST, Japan). The invited speakers at ISAAC 2006 are Tamal Dey, (Ohio State, USA) and Kazuo Iwama (Kyoto, Japan)

The conference website is at `http://www.isical.ac.in/~isaac06`.

Madhavan Mukund <`madhavan@cmi.ac.in`>
Chennai Mathematical Institute

News from Latin America

BY

Alfredo Viola

Instituto de Computación, Facultad de Ingenierìa
Universidad de la República
Casilla de Correo 16120, Distrito 6, Montevideo, Uruguay
viola@fing.edu.uy

In this issue I present the reports of ITW 2006 and LATIN 2006 and the Call for Participation of SBMF 2006. At the end I present a list of the main events in Theoretical Computer Science to be held in Latin America in the following months.

Report of ITW 2006 (by Sergio Verdú).

The 2006 IEEE Information Theory Workshop has taken place in Punta del Este, Uruguay, on March 13-17, 2006. Gadiel Seroussi and Alfredo "Tuba" Viola chaired the Workshop and Ron Roth and Marcelo Weinberger chaired the Program Committee.

Playground of the Argentinean upper crust, Punta de Este is a Summer resort of manicured lawns, manor houses, high-rise apartments, and endless beaches.

With 155 registrants the workshop was a great success technically and otherwise. Uruguay and the United States supplied two thirds of the attendees (in roughly equal parts), with the rest hailing from Argentina, Australia, Austria, Brazil, Britain, Canada, Finland, France, Germany, Greece, Hungary, Israel, Italy, Japan, Korea, Mexico, Norway, Portugal, Spain, and Switzerland.

Elwyn Berlekamp and Jorma Rissanen gave the keynote lectures and the invited and contributed sessions covered most major topics within the purview of the IT Society. The full lineup of talks along with a photo gallery can be seen at `http://www.fing.edu.uy/itw06`.

In addition to a superbly produced proceedings (CD-ROM available from IEEE), the registration package included a kaleidoscope, a professionally designed poster with an enigmatic binary inscription, a T-shirt, an ITW cap (sure to become a sought-after item in e-bay), and a tube of sunscreen (which, alas, remained unopened).

A visit to the atelier of Carlos Paez Vilaro took place on Wednesday. Not only were we able to chat with the jovial 82 year-old renowned artist, but we witnessed a breathtaking sunset from his cliffside house.

Performing at the end of the banquet, a professional troupe of musicians, singers and dancers, were joined by a number of workshop participants who showcased their dancing skills in fields such as tango, milonga and candombe. No information theorist seemed to be quite ready to quit their day job.

Overlapping with ITW2006, the superb facilities of the Conrad Resort, hosted the Miss Uruguay beauty contest. Rumor has it that some ITW participants skipped a paper or two to peek in the rehearsals of that event. Which just goes to show the proverbial wide range of interests of our members.

Report of LATIN 2006 (by Avi Wigderson).

LATIN 2006 was held in the charming city of Valdivia, in the Lake District of Chile, during March 20-24. Both the scientific program and local arrangements were chaired by Marcos Kiwi, who has done an excellent job on both fronts. Attendants from all over the world, seasoned by many conferences, voiced their excitement about the high level of care by Marcos and his team to make this meeting enjoyable and productive. Among the highlights was the tour to the smoke-puffing volcano Villarica, and the lava caves on it. The perfect organization included perfect weather for all by one day of the meeting - almost unheard of in this rainy part of the world.

LATIN 2006 received a record number of 224 submissions and 66 papers were accepted (29.5 %). Moreover, 115 registrants from 22 different countries participated in the Conference. Chile, Canada and the United Stated contributed around half of the attendees with the rest hailing form Brazil, China, Egypt, Finland, France, Germany, Greece, Hong-Kong, Israel, Italy, Japan, Mexico, Norway, Sweden, Switzerland, Uruguay, UK, and Spain.

Gastón Gonnet and Ricardo Baeza-Yates ended their term as members of the steering committee. Its current members are: Martín Farach-Colton, Marcos Kiwi, Yoshiharu Kohayakawa, Daniel Panario, Sergio Rajsbaum, and Gadiel Seroussi. See `http://www.latintcs.org` for details.

SBMF 2006 - Call for Participation.

The Brazilian Symposium on Formal Methods (SBMF) is the official event of the Brazilian Computer Society (SBC) in the area of formal methods. The event is annual, joining up to more than 100 participants registered each year. Beyond technical sessions, tutorials and mini-courses, the symposium also presents invited speakers from the international community. A selection of the conference proceedings is published in the Electronic Notes in Theoretical Computer Science series from Elsevier. In 2006, the symposium will be held in Natal, the largest city of Rio Grande do Norte and its capital, jointly with the International Conference on Graph Transformations as well as its satellite events.

Regional Events

- May 15 - 17, 2006, Itatiaia, RJ, Brazil: SBLP 2006 - 10th Brazilian Symposium on Programming Languages (`http://sblp.ime.eb.br/`).
- August 20 - 25, 2006, Santiago de Chile, Chile: 19th IFIP World Computer Congress 2006 (`http://www.wcc-2006.org`).
- August 22 - 24, 2006, Santiago de Chile, Chile: 4th IFIP International Conference on Theoretical Computer Science (`http://www.wcc-2006.org`).
- September 17 - 23, 2006, Natal, RN, Brazil: SMBF 2006 - Brazilian Symposium on Formal Methods (`http://www.dimap.ufrn.br/sbmf2006`).
- September 17 - 23, 2006, Natal, RN, Brazil: ICGT 2006 - International Conference on Graph Transformation (`http://www.dimap.ufrn.br/icgt2006`).
- September 18 - 22, 2006, San Luis Potosí, México: ENC 2006 - Encuentro Internacional de Ciencias de la Computación (`http://enc.smcc.org.mx`).
- November 27 - 30, 2006, Montevideo, Uruguay: XIII CLAIO - Congreso Latino-Iberoamericano de Investigaciónn Operativa (`http://www.fing.edu.uy/inco/eventos/claio2006`).
- January 10 - 13, 2007, Buenos Aires, Argentina: Conference on Logic Computability and Randomness 2007 (`http://www.dc.uba.ar/people/logic2007`).

News from New Zealand

BY

C.S. Calude

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

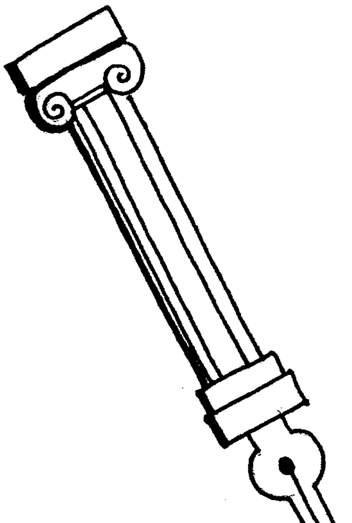
0. The Fifth International Conference UC06 will be held at the University of York, UK, on 4-8 September 2006. http://www.cs.york.ac.uk/nature/uc06/index.php.

Keynote Speakers are: Rainer Blatt (Innsbruck, Austria); Gerard Dreyfus (ESPCI, Paris, France); Michael C. Mozer (University of Colorado, USA); Erik Winfree (CalTech, USA); Damien Woods (University College Cork, Ireland).

1. The latest CDMTCS research reports are (http://www.cs.auckland.ac.nz/staff-cgi-bin/mjd/secondcgi.pl):

276. A. Juarna and V. Vajnovski. Combinatorial Isomorphisms Beyond a Simion -Schmidt's Bijection, 01/2006
277. C. S. Calude, E. Calude and M. J. Dinneen. A New Measure of the Difficulty of Problems, 02/2006
278. S. Schwarz. Lukasiewvicz Logics and Weighted Logics over MV Semirings, 05/2006
279. L. Staiger. The Kolmogorov Complexity of Infinite Words, 05/2006.

The EATCS Columns



THE ALGORITHMICS COLUMN

BY

GERHARD J WOEGINGER

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
gwoegi@win.tue.nl

SUBLINEAR-TIME ALGORITHMS [*]

Artur Czumaj [†] Christian Sohler [‡]

Abstract

We survey recent advances in the area of sublinear-time algorithms.

1 Introduction

The area of *sublinear-time algorithms* is a new rapidly emerging area of computer science. It has its roots in the study of massive data sets that occur more and more frequently in various applications. Financial transactions with billions of input data and Internet traffic analyses (Internet traffic logs, clickstreams, web data) are examples of modern data sets that show unprecedented scale. Managing and analyzing such data sets forces us to reconsider the traditional notions of efficient algorithms: processing such massive data sets in more than linear time is by far too expensive and often even linear time algorithms may be too slow. Hence, there

[*]Research supported in part by NSF ITR grant CCR-0313219 and NSF DMS grant 0354600.

[†]Department of Computer Science, New Jersey Institute of Technology and Department of Computer Science, University of Warwick. Email: aczumaj@acm.org

[‡]Department of Computer Science, Rutgers University and Heinz Nixdorf Institute, University of Paderborn. Email: csohler@uni-paderborn.de

is the desire to develop algorithms whose running times are not only polynomial, but in fact are *sublinear* in *n*.

Constructing a sublinear time algorithm may seem to be an impossible task since it allows one to read only a small fraction of the input. However, in recent years, we have seen development of sublinear time algorithms for optimization problems arising in such diverse areas as graph theory, geometry, algebraic computations, and computer graphics. Initially, the main research focus has been on designing efficient algorithms in the framework of *property testing* (for excellent surveys, see [26, 30, 31, 40, 49]), which is an alternative notion of approximation for decision problems. But more recently, we see some major progress in sublinear-time algorithms in the classical model of randomized and approximation algorithms. In this paper, we survey some of the recent advances in this area. Our main focus is on sublinear-time algorithms for combinatorial problems, especially for graph problems and optimization problems in metric spaces.

Our goal is to give a flavor of the area of sublinear-time algorithms. We focus on the most representative results in the area and we aim to illustrate main techniques used to design sublinear-time algorithms. Still, many of the details of the presented results are omitted and we recommend the readers to follow the original works. We also do not aim to cover the entire area of sublinear-time algorithms, and in particular, we do not discuss property testing algorithms [26, 30, 31, 40, 49], even though this area is very closely related to the research presented in this survey.

Organization. We begin with an introduction to the area and then we give some sublinear-time algorithms for a basic problem in computational geometry [14]. Next, we present recent sublinear-time algorithms for basic graph problems: approximating the average degree in a graph [25, 34] and estimating the cost of a minimum spanning tree [15]. Then, we discuss sublinear-time algorithms for optimization problems in metric spaces. We present the main ideas behind recent algorithms for estimating the cost of minimum spanning tree [19] and facility location [10], and then we discuss the quality of random sampling to obtain sublinear-time algorithms for clustering problems [20, 46]. We finish with some conclusions.

2 Basic Sublinear Algorithms

The concept of sublinear-time algorithms is known for a very long time, but initially it has been used to denote "pseudo-sublinear-time" algorithms, where after an appropriate *preprocessing*, an algorithm solves the problem in sublinear-time.

For example, if we have a set of n numbers, then after an $O(n \log n)$ preprocessing (sorting), we can trivially solve a number of problems involving the input elements. And so, if the after the preprocessing the elements are put in a sorted array, then in $O(1)$ time we can find the kth smallest element, in $O(\log n)$ time we can test if the input contains a given element x, and also in $O(\log n)$ time we can return the number of elements equal to a given element x. Even though all these results are folklore, this is not what we call nowadays a sublinear-time algorithm.

In this survey, our goal is to study algorithms for which the input is taken to be in any standard representation and with no extra assumptions. Then, an algorithm does not have to read the entire input but it may determine the output by checking only a subset of the input elements. It is easy to see that for many natural problems it is impossible to give any reasonable answer if not all or almost all input elements are checked. But still, for some number of problems we can obtain good algorithms that do not have to look at the entire input. Typically, these algorithms are *randomized* (because most of the problems have a trivial linear-time deterministic lower bound) and they return only an *approximate* solution rather than the exact one (because usually, without looking at the whole input we cannot determine the exact solution). In this survey, we present recently developed sublinear-time algorithm for some combinatorial optimization problems.

Searching in a sorted *list*. It is well-known that if we can store the input in a sorted array, then we can solve various problems on the input very efficiently. However, the assumption that the input array is sorted is not natural in typical applications. Let us now consider a variant of this problem, where our goal is to *search* for an element x in a linked sorted list containing n *distinct* elements[1]. Here, we assume that the n elements are stored in a doubly-linked, each list element has access to the next and preceding element in the list, and the list is sorted (that is, if x follows y in the list, then $y < x$). We also assume that we have access to all elements in the list, which for example, can correspond to the situation that all n list elements are stored in an array (but the array is not sorted and we do not impose any order for the array elements). How can we find whether a given number x is in our input or is not?

On the first glace, it seems that since we do not have direct access to the rank of any element in the list, this problem requires $\Omega(n)$ time. And indeed, if our goal is to design a deterministic algorithm, then it is impossible to do the search in $o(n)$ time. However, if we allow randomization, then we can complete the search in

[1]The assumption that the input elements are *distinct* is important. If we allow multiple elements to have the same key, then the search problem requires $\Omega(n)$ time. To see this, consider the input in which about a half of the elements has key 1, another half has key 3, and there is a single element with key 2. Then, searching for 2 requires $\Omega(n)$ time.

$O(\sqrt{n})$ expected time (and this bound is asymptotically tight).

Let us first sample uniformly at random a set S of $\Theta(\sqrt{n})$ elements from the input. Since we have access to all elements in the list, we can select the set S in $O(\sqrt{n})$ time. Next, we scan all the elements in S and in $O(\sqrt{n})$ time we can find two elements in S, p and q, such that $p \leq x < q$, and there is no element in S that is between p and q. Observe that since the input consist of n distinct numbers, p and q are uniquely defined. Next, we traverse the input list containing all the input elements starting at p until we find either the sought key x or we find element q.

Lemma 1. *The algorithm above completes the search in expected $O(\sqrt{n})$ time. Moreover, no algorithm can solve this problem in $o(\sqrt{n})$ expected time.*

Proof. The running time of the algorithm if equal to $O(\sqrt{n})$ plus the number of the input elements between p and q. Since S contains $\Theta(\sqrt{n})$ elements, the expected number of input elements between p and q is $O(n/|S|) = O(\sqrt{n})$. This implies that the expected running time of the algorithm is $O(\sqrt{n})$.

For a proof of a lower bound of $\Omega(\sqrt{n})$ expected time, see, e.g., [14]. $\square$

2.1 Geometry: Intersection of Two Polygons

Let us consider a related problem but this time in a geometric setting. Given two convex polygons A and B in $\mathbb{R}^2$, each with n vertices, determine if they intersect, and if so, then find a point in their intersection.

It is well known that this problem can be solved in $O(n)$ time, for example, by observing that it can be described as a linear programming instance in 2-dimensions, a problem which is known to have a linear-time algorithm (cf. [24]). In fact, within the same time one can either find a point that is in the intersection of A and B, or find a line $\mathcal{L}$ that separates A from B (actually, one can even find a bitangent separating line $\mathcal{L}$, i.e., a line separating A and B which intersects with each of A and B in exactly one point). The question is whether we can obtain a better running time.

The complexity of this problem depends on the input representation. In the most powerful model, if the vertices of both polygons are stored in an array in cyclic order, Chazelle and Dobkin [13] showed that the intersection of the polygons can be determined in logarithmic time. However, a standard geometric representation assumes that the input is not stored in an array but rather A and B are given by their doubly-linked lists of vertices such that each vertex has as its successor the next vertex of the polygon in the clockwise order. Can we then test if A and B intersect?

Chazelle et al. [14] gave an $O(\sqrt{n})$-time algorithm that reuses the approach discussed above for searching in a sorted list. Let us first sample uniformly at

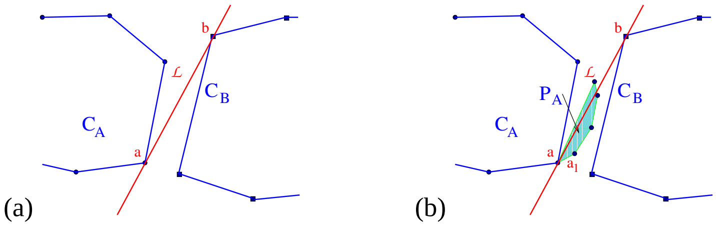


Figure 1: *(a) Bitangent line $\mathcal{L}$ separating C_A and C_B, and (b) the polygon P_A.*

random $\Theta(\sqrt{n})$ vertices from each A and B, and let C_A and C_B be the convex hulls of the sample point sets for the polygons A and B, respectively. Using the linear-time algorithm mentioned above, in $O(\sqrt{n})$ time we can check if C_A and C_B intersects. If they do, then the algorithm will get us a point that lies in the intersection of C_A and C_B, and hence, this point lies also in the intersection of A and B. Otherwise, let $\mathcal{L}$ be the bitangent separating line returned by the algorithm (see Figure 1 (a)).

Let a and b be the points in $\mathcal{L}$ that belong to A and B, respectively. Let a_1 and a_2 be the two vertices adjacent to a in A. We will define now a new polygon P_A. If none of a_1 and a_2 is on the side C_A of $\mathcal{L}$ the we define P_A to be empty. Otherwise, exactly one of a_1 and a_2 is on the side C_A of $\mathcal{L}$; let it be a_1. We define polygon P_A by walking from a to a_1 and then continue walking along the boundary of A until we cross $\mathcal{L}$ again (see Figure 1 (b)). In a similar way we define polygon P_B. Observe that the expected size of each of P_A and P_B is at most $O(\sqrt{n})$.

It is easy to see that A and B intersects if and only if either A intersects P_B or B intersects P_A. We only consider the case of checking if A intersects P_B. We first determine if C_A intersects P_B. If yes, then we are done. Otherwise, let $\mathcal{L}_A$ be a bitangent separating line that separates C_A from P_B. We use the same construction as above to determine a subpolygon Q_A of A that lies on the P_B side of $\mathcal{L}_A$. Then, A intersects P_B if and only if Q_A intersects P_B. Since Q_A has expected size $O(\sqrt{n})$ and so does P_B, testing the intersection of these two polygons can be done in $O(\sqrt{n})$ expected time. Therefore, by our construction above, we have solved the problem of determining if two polygons of size n intersect by reducing it to a constant number of problem instances of determining if two polygons of expected size $O(\sqrt{n})$ intersect. This leads to the following lemma.

Lemma 2. **[14]** *The problem of determining whether two convex n-gons intersect can be solved in $O(\sqrt{n})$ expected time, which is asymptotically optimal.*

Chazelle et al. [14] gave not only this result, but they also showed how to apply a similar approach to design a number of sublinear-time algorithms for some basic geometric problems. For example, one can extend the result discussed above to test the intersection of two convex polyhedra in $\mathbb{R}^3$ with n vertices in $O(\sqrt{n})$ expected time. One can also approximate the volume of an n-vertex convex polytope to within a relative error $\varepsilon > 0$ in expected time $O(\sqrt{n}/\varepsilon)$. Or even, for a pair of two points on the boundary of a convex polytope P with n vertices, one can estimate the length of an optimal shortest path outside P between the given points in $O(\sqrt{n})$ expected time.

In all the results mentioned above, the input objects have been represented by a linked structure: either every point has access to its adjacent vertices in the polygon in $\mathbb{R}^2$, or the polytope is defined by a doubly-connected edge list, or so. These input representations are standard in computational geometry, but a natural question is whether this is necessary to achieve sublinear-time algorithms — what can we do if the input polygon/polytop is represented by a set of points and no additional structure is provided to the algorithm? In such a scenario, it is easy to see that no $o(n)$-time algorithm can solve exactly any of the problems discussed above. That is, for example, to determine if two polygons with n vertices intersect one needs $\Omega(n)$ time. However, still, we can obtain some approximation to this problem, one which is described in the framework of *property testing*.

Suppose that we relax our task and instead of determining if two (convex) polytopes A and B in $\mathbb{R}^d$ intersects, we just want to distinguish between two cases: either A and B are intersection-free, or one has to "significantly modify" A and B to make them intersection-free. The definition of the notion of "significantly modify" may depend on the application at hand, but the most natural characterization would be to remove at least εn points in A and B, for an appropriate parameter ε (see [18] for a discussion about other geometric characterization). Czumaj et al. [23] gave a simple algorithm that for any $\varepsilon > 0$, can distinguish between the case when A and B do not intersect, and the case when at least εn points has to be removed from A and B to make them intersection-free: the algorithm returns the outcome of a test if a random sample of $O((d/\varepsilon)\ \log(d/\varepsilon))$ points from A intersects with a random sample of $O((d/\varepsilon)\ \log(d/\varepsilon))$ points from B.

Sublinear-time algorithms: perspective. The algorithms presented in this section should give a flavor of the area and give us the first impression of what do we mean by sublinear-time and what kind of results one can expect. In the following sections, we will present more elaborate algorithms for various combinatorial problems for graphs and for metric spaces.

3 Sublinear Time Algorithms for Graph Problems

In the previous section, we introduced the concept of sublinear-time algorithms and we presented two basic sublinear-time algorithms for geometric problems. In this section, we will discuss sublinear-time algorithms for graph problems. Our main focus is on sublinear-time algorithms for graphs, with special emphasizes on sparse graphs represented by adjacency lists where combinatorial algorithms are sought.

3.1 Approximating the Average Degree

Assume we have access to the degree distribution of the vertices of an undirected connected graph $G = (V, E)$, i.e., for any vertex $v \in V$ we can query for its degree. Can we achieve a good approximation of the average degree in G by looking at a sublinear number of vertices? At first sight, this seems to be an impossible task. It seems that approximating the average degree is equivalent to approximating the average of a set of n numbers with values between 1 and $n - 1$, which is not possible in sublinear time. However, Feige [25] proved that one can approximate the average degree in $O(\sqrt{n}/\varepsilon)$ time within a factor of $2 + \varepsilon$.

The difficulty with approximating the average of a set of n numbers can be illustrated with the following example. Assume that almost all numbers in the input set are 1 and a few of them are $n - 1$. To approximate the average we need to approximate how many occurrences of $n - 1$ exist. If there is only a constant number of them, we can do this only by looking at $\Omega(n)$ numbers in the set. So, the problem is that these large numbers can "hide" in the set and we cannot give a good approximation, unless we can "find" at least some of them.

Why is the problem less difficult, if, instead of an arbitrary set of numbers, we have a set of numbers that are the vertex degrees of a graph? For example, we could still have a few vertices of degree $n - 1$. The point is that in this case any edge incident to such a vertex can be seen at another vertex. Thus, even if we do not sample a vertex with high degree we will see all incident edges at other vertices in the graph. Hence, vertices with a large degree cannot "hide."

We will sketch a proof of a slightly weaker result than that originally proven by Feige [25]. Let d denote the average degree in $G = (V, E)$ and let d_S denote the random variable for the average degree of a set S of s vertices chosen uniformly at random from V. We will show that if we set $s \geq \beta \sqrt{n}/\varepsilon^{O(1)}$ for an appropriate constant β, then $d_S \geq (\frac{1}{2} - \varepsilon) \cdot d$ with probability at least $1 - \varepsilon/64$. Additionally, we observe that Markov inequality immediately implies that $d_S \leq (1 + \varepsilon) \cdot d$ with probability at least $1 - 1/(1 + \varepsilon) \geq \varepsilon/2$. Therefore, our algorithm will pick $8/\varepsilon$ sets S_i, each of size s, and output the set with the smallest average degree. Hence, the probability that all of the sets S_i have too high average degree is at most $(1 - \varepsilon/2)^{\varepsilon/8} \leq 1/8$. The probability that one of them has too small average degree is at most $\frac{8}{\varepsilon} \cdot \frac{\varepsilon}{64} = 1/8$. Hence, the output value will satisfy both inequalities with probability at least $3/4$. By replacing ε with $\varepsilon/2$, this will yield a $(2 + \varepsilon)$-approximation algorithm.

Now, our goal is to show that with high probability one does not underestimate the average degree too much. Let H be the set of the $\sqrt{\varepsilon n}$ vertices with highest degree in G and let $L = V \setminus H$ be the set of the remaining vertices. We first argue that the sum of the degrees of the vertices in L is at least $(\frac{1}{2} - \varepsilon)$ times the sum of the degrees of all vertices. This can be easily seen by distinguishing between edges incident to a vertex from L and edges within H. Edges incident to a vertex from L contribute with at least 1 to the sum of degrees of vertices in L, which is fine as this is at least $1/2$ of their full contribution. So the only edges that may cause problems are edges within H. However, since $|H| = \sqrt{\varepsilon n}$, there can be at most εn such edges, which is small compared to the overall number of edges (which is at least $n - 1$, since the graph is connected).

Now, let d_H be the degree of a vertex with the smallest degree in H. Since we aim at giving a lower bound on the average degree of the sampled vertices, we can safely assume that all sampled vertices come from the set L. We know that each vertex in L has a degree between 1 and d_H. Let X_i, $1 \le i \le s$, be the random variable for the degree of the ith vertex from S. Then, it follows from Hoeffding bounds that

$$\mathbf{Pr}[\sum_{i=1}^{s} X_i \le (1-\varepsilon) \cdot \mathbf{E}[\sum_{i=1}^{s} X_i]] \;\le\; e^{-\frac{\mathbf{E}[\sum_{i=1}^{r} X_i]\cdot\varepsilon^2}{d_H}} \quad .$$

We know that the average degree is at least $d_H \cdot |H|/n$, because any vertex in H has at least degree d_H. Hence, the average degree of a vertex in L is at least $(\frac{1}{2} - \varepsilon) \cdot d_H \cdot |H|/n$. This just means $\mathbf{E}[X_i] \ge (\frac{1}{2} - \varepsilon) \cdot d_H \cdot |H|/n$. By linearity of expectation we get $\mathbf{E}[\sum_{i=1}^{s} X_i] \ge s \cdot (\frac{1}{2} - \varepsilon) \cdot d_H \cdot |H|/n$. This implies that, for our choice of s, with high probability we have $d_S \ge (\frac{1}{2} - \varepsilon) \cdot d$.

Feige showed the following result, which is stronger with respect to the dependence on ε.

Theorem 3. [25] *Using $O(\varepsilon^{-1} \cdot \sqrt{n/d_0})$ queries, one can estimate the average degree of a graph within a ratio of $(2 + \varepsilon)$, provided that $d \ge d_0$.*

Feige also proved that $\Omega(\varepsilon^{-1} \cdot \sqrt{n/d})$ queries are required, where d is the average degree in the input graph. Finally, any algorithm that uses only degree queries and estimates the average degree within a ratio $2 - \delta$ for some constant δ requires $\Omega(n)$ queries.

Interestingly, if one can also use neighborhood queries, then it is possible to approximate the average degree using $\widetilde{O}(\sqrt{n}/\varepsilon^{O(1)})$ queries with a ratio of $(1 + \varepsilon)$, as shown by Goldreich and Ron [34]. The model for neighborhood queries is as follows. We assume we are given a graph and we can query for the ith neighbor of vertex v. If v has at least i neighbors we get the corresponding neighbor; otherwise we are told that v has less than i neighbors. We remark that one can simulate degree queries in this model with $O(\log n)$ queries. Therefore, the algorithm from [34] uses only neighbor queries.

For a sketch of a proof, let us assume that we know the set H. Then we can use the following approach. We only consider vertices from L. If our sample contains a vertex from H we ignore it. By our analysis above, we know that there are only few edges within H and that we make only a small error in estimating the number of edges within L. We loose the factor of two, because we "see" edges from L to H only from one side. The idea behind the algorithm from [34] is to approximate the fraction of edges from L to H and add it to the final estimate. This has the effect that we count any edge between L and H twice, canceling the effect that we see it only from one side. This is done as follows. For each vertex v we sample from L we take a random set of incident edges to estimate the fraction $\lambda(v)$ of its neighbors that is in H. Let $\hat{\lambda}(v)$ denote the estimate we obtain. Then our estimate for the average degree will be $\sum_{v \in S \cap L}(1+\hat{\lambda}(v)) \cdot d(v)/|S \cap L|$, where $d(v)$ denotes the degree of v. If for all vertices we estimate $\lambda(v)$ within an additive error of ε, the overall error induced by the $\hat{\lambda}$ will be small. This can be achieved with high probability querying $O(\log n/\varepsilon^2)$ random neighbors. Then the output value will be a $(1+\varepsilon)$-approximation of the average degree. The assumption that we know H can be dropped by taking a set of $O(\sqrt{n/\varepsilon})$ vertices and setting H to be the set of vertices with larger degree than all vertices in this set (breaking ties by the vertex number).

(We remark that the outline given above is different from the proof in [34].)

Theorem 4. [34] *Given the ability to make neighbor queries to the input graph G, there exists an algorithm that makes $O(\sqrt{n/d_0} \cdot \varepsilon^{-O(1)})$ queries and approximates the average degree in G to within a ratio of $(1+\varepsilon)$.*

3.2 Minimum Spanning Trees

One of the most fundamental graph problems is to compute a minimum spanning tree. Since the minimum spanning tree is of size linear in the number of vertices, no sublinear algorithm for sparse graphs can exists. It is also know that no constant factor approximation algorithm with $o(n^2)$ query complexity in dense graphs (even in metric spaces) exists [37]. Given these facts, it is somewhat surprising that it is possible to approximate the cost of a minimum spanning tree in sparse graphs [15] as well as in metric spaces [19] to within a factor of $(1+\varepsilon)$.

In the following we will explain the algorithm for sparse graphs by Chazelle et al. [15]. We will prove a slightly weaker result than in [15]. Let $G=(V,E)$ be an undirected connected weighted graph with maximum degree D and integer edge weights from $\{1,\dots,W\}$. We assume that the graph is given in adjacency list representation, i.e., for every vertex v there is a list of its at most D neighbors, which can be accessed from v. Furthermore, we assume that the vertices are stored in an array such that it is possible to select a vertex uniformly at random. We assume also that the values of D and W are known to the algorithm.

The main idea behind the algorithm is to express the cost of a minimum spanning tree as the number of connected components in certain auxiliary subgraphs of G. Then, one runs a randomized algorithm to estimate the number of connected components in each of these subgraphs.

To start with basic intuitions, let us assume that $W = 2$, i.e., the graph has only edges of weight 1 or 2. Let $G^{(1)} = (V, E^{(1)})$ denote the subgraph that contains all edges of weight (at most) 1 and let $c^{(1)}$ be the number of connected components in $G^{(1)}$. It is easy to see that the minimum spanning tree has to link these connected components by edges of weight 2. Since any connected component in $G^{(1)}$ can be spanned by edges of weight 1, any minimum spanning tree of G has $c^{(1)} - 1$ edges of weight 2 and $n-1-(c^{(1)}-1)$ edges of weight 1. Thus, the weight of a minimum spanning tree is

$$n - 1 - (c^{(1)} - 1) + 2 \cdot (c^{(1)} - 1) \quad = \quad n - 2 + c^{(1)} \; = \; n - W + c^{(1)} \quad .$$

Next, let us consider an arbitrary integer value for W. Defining $G^{(i)} = (V, E^{(i)})$, where $E^{(i)}$ is the set of edges in G with weight at most i, one can generalize the formula above to obtain that the cost MST of a minimum spanning tree can be expressed as

$$MST \quad = \quad n - W + \sum_{i=1}^{W-1} c^{(i)} \quad .$$

This gives the following simple algorithm.

```
ApproxMSTWeight($G, \varepsilon$)
    for $i = 1$ to $W - 1$
        Compute estimator $\widehat{c}^{(i)}$ for $c^{(i)}$
    output $\widehat{MST} = n - W + \sum_{i=1}^{W-1} \widehat{c}^{(i)}$
```

Thus, the key question that remains is how to estimate the number of connected components. This is done by the following algorithm.

```
APPROXCONNECTEDCOMPS($G$, $s$)
{ Input: an arbitrary undirected graph G }
{ Output: $\hat{c}$: an estimation of the number of connected components of G }
    choose $s$ vertices $u_1, \dots, u_s$ uniformly at random
    for $i = 1$ to $s$ do
        choose $X$ according to $\mathbf{Pr}[X \geq k] = 1/k$
        run breadth-fist-search (BFS) starting at $u_i$ until either
            (1) the whole connected component containing $u_i$ has
                been explored, or
            (2) $X$ vertices have been explored
        if BFS stopped in case (1) then $b_i = 1$
        else $b_i = 0$
    output $\hat{c} = \frac{n}{s} \sum_{i=1}^{s} b_i$
```

To analyze this algorithm let us fix an arbitrary connected component C and let $|C|$ denote the number of vertices in the connected component. Let c denote the number of connected components in G. We can write

$$\mathbf{E}[b_i] = \sum_{\text{connected component } C} \mathbf{Pr}[u_i \in C] \cdot \mathbf{Pr}[X \geq |C|] = \sum_{\text{connected component } C} \frac{|C|}{n} \cdot \frac{1}{|C|} = \frac{c}{n} \ .$$

And by linearity of expectation we obtain $\mathbf{E}[\hat{c}] = c$.

To show that $\hat{c}$ is concentrated around its expectation, we apply Chebyshev inequality. Since b_i is an indicator random variable, we have

$$\mathbf{Var}[b_i] = \mathbf{E}[b_i^2] - \mathbf{E}[b_i]^2 \leq \mathbf{E}[b_i^2] = \mathbf{E}[b_i] = c/n \ .$$

The b_i are mutually independent and so we have

$$\mathbf{Var}[\hat{c}] = \mathbf{Var}[\frac{n}{s} \cdot \sum_{i=1}^{s} b_i] = \frac{n^2}{s^2} \cdot \sum_{i=1}^{s} \mathbf{Var}[b_i] \leq \frac{n \cdot c}{s} \ .$$

With this bound for $\mathbf{Var}[\hat{c}]$, we can use Chebyshev inequality to obtain

$$\mathbf{Pr}[|\hat{c} - \mathbf{E}[\widehat{c}]| \geq \lambda n] \leq \frac{n \cdot c}{s \cdot \lambda^2 \cdot n^2} \leq \frac{1}{\lambda^2 \cdot s} \ .$$

From this it follows that one can approximate the number of connected components within additive error of λn in a graph with maximum degree D in $O(\frac{D \cdot \log n}{\lambda^2 \cdot \varrho})$ time and with probability $1 - \varrho$. The following somewhat stronger result has been obtained in [15]. Notice that the obtained running time is *independent of the input size n.*

Theorem 5. [15] *The number of connected components in a graph with maximum degree D can be approximated with additive error at most $\pm\lambda\, n$ in $O(\frac{D}{\lambda^2}\log(D/\lambda))$ time and with probability* $3/4$.

Now, we can use this procedure with parameters $\lambda = \varepsilon/(2W)$ and $\varrho = \frac{1}{4W}$ in algorithm ApproxMSTWeight. The probability that at least one call to ApproxConnectedComps is not within an additive error $\pm\lambda n$ is at most $1/4$. The overall additive error is at most $\pm\varepsilon n/2$. Since the cost of the minimum spanning tree is at least $n - 1 \geq n/2$, it follows that the algorithms computes in $O(D \cdot W^3 \cdot \log n/\varepsilon^2)$ time a $(1 \pm \varepsilon)$-approximation of the weight of the minimum spanning tree with probability at least $3/4$. In [15], Chazelle et al. proved a slightly stronger result which has running time *independent of the input size*.

Theorem 6. [15] *Algorithm* ApproxMSTWeight *computes a value $\widetilde{MST}$ that with probability at least* $3/4$ *satisfies*

$$(1-\varepsilon)\cdot MST \quad \leq \quad \widetilde{MST} \leq (1+\varepsilon)\cdot MST \enspace .$$

The algorithm runs in $\widetilde{O}(D \cdot W/\varepsilon^2)$ time.

The same result also holds when D is only the average degree of the graph (rather than the maximum degree) and the edge weights are reals from the interval $[1, W]$ (rather than integers) [15]. Observe that, in particular, for sparse graphs for which the ratio between the maximum and the minimum weight is constant, the algorithm from [15] *runs in constant time*!

It was also proved in [15] that any algorithm estimating MST requires $\Omega(D \cdot W/\varepsilon^2)$ time.

3.3 Other Sublinear-time Results for Graphs

In this section, our main focus was on combinatorial algorithms for sparse graphs. In particular, we did not discuss a large body of algorithms for dense graphs represented in the adjacency matrix model. Still, we mention the results of approximating the size of the maximum cut in *constant time* for dense graphs [28, 32], and the more general results about approximating all dense problems in Max-SNP in *constant time* [2, 8, 28]. Similarly, we also have to mention about the existence of a large body of property testing algorithms for graphs, which in many situations can lead to sublinear-time algorithms for graph problems. To give representative references, in addition to the excellent survey expositions [26, 30, 31, 40, 49], we want to mention the recent results on testability of graph properties, as described, e.g., in [3, 4, 5, 6, 11, 21, 33, 43].

4 Sublinear-Time Algorithms for Metric Problems

One of the most widely considered models in the area of sublinear time approximation algorithms is the *distance oracle model* for metric spaces. In this model, the input of an algorithm is a set P of n points in a metric space (P, d). We assume that it is possible to compute the distance $d(p, q)$ between any pair of points p, q in constant time. Equivalently, one could assume that the algorithm is given access to the $n \times n$ distance matrix of the metric space, i.e., we have oracle access to the matrix of a weighted undirected complete graph. Since the full description size of this matrix is $\Theta(n^2)$, we will call any algorithm with $o(n^2)$ running time a *sublinear algorithm*.

Which problems can and cannot be approximated in sublinear time in the distance oracle model? One of the most basic problems is to find (an approximation) of the shortest or the longest pairwise distance in the metric space. It turns out that the shortest distance cannot be approximated. The counterexample is a uniform metric (all distances are 1) with one distance being set to some very small value ε. Obviously, it requires $\Omega(n^2)$ time to find this single short distance. Hence, no sublinear time approximation algorithm for the shortest distance problem exists. What about the longest distance? In this case, there is a very simple $\frac{1}{2}$-approximation algorithm, which was first observed by Indyk [37]. The algorithm chooses an arbitrary point p and returns its furthest neighbor q. Let r, s be the furthest pair in the metric space. We claim that $d(p, q) \geq \frac{1}{2}\, d(r, s)$. By the triangle inequality, we have $d(r, p) + d(p, s) \geq d(r, s)$. This immediately implies that either $d(p, r) \geq \frac{1}{2}\, d(r, s)$ or $d(p, s) \geq \frac{1}{2}\, d(r, s)$. This shows the approximation guarantee.

In the following, we present some recent sublinear-time algorithms for a few optimization problems in metric spaces.

4.1 Minimum Spanning Trees

We can view a metric space as a weighted complete graph G. A natural question is whether we can find out anything about the minimum spanning tree of that graph. As already mentioned in the previous section, it is not possible to find in $o(n^2)$ time a spanning tree in the distance oracle model that approximates the minimum spanning tree within a constant factor [37]. However, it is possible to *approximate the weight* of a minimum spanning tree within a factor of $(1+\varepsilon)$ in $\widetilde{O}(n/\varepsilon^{O(1)})$ time [19].

The algorithm builds upon the ideas used to approximate the weight of the minimum spanning tree in graphs described in Section 3.2 [15]. Let us first observe that for the metric space problem we can assume that the maximum distance is $O(n/\varepsilon)$ and the shortest distance is 1. This can be achieved by first approximating the longest distance in $O(n)$ time and then scaling the problem appropriately.

Since by the triangle inequality the longest distance also provides a lower bound on the minimum spanning tree, we can round up to 1 all edge weights that are smaller than 1. Clearly, this does not significantly change the weight of the minimum spanning tree. Now we could apply the algorithm ApproxMSTWeight from Section 3.2, but this would not give us an $o(n^2)$ algorithm. The reason is that in metric case we have a complete graph, i.e., the average degree is $D = n - 1$, and the edge weights are in the interval $[1, W]$, where $W = O(n/\varepsilon)$. So, we need a different approach. In the following we will outline an idea how to achieve a randomized $o(n^2)$ algorithm. To get a near linear time algorithm as in [19] further ideas have to be applied.

The first difference to the algorithm from Section 3.2 is that when we develop a formula for the minimum spanning tree weight, we use geometric progression instead of arithmetic progression. Assuming that all edge weights are powers of $(1 + \varepsilon)$, we define $G^{(i)}$ to be the subgraph of G that contains all edges of length at most $(1 + \varepsilon)^i$. We denote by $c^{(i)}$ the number of connected components in $G^{(i)}$. Then we can write

$$MST \quad = \quad n - W + \varepsilon \cdot \sum_{i=0}^{r-1} (1 + \varepsilon)^i \cdot c^{(i)} \quad , \tag{1}$$

where $r = \log_{1+\varepsilon} W - 1$.

Once we have (1), our approach will be to approximate the number of connected components $c^{(i)}$ and use formula (1) as an estimator. Although geometric progression has the advantage that we only need to estimate the connected components in $r = O(\log n/\varepsilon)$ subgraphs, the problem is that the estimator is multiplied by $(1 + \varepsilon)^i$. Hence, if we use the procedure from Section 3.2, we would get an additive error of $\varepsilon n \cdot (1 + \varepsilon)^i$, which, in general, may be much larger than the weight of the minimum spanning tree.

The basic idea how to deal with this problem is as follows. We will use a different graph traversal than BFS. Our graph traversal runs only on a subset of the vertices, which are called *representative vertices*. Every pair of representative vertices are at distance at least $\varepsilon \cdot (1+\varepsilon)^i$ from each other. Now, assume there are m representative vertices and consider the graph induced by these vertices (there is a problem with this assumption, which will be discussed later). Running algorithm ApproxConnectedComps on this induced graph makes an error of $\pm\lambda m$, which must be multiplied by $(1 + \varepsilon)^i$ resulting in an additive error of $\pm\lambda \cdot (1 + \varepsilon)^i \cdot m$. Since the m representative vertices have pairwise distance $\varepsilon \cdot (1 + \varepsilon)^i$, we have a lower bound $MST \geq m \cdot \varepsilon \cdot (1 + \varepsilon)^i$. Choosing $\lambda = \varepsilon^2/r$ would result in a $(1 + \varepsilon)$-approximation algorithm.

Unfortunately, this simple approach does not work. One problem is that we cannot choose a random representative point. This is because we have no a priori knowledge of the set of representative points. In fact, in the algorithm the points are chosen greedily during the graph traversal. As a consequence, the decision whether a vertex is a representative vertex or not, depends on the starting point of the graph traversal. This may also mean that the number of representative vertices in a connected component also depends on the starting point of the graph traversal. However, it is still possible to cope with these problems and use the approach outlined above to get the following result.

Theorem 7. [19] *The weight of a minimum spanning tree of an n-point metric space can be approximated in $\widetilde{O}(n/\varepsilon^{O(1)})$ time to within a $(1+\varepsilon)$ factor and with confidence probability at least $\frac{3}{4}$.*

4.1.1 Extensions: Sublinear-time $(2+\varepsilon)$-approximation of metric TSP and Steiner trees

Let us remark here one direct corollary of Theorem 7. By the well known relationship (see, e.g., [51]) between minimum spanning trees, travelling salesman tours, and minimum Steiner trees, the algorithm for estimating the weight of the minimum spanning tree from Theorem 7 immediately yields $\widetilde{O}(n/\varepsilon^{O(1)})$ time $(2+\varepsilon)$-approximation algorithms for two other classical problems in metric spaces (or in graphs satisfying the triangle inequality): estimating the weight of the *travelling salesman tour* and the *minimum Steiner tree*.

4.2 Uniform Facility Location

Similarly to the minimum spanning tree problem, one can estimate the cost of the *metric uniform facility location* problem in $\widetilde{O}(n/\varepsilon^{O(1)})$ time [10]. This problem is defined as follows. We are given an n-point metric space (P,d). We want to find a subset $F \subseteq P$ of open facilities such that

$$|F| + \sum_{p \in P} d(p,F)$$

is minimized. Here, $d(p,F)$ denote the distance from p to the nearest point in F. It is known that one cannot find a solution that approximates the optimal solution within a constant factor in $o(n^2)$ time [50]. However, it is possible to approximate the *cost* of an optimal solution within a constant factor.

The main idea is as follows. Let us denote by $B(p,r)$ the set of points from P with distance at most r from p. For each $p \in P$ let r_p be the unique value that satisfies

$$\sum_{q \in B(p,r_p)} (r_p - d(p,q)) = 1 \ .$$

Then one can show that

Lemma 8. [10]

$$\frac{1}{4} \cdot Opt \ \leq \ \sum_{p \in P} r_p \ \leq \ 6 \cdot Opt \ ,$$

where Opt denotes the cost of an optimal solution to the metric uniform facility location problem.

Now, the algorithm is based on a randomized algorithm that for a given point p, estimates r_p to within a constant factor in time $O(r_p \cdot n \cdot \log n)$ (recall that $r_p \leq 1$). Thus, the smaller r_p, the faster the algorithm. Now, let p be chosen uniformly at random from P. Then the expected running time to estimate r_p is $O(n \log n \cdot \sum_{p \in P} r_p/n) = O(n \log n \cdot \mathbf{E}[r_p])$. We pick a random sample set S of $s = 100 \log n/\mathbf{E}[r_p]$ points uniformly at random from P. (The fact that we do not know $\mathbf{E}[r_p]$ can be dealt with by using a logarithmic number of guesses.) Then we use our algorithm to compute for each $p \in S$ a value $\widehat{r_p}$ that approximates r_p within a constant factor. Our algorithm outputs $\frac{n}{s} \cdot \sum_{p \in S} \widehat{r_p}$ as an estimate for the cost of the facility location problem. Using Hoeffding bounds it is easy to prove that $\frac{n}{s} \cdot \sum_{p \in S} r_p$ approximates $\sum_{p \in P} r_p = Opt$ within a constant factor and with high probability. Clearly, the same statement is true, when we replace the r_p values by their constant approximations $\widehat{r_p}$. Finally, we observe that expected running time of our algorithm will be $\widetilde{O}(n/\varepsilon^{O(1)})$. This allows us to conclude with the following.

Theorem 9. [10] *There exists an algorithm that computes a constant factor approximation to the cost of the metric uniform facility location problem in* $O(n \log^2 n)$ *time and with high probability.*

4.3 Clustering via Random Sampling

The problems of clustering large data sets into subsets (clusters) of similar characteristics are one of the most fundamental problems in computer science, operations research, and related fields. Clustering problems arise naturally in various massive datasets applications, including data mining, bioinformatics, pattern classification, etc. In this section, we will discuss the *uniformly random sampling* for clustering problems in metric spaces, as analyzed in two recent papers [20, 46].

Let us consider a classical clustering problem known as the *k-median problem*. Given a finite metric space (P, d), the goal is to find a set $C \subseteq P$ of k centers (points in P) that minimizes $\sum_{p \in P} d(p, C)$, where $d(p, C)$ denotes the distance from p to the nearest point in C. The k-median problem has been studied in numerous research papers. It is known to be $\mathcal{NP}$-hard and there exist constant-factor approximation algorithms running in $\widetilde{O}(n\,k)$ time. In two recent papers [20, 46], the authors asked the question about the quality of the uniformly random sampling approach to k-median, that is, is the quality of the following generic scheme:

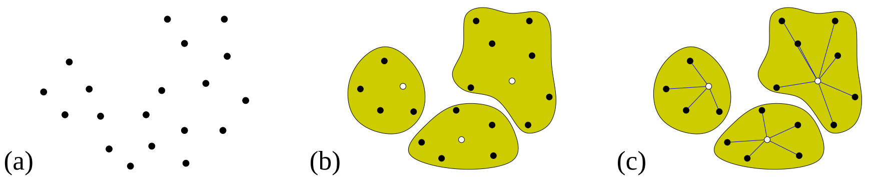


Figure 2: *(a) A set of points in a metric space, (b) its* 3*-clustering (white points correspond to the center points), and (c) the distances used in the cost for the* 3*-median.*

(1) choose a multiset $S \subseteq P$ of size s i.u.r. (with repetitions),
(2) run an α-approximation algorithm $\mathbb{A}_\alpha$ on input S to compute a solution C^*
(3) **return** set C^* (the clustering induced by the solution for the sample).

The goal is to show that already a sublinear-size sample set S will suffice to obtain a good approximation guarantee. Furthermore, as observed in [46] (see also [45]), in order to have any guarantee of the approximation, one has to consider the quality of the approximation as a function of the diameter of the metric space. Therefore, we consider a model with the diameter of the metric space Δ given, that is, with $d : P \times P \to [0, \Delta]$.

Using techniques from statistics and computational learning theory, Mishra et al. [46] proved that if we sample a set S of $s = \widetilde{O}\left(\left(\frac{\alpha\Delta}{\varepsilon}\right)^2 (k \ln n + \ln(1/\delta))\right)$ points from P i.u.r. (*independently and uniformly at random*) and run α-approximation algorithm $\mathbb{A}_\alpha$ to find an approximation of the k-median for S, then with probability at least $1 - \delta$, the output set of k centers has *average distance* to the nearest center of at most $2 \cdot \alpha \cdot \overline{\mathsf{med}}(P, k) + \varepsilon$, where $\overline{\mathsf{med}}(P, k)$ denotes the *average distance* to the k-median C, that is, $\overline{\mathsf{med}}(P, k) = \frac{\sum_{v \in P} d(v,C)}{n}$. We will now briefly sketch the analysis due to Czumaj and Sohler [20] of a similar approximation guarantee but with a smaller bound for s.

Let C_{opt} denote an optimal set of centers for P and let $\overline{\mathsf{cost}}(X, C)$ be the average cost of the clustering of set X with center set C, that is, $\overline{\mathsf{cost}}(X, C) = \frac{\sum_{x \in X} d(x,C)}{|X|}$. Notice that $\overline{\mathsf{cost}}(P, C_{opt}) = \overline{\mathsf{med}}(P, k)$. The analysis of Czumaj and Sohler [20] is performed in two steps.

(i) We first show that there is a set of k centers $C \subseteq S$ such that $\overline{\mathsf{cost}}(S, C)$ is a good approximation of $\overline{\mathsf{med}}(P, k)$ with high probability.

(ii) Next we show that with high probability, every solution C for P with cost much bigger than $\overline{\mathsf{med}}(P, k)$ is either not a feasible solution for S (i.e., $C \not\subseteq S$) or $\overline{\mathsf{cost}}(S, C) \gg \alpha \cdot \overline{\mathsf{med}}(P, k)$ (that is, the cost of C for the sample set S is large with high probability).

Since S contains a solution with cost at most $c \cdot \overline{\mathsf{med}}(P, k)$ for some small c, $\mathbb{A}_\alpha$ will compute a solution C^* with cost at most $\alpha \cdot c \cdot \overline{\mathsf{med}}(P, k)$. Now we have to prove that no solution C for P with cost much bigger than $\overline{\mathsf{med}}(P, k)$ will be returned, or in other words, that if C is feasible for S then its cost is larger than $\alpha \cdot c \cdot \overline{\mathsf{med}}(P, k)$. But this is implied by (ii). Therefore, the algorithm will not return a solution with too large cost, and the sampling is a $(c \cdot \alpha)$-approximation algorithm.

Theorem 10. **[20]** *Let* $0 < \delta < 1$, $\alpha \geq 1$, $0 < \beta \leq 1$ *and* $\varepsilon > 0$ *be approximation parameters. If* $s \geq \frac{c \cdot \alpha}{\beta} \cdot \left(k + \frac{\Delta}{\varepsilon \cdot \beta} \cdot \left(\alpha \cdot \ln(1/\delta) + k \cdot \ln\left(\frac{k \Delta \alpha}{\varepsilon \beta^2}\right)\right)\right)$ *for an appropriate constant c, then for the solution set of centers* C^**, with probability at least* $1 - \delta$ *it holds the following*

$$\overline{cost}(V, C^*) \;\leq\; 2\,(\alpha + \beta) \cdot \overline{\mathsf{med}}(P, k) + \varepsilon \;\; .$$

To give the flavor of the analysis, we will sketch (a simpler) part (i) of the analysis:

Lemma 11. *If* $s \geq \frac{3\Delta\alpha(1+\alpha/\beta)\ln(1/\delta)}{\beta \cdot \overline{\mathsf{med}}(P,k)}$ *then* $\mathbf{Pr}[\overline{cost}(S, C^*) \leq 2\,(\alpha + \beta) \cdot \overline{\mathsf{med}}(P, k)] \geq 1 - \delta$.

Proof. We first show that if we consider the clustering of S with the optimal set of centers C_{opt} for P, then $\overline{\mathrm{cost}}(S, C_{opt})$ is a good approximation of $\overline{\mathsf{med}}(P, k)$. The problem with this bound is that in general, we cannot expect C_{opt} to be contained in the sample set S. Therefore, we have to show also that the optimal set of centers for S cannot have cost much worse than $\overline{\mathrm{cost}}(S, C_{opt})$.

Let X_i be the random variable for the distance of the ith point in S to the nearest center of C_{opt}. Then, $\overline{\mathrm{cost}}(S, C_{opt}) = \frac{1}{s} \sum_{1 \leq i \leq s} X_i$, and, since $\mathbf{E}[X_i] = \overline{\mathsf{med}}(P, k)$, we also have $\overline{\mathsf{med}}(P, k) = \frac{1}{s} \cdot \mathbf{E}[\sum X_i]$. Hence,

$$\mathbf{Pr}[\overline{\mathrm{cost}}(S, C_{opt}) > (1 + \tfrac{\beta}{\alpha}) \cdot \overline{\mathsf{med}}(P, k)] = \mathbf{Pr}[\sum_{1 \leq i \leq s} X_i > (1 + \tfrac{\beta}{\alpha}) \cdot \mathbf{E}[\sum_{1 \leq i \leq s} X_i]] \;\; .$$

Observe that each X_i satisfies $0 \leq X_i \leq \Delta$. Therefore, by Chernoff-Hoeffding bound we obtain:

$$\mathbf{Pr}[\sum_{1 \leq i \leq s} X_i > (1 + \beta/\alpha) \cdot \mathbf{E}[\sum_{1 \leq i \leq s} X_i]] \;\leq\; e^{-\frac{s \cdot \overline{\mathsf{med}}(P,k) \cdot \min\{(\beta/\alpha),(\beta/\alpha)^2\}}{3\Delta}} \;\leq\; \delta \;\; . \qquad (2)$$

This gives us a good bound for the cost of $\overline{\mathrm{cost}}(S, C_{opt})$ and now our goal is to get a similar bound for the cost of the optimal set of centers for S. Let C be the set of k centers in S obtained by replacing each $c \in C_{opt}$ by its nearest neighbor in S. By the triangle inequality, $\overline{\mathrm{cost}}(S, C) \leq 2 \cdot \overline{\mathrm{cost}}(S, C_{opt})$. Hence, multiset S contains a set of k centers whose cost is at most $2 \cdot (1 + \beta/\alpha) \cdot \overline{\mathsf{med}}(P, k)$ with probability at least $1 - \delta$. Therefore, the lemma follows because $\mathbb{A}_\alpha$ returns an α-approximation C^* of the k-median for S. $\square$

Next, we only state the other lemma that describe part (ii) of the analysis of Theorem 10.

Lemma 12. *Let* $s \geq \frac{c \cdot \alpha}{\beta} \cdot \left(k + \frac{\Delta}{\varepsilon \cdot \beta} \cdot \left(\alpha \cdot \ln(1/\delta) + k \cdot \ln\left(\frac{k \Delta \alpha}{\varepsilon \beta^2}\right)\right)\right)$ *for an appropriate constant c. Let* $\mathbb{C}$ *be the set of all sets of k centers C of P with* $\overline{cost}(P, C) > (2\alpha + 6\beta) \cdot \overline{\mathsf{med}}(P, k)$. *Then,*

$$\mathbf{Pr}[\exists C_b \in \mathbb{C} : C_b \subseteq S \ \textit{and}\ \overline{cost}(S, C_b) \leq 2(\alpha + \beta)\overline{\mathsf{med}}(P, k)] \leq \delta \ . \qquad \square$$

Observe that comparing the result from [46] to the result in Theorem 10, Theorem 10 improves the sample complexity by a factor of $\Delta \cdot \log n / \varepsilon$ while obtaining a slightly worse approximation ratio of $2(\alpha + \beta)\overline{\mathsf{med}}(P, k) + \varepsilon$, instead of $2\alpha\overline{\mathsf{med}}(P, k) + \varepsilon$ as in [46]. However, since the polynomial-time algorithm with the best known approximation guarantee has $\alpha = 3 + \frac{1}{c}$ for the running time of $O(n^c)$ time [9], this significantly improves the running time of [46] for all realistic choices of the input parameters while achieving the same approximation guarantee. As a highlight, Theorem 10 yields a sublinear-time algorithm that in time $\widetilde{O}((\frac{\Delta}{\varepsilon} \cdot (k + \log(1/\delta)))^2)$ — *fully independent of n* — returns a set of k centers for which the average distance to the nearest median is at most $O(\overline{\mathsf{med}}(P, k)) + \varepsilon$ with probability at least $1 - \delta$.

Extensions. The result in Theorem 10 can be significantly improved if we assume the input points are in *Euclidean space* $\mathbb{R}^d$. In this case the approximation guarantee can be improved to $(\alpha + \beta)\overline{\mathsf{med}}(P, k) + \varepsilon$ at the cost of increasing the sample size to $\widetilde{O}(\frac{\Delta \cdot \alpha}{\varepsilon \cdot \beta^2} \cdot (k\,d + \log(1/\delta)))$.

Furthermore, a similar approach as that sketched above can be applied to study similar generic sample schemes for other clustering problems. As it is shown in [20], almost identical analysis lead to sublinear (independent on n) sample complexity for the classical *k-means problem*. Also, a more complex analysis can be applied to study the sample complexity for the *min-sum k-clustering problem* [20].

4.4 Other Results

Indyk [37] was the first who observed that some optimization problems in metric spaces can be solved in sublinear-time, that is, in $o(n^2)$ time. He presented $(\frac{1}{2} - \varepsilon)$-approximation algorithms for MaxTSP and the maximum spanning tree problems that run in $O(n/\varepsilon)$ time [37]. He also gave a $(2 + \varepsilon)$-approximation algorithm for the minimum routing cost spanning tree problem and a $(1 + \varepsilon)$ approximation algorithm for the average distance problem; both algorithms run in $O(n/\varepsilon^{O(1)})$ time.

There is also a number of sublinear-time algorithms for various clustering problems in either Euclidean spaces or metric spaces, when the number of clusters is small. For radius (*k-center*) and *diameter clustering* in Euclidean spaces, sublinear-time property testing algorithms [1, 21] and tolerant testing algorithms [48] have been developed. The first sublinear algorithm for the *k-median* problem was a bicriteria approximation algorithm [37]. This algorithm computes in $\widetilde{O}(n\,k)$ time a set of $O(k)$ centers that are a constant factor approximation to the k-median objective function. Later, standard constant factor approximation algorithms were given that run in time $\widetilde{O}(n\,k)$ (see, e.g., [44, 50]). These sublinear-time results have been extended in many different ways, e.g., to efficient data streaming algorithms and very fast algorithms for Euclidean k-median and also to *k-means*, see, e.g., [9, 12, 16, 27, 35, 36, 41, 42, 45]. For another clustering problem, the *min-sum k-clustering problem* (which is complement to the Max-k-Cut), for the basic case of $k = 2$, Indyk [39] (see also [38]) gave a $(1+\epsilon)$-approximation algorithm that runs in time $O(2^{1/\epsilon^{O(1)}}\, n\,(\log n)^{O(1)})$, which is sublinear in the full input description size. No such results are known for $k \geq 3$, but recently, [22] gave a constant-factor approximation algorithm for min-sum k-clustering that runs in time $O(n\,k\,(k\,\log n)^{O(k)})$ and a polylogarithmic approximation algorithm running in time $\widetilde{O}(n\,k^{O(1)})$.

4.5 Limitations: What Cannot be done in Sublinear-Time

The algorithms discussed in the previous sections may suggest that many optimization problems in metric spaces have sublinear-time algorithms. However, it turns out that the problems listed in the previous sections are more like exceptions than a norm. Indeed, most of the problems have a trivial lower bound that exclude sublinear-time algorithms. We have already mentioned in Section 4 that the problem of approximating the cost of the lightest edge in a finite metric space (P, d) requires $\Omega(n^2)$, even if randomization is allowed. The other problems for which no sublinear-time algorithms are possible include estimation of the cost of minimum-cost matching, the cost of minimum-cost bi-chromatic matching, the cost of minimum *non-uniform* facility location, the cost of k-median for $k = n/2$; all these problems require $\Omega(n^2)$ (randomized) time to estimate the cost of their optimal solution to within any constant factor [10].

To illustrate the lower bounds, we give two instances of the metric spaces which are indistinguishable by any $o(n^2)$-time algorithm for which the cost of the minimum-cost matching in one instance is greater than λ times the one in the other instance (see Figure 3). Consider a metric space (P, d) with $2n$ points, n points in L and n points in R. Take a random perfect matching $\mathbb{M}$ between the points in L

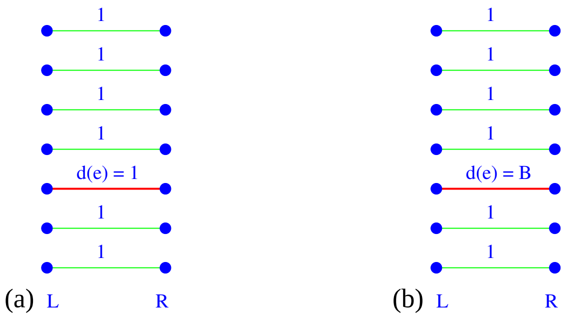


Figure 3: *Two instance of the metric matching which are indistinguishable in* $o(n^2)$ *time and whose cost differ by a factor greater than* λ. *The perfect matching connecting L with R is selected at random and the edge e is selected as a random edge from the matching. We set* $B = n(\lambda - 1) + 2$. *The distances not shown are all equal to* $n^3\lambda$.

and R, and then choose an edge $e \in \mathbb{M}$ at random. Next, define the distance in (P, d) as follows:

- $d(e)$ is either 1 or B, where we set $B = n(\lambda - 1) + 2$,
- for any $e^* \mathbb{M} \setminus \{e\}$ set $d(e^*) = 1$, and
- for any other pair of points $p, q \in P$ not connected by an edge from $\mathbb{M}$, $d(p, q) = n^3\lambda$.

It is easy to see that both instances define properly a metric space (P, d). For such problem instances, the cost of the minimum-cost matching problem will depend on the choice of $d(e)$: if $d(e) = B$ then the cost will be $n - 1 + B > n\lambda$, and if $d(e) = 1$, then the cost will be n. Hence, any λ-factor approximation algorithm for the matching problem must distinguish between these two problem instances. However, this requires to find if there is an edge of length B, and this is known to require time $\Omega(n^2)$, even if a randomized algorithm is used.

5 Conclusions

It would be impossible to present a complete picture of the large body of research known in the area of sublinear-time algorithms in such a short paper. In this survey, our main goal was to give some flavor of the area and of the types of the results achieved and the techniques used. For more details, we refer to the original works listed in the references.

We did not discuss two important areas that are closely related to sublinear-time algorithms: property testing and data streaming algorithms. For interested readers, we recommend the surveys in [7, 26, 30, 31, 40, 49] and [47], respectively.

References

[1] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. *SIAM Journal on Discrete Mathematics*, 16(3): 393–417, 2003.

[2] N. Alon, W. Fernandez de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of MAX-CSPs. *Journal of Computer and System Sciences*, 67(2): 212–243, 2003.

[3] N. Alon, E. Fischer, M. Krivelevich, M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4): 451–476, 2000.

[4] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, 2006.

[5] N. Alon and A. Shapira. Every monotone graph property is testable. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 128–137, 2005.

[6] N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 429–438, 2005.

[7] N. Alon and A. Shapira. Homomorphisms in graph property testing - A survey. *Electronic Colloquium on Computational Complexity (ECCC)*, Report No. 85, 2005.

[8] S. Arora, D. R. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of $\mathcal{NP}$-hard problems. Journal of Computer and System Sciences, 58(1): 193–210, 1999.

[9] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 21–30, 2001.

[10] M. Bădoiu, A. Czumaj, P. Indyk, and C. Sohler. Facility location in sublinear time. *Proceedings of the 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 866-877, 2005.

[11] C. Borgs, J. Chayes, L. Lovász, V. T. Sos, B. Szegedy, and K. Vesztergombi. Graph limits and parameter testing. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, 2006.

[12] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 30–39, 2003.

[13] B. Chazelle and D. P. Dobkin. Intersection of convex objects in two and three dimensions. *Journal of the ACM*, 34(1): 1–27, 1987.

[14] B. Chazelle, D. Liu, and A. Magen. Sublinear geometric algorithms. *SIAM Journal on Computing*, 35(3): 627–646, 2006.

[15] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing*, 34(6): 1370–1379, 2005.

[16] K. Chen. On *k*-median clustering in high dimensions. *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1177–1185, 2006.

[17] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler. Sublinear-time approximation of Euclidean minimum spanning tree. *SIAM Journal on Computing*, 35(1): 91–109, 2005.

[18] A. Czumaj and C. Sohler. Property testing with geometric queries. *Proceedings of the 9th Annual European Symposium on Algorithms (ESA)*, pp. 266–277, 2001.

[19] A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 175–183, 2004.

[20] A. Czumaj and C. Sohler. Sublinear-time approximation for clustering via random sampling. *Proceedings of the 31st Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 396–407, 2004.

[21] A. Czumaj and C. Sohler. Abstract combinatorial programs and efficient property testers. *SIAM Journal on Computing*, 34(3): 580–615, 2005.

[22] A. Czumaj and C. Sohler. Small space representations for metric min-sum *k*-clustering and their applications. Manuscript, 2006.

[23] A. Czumaj, C. Sohler, and M. Ziegler. Property testing in computational geometry. *Proceedings of the 8th Annual European Symposium on Algorithms (ESA)*, pp. 155–166, 2000.

[24] M. Dyer, N. Megiddo, and E. Welzl. Linear programming. In *Handbook of Discrete and Computational Geometry*, 2nd edition, edited by J. E. Goodman and J. O'Rourke, CRC Press, 2004, pp. 999–1014.

[25] U. Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4): 964–984, 2006.

[26] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the EATCS*, 75: 97–126, October 2001.

[27] G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 209–217, 2005.

[28] A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2): 175–220, 1999.

[29] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6): 1025–1041, 2004.

[30] O. Goldreich. Combinatorial property testing (a survey). In P. Pardalos, S. Rajasekaran, and J. Rolim, editors, *Proceedings of the DIMACS Workshop on Randomization Methods in Algorithm Design*, volume 43 of *DIMACS, Series in Discrete Mathetaics and Theoretical Computer Science*, pp. 45–59, 1997. American Mathematical Society, Providence, RI, 1999.

[31] O. Goldreich. Property testing in massive graphs. In J. Abello, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of massive data sets*, pp. 123–147. Kluwer Academic Publishers, 2002.

[32] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4): 653–750, 1998.

[33] O. Goldreich and D. Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[34] O. Goldreich and D. Ron. Approximating average parameters of graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, Report No. 73, 2005.

[35] S. Har-Peled and S. Mazumdar. Coresets for *k*-means and *k*-medians and their applications. *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 291–300, 2004.

[36] S. Har-Peled and A. Kushal. Smaller coresets for *k*-median and *k*-means clustering. *Proceedings of the 21st Annual ACM Symposium on Computational Geometry*, pp. 126–134, 2005.

[37] P. Indyk. Sublinear time algorithms for metric space problems. *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 428–434, 1999.

[38] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 154–159, 1999.

[39] P. Indyk. *High-Dimensional Computational Geometry*. PhD thesis, Stanford University, 2000.

[40] R. Kumar and R. Rubinfeld. Sublinear time algorithms. *SIGACT News*, 34: 57–67, 2003.

[41] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1 + \varepsilon)$-approximation algorithm for *k*-means clustering in any dimensions. *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 454–462, 2004.

[42] A. Kumar, Y. Sabharwal, and S. Sen. Linear time algorithms for clustering problems in any dimensions. *Proceedings of the 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 1374–1385, 2005.

[43] L. Lovász and B. Szegedy. Graph limits and testing hereditary graph properties. Technical Report, MSR-TR-2005-110, Microsoft Research, August 2005.

[44] R. Mettu and G. Plaxton. Optimal time bounds for approximate clustering. *Machine Learning*, 56(1-3):35–60, 2004.

[45] A. Meyerson, L. O'Callaghan, and S. Plotkin. A *k*-median algorithm with running time independent of data size. *Machine Learning,* 56(1–3): 61–87, July 2004.

[46] N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA),* pp. 439–447, 2001.

[47] S. Muthukrishnan. Data streams: Algorithms and applications. In *Foundations and Trends in Theoretical Computer Science,* volume 1, issue 2, August 2005.

[48] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *Electronic Colloquium on Computational Complexity (ECCC),* Report No. 10, 2004.

[49] D. Ron. Property testing. In P. M. Pardalos, S. Rajasekaran, J. Reif, and J. D. P. Rolim, editors, *Handobook of Randomized Algorithms,* volume II, pp. 597–649. Kluwer Academic Publishers, 2001.

[50] M. Thorup. Quick *k*-median, *k*-center, and facility location for sparse graphs. *SIAM Journal on Computing,* 34(2):405–432, 2005.

[51] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, New York, 2004.

The Computational Complexity Column

by

Jacobo Torán

Dept. Theoretische Informatik, Universität Ulm
Oberer Eselsberg, 89069 Ulm, Germany

`jacobo.toran@uni-ulm.de`

`http://theorie.informatik.uni-ulm.de/Personen/jt.html`

In the last years significant improvements have been obtained in the area of learning in the PAC model under the uniform distribution. The Fourier transform has been a common ingredient in many of these results. Johannes Köbler and Wolfgang Lindner present here a very interesting survey on this important but maybe not so well known area of Complexity.

Learning Boolean Functions under the uniform distribution via the Fourier Transform

Johannes Köbler [*] Wolfgang Lindner [†]

Abstract

In this article we give a brief overview on some important learning results using the Fourier Transform and highlight some open questions.

---

[*]Institut für Informatik, Humboldt-Universität zu Berlin, D-10099 Berlin, Germany. Email: `koebler@informatik.hu-berlin.de`

[†]Abt. Theoretische Informatik, Universität Ulm, Oberer Eselsberg, D-89069 Ulm, Germany. Email: `lindner@informatik.uni-ulm.de`

1 Introduction

Learning via the Fourier transform is a basic tool when learning in the PAC model under the uniform distribution. It has been successfully applied to various natural concept classes ranging from decision trees to constant depth circuits. The most remarkable example is Jackson's Harmonic Sieve algorithm [19] for learning DNF formulas in polynomial time with membership queries. Learning via the Fourier transform has thus provided a successful attack on the notoriously open problem of learning DNF formulas in the distribution-free PAC model without membership queries. The fastest known algorithm for this problem runs in time $2^{\tilde{O}(n^{1/3})}$ [27].

The Fourier spectrum of Boolean functions has been first applied in theoretical computer science by Kahn, Kalai and Linial [22] to answer a question posed by Ben-David and Linial [3] concerning the sensitivity of Boolean functions. The first application in computational learning is due to Linial, Mansour and Nisan [33]. The Fourier transform of a Boolean function f can be regarded as a representation of f as a linear combination over the basis of all parity functions. Each coefficient is given by the correlation between f and the corresponding basis function. Learning can then be achieved through estimating the Fourier coefficients based on a sufficiently large sample of f. For simple concept classes it is often possible to establish a certain property of the concepts in terms of their Fourier transform, which implies that each concept in the class can be approximated by paying attention to only a small part of its Fourier spectrum. The learning problem is then reduced to estimating the Fourier coefficients in the important part of the spectrum.

In this column we concentrate on learning functions with properties which can be expressed in terms of their Fourier spectrum. In a first part we review some basic algorithms, starting with the ubiquitous low-degree algorithm of Linial et al. [33]. Then we present the KM-algorithm of Kushilevitz and Mansour [30] for finding all significant Fourier coefficients. Next we describe Jackson's Harmonic Sieve algorithm which combines the KM-algorithm with boosting, and finally we present the more recent algorithm of Jackson et al. [20] which can be regarded as a simplified Harmonic Sieve obtained by replacing the KM-algorithm by an exhaustive search.

In a second part we concentrate on learning classes of monotone functions based on sensitivity arguments, including Bshouty and Tamon's work on monotone functions [9], Servedio's learnability result for monotone DNF [40], and the very recent learning algorithm of O'Donnell and Servedio for monotone decision trees [39] based on a sensitivity result due to Friedgut [14].

We do not make any attempt to be comprehensive. However, we do hope to convince the reader that learning via the Fourier transform is a true success story with no end in sight.

2 Notation and basic facts

In this section we fix the notation and give formal definitions for some of the concepts used in this paper. Further we state some basic facts for further reference.

Fourier Transform

We are interested in learning Boolean concept classes, i.e., each concept can be represented as a Boolean function $f\colon \{0,1\}^n \to \{\mathit{true}, \mathit{false}\}$. We denote the class of all Boolean functions of arity n by B_n. If we identify *true* with 1 and *false* with 0, then B_n forms a vector space of dimension 2^n over the field $\mathbb{F}_2$. The regular basis for B_n consists of the 2^n functions $\mathit{term}_a(x)$, $a \in \{0,1\}^n$, mapping x to 1, if $x = a$, and to 0 otherwise. Clearly, any Boolean function $f \in B_n$ has a unique representation $f = \sum_{a\in\{0,1\}^n} c_a \mathit{term}_a$ with coefficients $c_a = f(a)$. For many applications it is important to have a basis with the following useful properties.

(a) "Simple" functions should have small representations. E.g., if the value of f is already determined by a small subset of its variables, then many coefficients should vanish.

(b) Transforming f to a "similar" function f' (e.g., $f'(x) = f(x \oplus a)$) should allow for an easy conversion of the coefficient representations of the two functions.

By embedding B_n into a richer structure, namely the vector space $\mathbb{R}^{\{0,1\}^n} = \{f\colon \{0,1\}^n \to \mathbb{R}\}$ over $\mathbb{R}$, these properties can be easily achieved. In fact, if we require from our basis $\{\chi_a \mid a \in \{0,1\}^n\}$ that it contains the functions $\chi_{e_i}(x) = (-1)^{x_i}$ (in order to fulfill property (a)) and that all basis functions χ_a have the property

$$\chi_a(x \oplus b) = \chi_a(x)\chi_a(b), \tag{1}$$

then also property (b) is fulfilled. This is easy to see, since for any function f with the representation $f(x) = \sum_{a\in\{0,1\}^n} c_a\chi_a(x)$ it follows that the function $f_{\oplus b}(x) = f(x \oplus b)$ has the representation

$$f_{\oplus b}(x) = f(x \oplus b) = \sum_{a\in\{0,1\}^n} c_a\chi_a(x \oplus b) = \sum_{a\in\{0,1\}^n} c_a\chi_a(x)\chi_a(b), \tag{2}$$

implying that the coefficients c'_a of $f_{\oplus b}$ can be written as $c'_a = \chi_a(b)c_a$. Property (1) requires that the base functions are homomorphisms from the Abelian group $(\{0,1\}^n, \oplus)$ to the multiplicative group $(\mathbb{R}^*, \cdot)$ of non-zero real numbers. It is easy to show that exactly the *parity functions*

$$\chi_a(x) = (-1)^{\sum_{i=1}^n a_i x_i},\ a \in \{0,1\}^n$$

have this property and that these functions indeed form a basis for the vector space $\mathbb{R}^{\{0,1\}^n}$. By identifying the vector $a \in \{0,1\}^n$ with the set $S = \{i \in [n] \mid a_i = 1\}$, where we use $[n]$ to denote the set $\{1, \ldots, n\}$, we get the more convenient representation

$$\chi_S(x) = (-1)^{\sum_{i \in S} x_i}, \; S \subseteq [n].$$

More generally it can be shown that for any finite Abelian group G the class of all homomorphisms from G to the multiplicative group $(\mathbb{C}^*, \cdot)$ of non-zero complex numbers form a group F (under multiplication, with the constant 1 function as neutral element) that is isomorphic to G. Moreover, the functions in F form an orthogonal basis for the vector space $\mathbb{C}^G$ over $\mathbb{C}$ of all functions $f\colon G \to \mathbb{C}$. The elements of F are called the *characters* of G and F is called the *Fourier basis* for $\mathbb{C}^G$.

In the case $G = (\{0,1\}^n, \oplus)$ all functions χ_S in the Fourier basis are real-valued and hence also form a basis for the subspace $\mathbb{R}^{\{0,1\}^n}$ of all functions $f\colon \{0,1\}^n \to \mathbb{R}$. In fact, using the natural notion of *inner-product*

$$\langle f, g \rangle = 2^{-n} \sum_{x \in \{0,1\}^n} f(x)g(x) = E[f(x)g(x)],$$

it is easy to verify that the Fourier basis $F = \{\chi_S \mid S \subseteq [n]\}$ forms an orthonormal system of functions, i.e.

$$\langle \chi_S, \chi_T \rangle = E[\chi_S(x)\chi_T(x)] = E[\chi_{S \Delta T}(x)] = \begin{cases} 1, & S = T, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

implying that all parity functions have unit norm $\|\chi_S\| = 1$, where the *norm* of a function f is induced by the inner-product using the rule

$$\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{2^{-n} \sum_x f(x)^2}.$$

More generally, for $p > 0$ the *p-norm* is defined as

$$\|f\|_p = E[|f(x)|^p]^{1/p} = (2^{-n} \sum_x |f(x)|^p)^{1/p}$$

and the ∞-*norm* is

$$\|f\|_\infty = \max_x |f(x)|.$$

Notice that the norm induced by the inner-product coincides with the 2-norm. We remark for further use that for $0 < p \leq q \leq \infty$, $\|f\|_p \leq \|f\|_q$. Since the Fourier basis is orthonormal, the *Fourier coefficients* $\hat{f}(S)$ in the *Fourier expansion*

$$f = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S$$

of f can be written as the inner-product $\hat{f}(S) = \langle f, \chi_S \rangle$ of f and the basis functions χ_S. The function $\hat{f}\colon 2^{[n]} \mapsto \mathbb{R}$ mapping each *frequency* $S \subseteq [n]$ to the corresponding Fourier coefficient $\hat{f}(S)$ is called the *Fourier transform* of f. A crucial property of $\hat{f}$ is that if f does not depend on a variable x_j, then all coefficients $\hat{f}(S)$ with $j \in S$ vanish (cf. property (a)). Using (3) it follows for any functions $f, g\colon \{0,1\}^n \to \mathbb{R}$ that

$$\begin{aligned} E[fg] &= E[(\textstyle\sum_S \hat{f}(S)\chi_S)(\sum_T \hat{g}(T)\chi_T)] = \sum_{S,T} \hat{f}(S)\hat{g}(T)E[\chi_S\chi_T] \\ &= \textstyle\sum_S \hat{f}(S)\hat{g}(S). \end{aligned}$$

Hence, a further consequence of the orthonormality of the Fourier basis is *Parseval's identity* stating that

$$\|f\|^2 = \sum_S \hat{f}(S)^2.$$

If we identify *true* with -1 and *false* with 1, i.e., if we let $B_n = \{f\colon \{0,1\}^n \to \{-1,1\}\}$, then Parseval's identity implies that for a Boolean function $f \in B_n$, the squares $\hat{f}(S)^2$ of the Fourier coefficients of f sum up to 1 and thus induce a probability distribution on the frequencies. For a collection $G \subseteq 2^{[n]}$ of frequencies we refer to the "probability" $\sum_{S \in G} \hat{f}(S)^2$ of G as the *weight of the Fourier spectrum of f on G* or simply as *f's Fourier weight on G*.

Clearly, by the linearity of the vector space $\mathbb{R}^{\{0,1\}^n}$, the Fourier transform of $f + g$ is obtained as the sum $\widehat{f+g} = \hat{f} + \hat{g}$ of the Fourier transforms of f and g. Hence, if g is obtained from a function f by removing all coefficients outside G, i.e., $g = \sum_{S \in G} \hat{f}(S)\chi_S$, then Parseval's identity implies

$$\|f - g\|^2 = \sum_S \|\widehat{f-g}(S)\|^2 = \sum_S \|\hat{f}(S) - \hat{g}(S)\|^2 = \sum_{S \notin G} \hat{f}(S)^2.$$

Notice that g need not be Boolean. But it is easy to see that we can approximate any real-valued function g by the Boolean function

$$\mathrm{sgn}(g(x)) = \begin{cases} 1, & g(x) \geq 0 \\ -1, & \text{otherwise,} \end{cases}$$

where

$$\Pr[f(x) \neq \mathrm{sgn}(g(x))] \leq \|f - g\|^2.$$

We close this subsection with a useful inequality due to Beckner and Bonami [2, 6]. For any real $\delta \in [0,1]$ let T_δ be the linear operator mapping a function f to the function $T_\delta(f) = \sum_S \delta^{|S|}\hat{f}(S)\chi_S$, i.e., T_δ in some sense reduces the Fourier weight of the high frequencies (where high means that $|S|$ is large). The Beckner-Bonami inequality states that $\|T_\delta(f)\| \leq \|f\|_{1+\delta^2}$. Since for any function f taking only values in the range $\{-1, 0, 1\}$ we have $||f||_2^2 = ||f||_p^p$ for all $p > 0$, the Beckner-Bonami inequality implies for such functions that

$$||T_\delta(f)||^2 \leq ||f||_{1+\delta^2}^2 = (||f||_{1+\delta^2}^{1+\delta^2})^{2/(1+\delta^2)} = (||f||^2)^{2/(1+\delta^2)}. \tag{4}$$

For further information and background on the Fourier transform of Boolean functions we refer to [12, 42, 38, 10, 32].

Learning

We consider the well-known distribution-specific variant of Valiant's Probably Approximate Correct (PAC) learning model [43]. Let f be a Boolean function and D be a distribution on the instance space $\{0, 1\}^n$. For $\varepsilon > 0$ we say that a Boolean function h is an (ε, D)-*approximation* of f if $\Pr_D[h(x) \neq f(x)] \leq \varepsilon$, where x is chosen according to D. We use $EX(f, D)$ to denote an oracle which, when invoked, returns a random *labeled example* $(x, f(x))$ where x is chosen according to D. A *concept class* $C \subseteq B_n$ is *learnable with respect to* D if there is a randomized *learning algorithm* A which, for all *targets* $f \in C$ and parameters $\varepsilon, \delta > 0$, when given inputs ε, δ and oracle access to $EX(f, D)$, $A(\varepsilon, \delta, EX(f, D))$ outputs with probability $1 - \delta$ a Boolean function h which is an (ε, D)-approximation of f. Here, the probability is taken over the random choices of A and the random labeled examples returned by $EX(f, D)$, where we assume that the random examples are selected independently from each other and from the random choices of A.

We also consider *weak learnability*, where the hypothesis h produced by the learning algorithm A is only slightly more accurate than random guessing. For $\gamma > 0$ we say that a Boolean function h is a *weak* (γ, D)-*approximation* of f if $\Pr_D[h(x) \neq f(x)] \leq 1/2 - \gamma$. Now a concept class C is *weakly learnable with advantage* γ *and with respect to* D if there is a learning algorithm A such that for all $f \in C$ and parameters $\gamma, \delta > 0$, $A(\gamma, \delta, EX(f, D))$ outputs with probability $1 - \delta$ a weak (γ, D)-approximation of f.

If the learning algorithm A requires direct access to the oracle f rather than $EX(f, D)$, then we say that C is learnable *with membership queries*. If D is the uniform distribution, we generally omit the reference to D. For more background on learning we refer to [24].

Prominent examples of natural concept classes are *decision trees* with a single variable at each inner node, *DNF formulas*, and *constant depth* AC^0 *circuits*. Note that every decision tree with m nodes can be transformed into a DNF formula with at most m terms, and that every m-term DNF formula can be regarded as a constant depth AC^0 circuit of size $m + 1$ and depth 2. For background on circuit complexity we refer the reader to a standard textbook like [46]. For a discussion of the Fourier transform of these concept classes we refer to the excellent overview of Mansour [35]. A more recent overview is provided by Jackson and Tamon's tutorial [21].

Probability Theory

For later use we state the following result to which we refer to as the Chernoff-Hoeffding bound. Let $X_1, \dots, X_m$ be independent random variables taking values in the real interval $[a, b]$ and having expectation $E[X_i] = \mu$. Then for any $\lambda > 0$, with probability at most $2e^{-2\lambda^2 m/(b-a)^2}$, the additive error of the estimate for μ obtained by taking the arithmetic mean of m observations of the random variables $X_1, \dots X_m$ is greater than or equal to λ,

$$\Pr\left[\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \lambda\right] \leq 2e^{-2\lambda^2 m/(b-a)^2}.$$

In other words, with confidence $1 - 2e^{-2\lambda^2 m/(b-a)^2}$, the arithmetic mean provides a λ-accurate estimate for μ.

Further Notation

We measure the closeness of two real-valued functions f and g in terms of the 2-norm squared of the difference of f and g. That is, we say that f and g are *ε-close* if $\|f - g\|^2 \leq \varepsilon$. Note that for Boolean functions f and g it holds that $\|f - g\|^2 = 4\Pr[f(x) \neq g(x)]$ and hence, f and g are ε-close if and only if f is an $(\varepsilon/4)$-approximation of g.

3 Basic algorithms

In this section we review some basic algorithms for learning via the Fourier transform. We start with the low-degree algorithm of Linial et al. [33] and its application to AC^0 circuits. Next we present the KM-algorithm of Kushilevitz and Mansour [30] which can be used to learn decision trees in polynomial time with membership queries and its application to DNF formulas due to Mansour [36]. Then we describe Jackson's Harmonic Sieve [20] which learns DNF formulas in polynomial time with membership queries. Finally we brievly review the simple exhaustive search algorithm of Jackson, Klivans and Servedio [20] which can be applied to majorities over AC^0 circuits.

3.1 The low-degree algorithm

The first application of Fourier analysis in computational learning theory is due to Linial et al. [33]. The learning result is based on an upper bound for the 2-

norm of $\hat{f}$ restricted to high frequencies, where the bound depends on the circuit complexity of f. So suppose that f is a Boolean function satisfying

$$\sum_{|S|>t} \hat{f}(S)^2 \leq \varepsilon.$$

For the real-valued function $g = \sum_{|S|\leq t} \hat{f}(S)\chi_S$ it follows by Parseval's identity that

$$\|f - g\|^2 = \sum_{|S|>t} \hat{f}(S)^2 \leq \varepsilon,$$

which means that f can be ε-approximated by fading out the high frequencies S with $|S| > t$. This observation reduces the learning problem for f to the problem of computing the Fourier coefficients $\hat{f}(S)$ for all low-order frequencies S with $|S| \leq t$.

This task can be accomplished by the *low-degree algorithm* which is the most fundamental algorithm in the context of learning via the Fourier transform. The algorithm is presented in Figure 1. The simple but crucial observation is that each coefficient $\hat{f}(S)$ can be expressed as the expectation $E[f(x)\chi_S(x)]$, where x is chosen uniformly at random. Thus each coefficient $\hat{f}(S)$ can be accurately estimated with high confidence by drawing a sufficiently large sample from $EX(f)$. More specifically, the low-degree algorithm draws a polynomial number in n^t, $1/\varepsilon$ and $\log 1/\delta$ of labeled examples from $EX(f)$ and computes for each S of size at most t an empirical estimate a_S for $\hat{f}(S)$. By applying the Chernoff-Hoeffding bound it follows that with probability $1 - \delta$, each estimate a_S is within additive error $\lambda = (\varepsilon/n^t)^{1/2}$ from its expected value $\hat{f}(S)$. In this case the approximation $g = \sum_{|S|\leq t} a_S\chi_S$ satisfies

$$\begin{aligned}\|f - g\|^2 &= \sum_{S}(\hat{f}(S) - \hat{g}(S))^2 = \sum_{|S|\leq t}(\hat{f}(S) - a_S)^2 + \sum_{|S|>t} \hat{f}(S)^2\\ &\leq n^t\lambda^2 + \varepsilon = 2\varepsilon,\end{aligned}$$

and since $\Pr[\mathrm{sgn}(g(x)) \neq f(x)] \leq \|f - g\|^2$, it follows that the output hypothesis $h = \mathrm{sgn}(g)$ is an $(\varepsilon/2)$-approximation of f. The running-time is dominated by the sample size and thus polynomial in n^t, $1/\varepsilon$ and $\log(1/\delta)$.

Theorem 1. *For any Boolean function f satisfying*

$$\sum_{|S|>t} \hat{f}(S)^2 \leq \varepsilon,$$

the low-degree algorithm on inputs t, ε, δ and access to $EX(f)$ outputs with probability $1 - \delta$ an $O(\varepsilon)$-approximation of f in time $\mathrm{poly}(n^t, 1/\varepsilon, \log(1/\delta))$.

input: frequency bound t, accuracy ε, confidence δ and access to $EX(f)$
output: a Boolean function h approximating f

1. request $m = \frac{2n^t}{\varepsilon} \ln(\frac{2n^t}{\delta})$ labeled examples $(x_i, f(x_i))$ from $EX(f)$
2. for each $S \subseteq [n]$ with $|S| \le t$ compute $a_S = \frac{1}{m} \sum_{i=1}^{m} f(x_i)\chi_S(x_i)$
3. output $h = \mathrm{sgn}(\sum_{|S|\le t} a_S \chi_S)$

Figure 1: The low-degree algorithm

Let us remark for further reference, that the low-degree algorithm can be easily generalized to Boolean functions f satisfying $\sum_{S\in G} \hat{f}(S)^2 \le \varepsilon$ for an arbitrary collection $G \subseteq 2^{[n]}$ of frequencies, provided that G is explicitly given as part of the input. In this case, the running-time is poly$(n, |G|, 1/\varepsilon, \log(1/\delta))$.

Based on Hastad's Switching Lemma [17], Linial et al. [33] showed that for any Boolean function f which is computable by an AC^0 circuit of depth d and size M it holds that

$$\sum_{|S|>t} \hat{f}(S)^2 \le 2M2^{-t^{1/d}/20}.$$

Applying the low-degree algorithm with $t = O(\log(M/\varepsilon)^d)$ immediately yields the following learning result.

Corollary 2. [33] *The class of* AC^0 *circuits of depth d and size M over n variables is learnable in time*

$$\mathrm{poly}(n^{\log(M/\varepsilon)^d}, \log(1/\delta)).$$

Since an m-term DNF formula is computable by a circuit of size $m + 1$ and depth 2 it further follows that m-term DNF formulas are learnable in time poly$(n^{\log(m/\varepsilon)^2}, \log(1/\delta))$.

By a result due to Kharitonov [25], Corollary 2 cannot be significantly improved under a plausible cryptographic assumption. However, as we will see in Section 3.4, the exponent d can be reduced to $d - 1$. This will imply that m-term DNF formulas are in fact learnable in time poly$(n^{\log(m/\varepsilon)}, \log(1/\delta))$.

3.2 The KM-Algorithm

For sufficiently simple functions it is sometimes possible to bound the 1-norm of the Fourier transform. A nice example provide decision trees, for which the 1-norm of the Fourier transform can be bounded by the number of nodes in the tree by fairly elementary methods [30]. So suppose that f is a Boolean function satisfying

$$\|\hat{f}\|_1 \le k.$$

```
Coef($S, k$):
    1. if $k = n$ and $C(S,k) \geq \theta^2$ then return($\{S\}$)
    2. if $k < n$ and $C(S,k) \geq \theta^2$ then
           return(Coef($S, k+1$) $\cup$ Coef($S \cup \{k+1\}, k+1$))
    3. return($\emptyset$)
```

Figure 2: The recursive procedure Coef

Then clearly,

$$\sum_{|\hat{f}(S)|\leq\varepsilon/k} \hat{f}(S)^2 \;\leq\; (\varepsilon/k) \sum_{|\hat{f}(S)|\leq\varepsilon/k} \hat{f}(S) \;\leq\; \varepsilon,$$

which means that f can be ε-approximated by ignoring the small Fourier coefficients having absolute value at most ε/k. This observation reduces the learning problem for f to the problem of finding all frequencies S whose Fourier coefficients are larger in absolute value than some given threshold θ.

This problem can be solved by an algorithm of Goldreich and Levin [16] which is a key ingredient in their proof that parity functions are hard-core predicates for one-way functions. The algorithm has been first applied in the learning setting by Kushilevitz and Mansour [30]. The idea behind the algorithm is best described in terms of the recursive procedure Coef given in Figure 2, which can be regarded as a depth-first search on the binary tree of depth n. Here, each node is given by its level $k \in [n]$ and a set $S \subseteq [1,k]$, where we use the notation $[m,n]$ to describe the set $\{m,\ldots,n\}$. In each node (S,k) we consider the sum

$$C(S,k) = \sum_{T\subseteq[k+1,n]} \hat{f}(S \cup T)^2.$$

If $k = n$, then Coef has reached at a leaf of the tree, and since in this case $C(S,k) = \hat{f}(S)^2$, the procedure returns S if and only if $C(S,k) \geq \theta^2$. If (S,k) is an inner node with $C(S,k) < \theta^2$, then there is no set $T \subseteq [k+1,n]$ with $|\hat{f}(S \cup T)| \geq \theta$. Thus there is no need to further explore the subtree of this node. Otherwise, Coef recursively continues the search with the two children $(S, k+1)$ and $(S \cup \{k+1\}, k+1)$ of (S,k). Invoking Coef with the root node $(\emptyset, 0)$, the procedure returns the collection of all frequencies corresponding to Fourier coefficients with an absolute value greater than θ.

Concerning the number of recursive calls performed by Coef, first note that by Parseval's identity,

$$\sum_{S\subseteq[1,k]} C(S,k) = \sum_{S\subseteq[n]} f(S)^2 = ||f||^2 = 1,$$

implying that in each level k there are at most $1/\theta^2$ nodes (S,k) with $C(S,k) \geq \theta^2$. Thus, the total number of recursive calls can be bounded by n/θ^2. The main algorithmic difficulty, however, is the computation of the sums $C(S,k)$. This can be solved by expressing $C(S,k)$ as a nested expectation. More precisely, consider the function $g\colon \{0,1\}^{n-k} \to \mathbb{R}$ which maps a suffix $x \in \{0,1\}^{n-k}$ to the expectation $E_y[f(yx)\chi_S(y)]$ for a uniformly chosen prefix $y \in \{0,1\}^k$. For any prefix $y \in \{0,1\}^k$ and $T \subseteq [k+1,n]$ we have $\chi_{S\cup T}(z) = \chi_{S\Delta T}(z) = \chi_S(y)\chi_T(x)$, where $z = yx$. Hence,

$$\hat{f}(S \cup T) = E_z[f(z)\chi_{S\cup T}(z)] = E_x[E_y[f(yx)\chi_S(y)\chi_T(x)]] = E_x[g(x)\chi_T(x)],$$

implying that $\hat{f}(S \cup T) = \hat{g}(T)$. Now, by using Parseval's identity, we can express $C(S,k)$ as

$$C(S,k) = \sum_{T\subseteq[k+1,n]} \hat{g}(T)^2 = E_x[g(x)^2] = E_x[E_y[f(yx)\chi_S(y)]^2]$$

for uniformly chosen $x \in \{0,1\}^{n-k}$ and $y \in \{0,1\}^k$. By applying the Chernoff-Hoeffding bound, this means that we can estimate $C(S,k)$ by first estimating the inner expectation for a small number of random suffixes x and then estimate the outer expectation as the average sum of the squared estimates for the inner expectation. For each given suffix x, the estimate for the inner expectation can be obtained from a small number of values $f(yx)$ for random prefixes y, which can be obtained by asking the oracle f. By using sufficiently accurate estimates of $C(S,k)$ it can be shown that with high probability, the modified search still runs in polynomial time and returns a set containing all frequencies S with $|\hat{f}(S)| \geq \theta$. Furthermore, each frequency S in the set satisfies $|\hat{f}(S)| = \Omega(\theta)$, which by Parseval's identity implies that the number of frequencies in the returned collection is at most $O(1/\theta^2)$.

Lemma 3. [30] *There is a randomized algorithm A such that for each Boolean function f and threshold $\theta > 0$, $A(\theta, \delta, f)$ outputs with probability $1-\delta$ a collection $G \subseteq 2^{[n]}$ of size $O(1/\theta^2)$ containing all frequencies S with $|\hat{f}(S)| \geq \theta$. A runs in time* $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$.

As observed by Jackson [19] the algorithm of Lemma 3 can also be applied to real-valued functions $f\colon \{0,1\}^n \to \mathbb{R}$. In this case, the number of recursive calls of the procedure Coef is bounded by $\|f\|n/\theta^2$ rather than n/θ^2. Further notice that the confidence in the Chernoff-Hoeffding bound depends on the range of the random variables whose mean we want to estimate. In case f is real-valued, the range of these random variables is $[-\|f\|_\infty^2, \|f\|_\infty^2]$ rather than $[-1,1]$, implying that the number of labeled examples from f needed to estimate $C(S,k)$ additionally depends on the parameter $\|f\|_\infty$. Since $\|f\| \leq \|f\|_\infty$, the running-time becomes polynomial in n, $1/\theta$, $\log(1/\delta)$ and $\|f\|_\infty$. Let us state this extension to real-valued functions for later use.

Lemma 4. [19] *There is a randomized algorithm A such that for each function* $f\colon \{0,1\}^n \to \mathbb{R}$ *satisfying* $\|f\|_\infty \le k$ *and threshold* $\theta > 0$, $A(\theta,\delta,k,f)$ *outputs with probability* $1-\delta$ *a collection* $G \subseteq 2^{[n]}$ *containing all frequencies S with* $|\hat{f}(S)| \ge \theta$. *A runs in time* $\mathrm{poly}(n,k,1/\theta,\log(1/\delta))$.

Coming back to Boolean functions $f \in B_n$ satisfying $\|\hat{f}\|_1 \le k$, we can use the algorithm of Lemma 3 to find a small collection G containing all frequencies S for which $|\hat{f}(S)| \ge \varepsilon/k$ in time $\mathrm{poly}(n,k,1/\varepsilon,\log(1/\delta))$, and then use the generalized low-degree algorithm on G to output an $O(\varepsilon)$-approximation of f. This algorithm is known as the *Kushilevitz-Mansour algorithm*, or simply *KM-algorithm*, and we state its properties in the following theorem.

Theorem 5. [30] *For each Boolean function f satisfying* $\|\hat{f}\|_1 \le k$, *the KM-algorithm, given inputs k, ε, δ and oracle access to f, outputs with probability* $1-\delta$ *an* $O(\varepsilon)$*-approximation of f in time* $\mathrm{poly}(n,k,1/\varepsilon,\log(1/\delta))$.

As already mentioned at the beginning of this section, any Boolean function f computable by a decision tree with m nodes satisfies $\|\hat{f}\|_1 \le m$. Thus, the KM-algorithm learns decision trees in polynomial time, although, a disadvantage of this result is that membership queries are needed.

Corollary 6. [30] *The class of decision trees with m nodes is learnable with membership queries in time* $\mathrm{poly}(n,m,1/\varepsilon,\log(1/\delta))$.

For some applications, it is more convenient to consider the *sparseness* of a Boolean function f. This property is closely related to the 1-norm of $\hat{f}$. We say that a function f is k*-sparse*, if the support $\{S \subseteq [n] \mid \hat{f}(S) \ne 0\}$ of $\hat{f}$ has size at most k. Clearly, if f is k-sparse then $\|\hat{f}\|_1 \le k$. On the other hand, if $\|\hat{f}\|_1 \le k$, then f is ε-close to the function $g = \sum_{|\hat{f}(S)|>\varepsilon/k} \hat{f}(S)\chi_S$. By Parseval's identity, the number of frequencies S with $|\hat{f}(S)| > \varepsilon/k$ is less than k^2/ε^2. Hence it follows that g is (k^2/ε^2)-sparse. We call a Boolean function (ε,k)*-sparse* if it is ε-close to a k-sparse function. It is not hard to show (see [30]) that for any (ε,k)-sparse function f,

$$\sum_{|\hat{f}(S)|\le\varepsilon/k} \hat{f}(S)^2 \le \varepsilon + \varepsilon^2/k.$$

Thus, the KM-algorithm is applicable to (ε,k)-sparse Boolean functions.

Corollary 7. [30] *For each* (ε,k)*-sparse function f, the KM-algorithm given inputs k, ε, δ and oracle access to f outputs an* $O(\varepsilon)$*-approximation of f in time* $\mathrm{poly}(n,k,1/\varepsilon,\log(1/\delta))$.

Mansour [36] showed that each DNF with terms of size at most d is (ε,k)-sparse for $k = d^{O(d\log(1/\varepsilon))}$. By an argument attributed to Warmuth in [44], every m-term DNF is ε-close to a DNF with terms of size at most $\log(m/\varepsilon)$ (by simply ignoring all terms of size larger than $\log(m/\varepsilon)$). Hence, an m-term DNF f is ε-close to a (ε, k)-sparse function for $k = (\log(m/\varepsilon))^{O(\log(m/\varepsilon)\log(1/\varepsilon))} = (m/\varepsilon)^{O(\log\log(m/\varepsilon)\log(1/\varepsilon))}$, which by the triangle inequality implies that f itself is $(O(\varepsilon), k)$-sparse.

Corollary 8. [36] *The class of m-term DNF formulas is learnable with membership queries in time* $\mathrm{poly}(n, (m/\varepsilon)^{\log\log(m/\varepsilon)\log(1/\varepsilon)}, 1/\varepsilon, \log(1/\delta))$.

3.3 The Harmonic Sieve

In the last section we described a quasipolynomial-time learning algorithm for DNF formulas based on the sparseness of these functions. Another property of the Fourier transform of m-term DNF is that the ∞-norm can be lower bounded in terms of m [4]. This property provides the basis for Jackson's celebrated polynomial-time learning algorithm for DNF formulas which we present in this section. So, for a Boolean function f satisfying

$$||\hat{f}||_\infty \geq \gamma,$$

assume that S is a frequency with $\hat{f}(S) \geq \gamma$. Expressing the coefficient $\hat{f}(S)$ in terms of the probability that $f(x) \neq \chi_S(x)$,

$$\hat{f}(S) = E[f(x)\chi_S(x)] = 1 - 2\Pr[f(x) \neq \chi_S(x)],$$

it follows that

$$\Pr[f(x) \neq \chi_S(x)] \leq \frac{1 - \hat{f}(S)}{2}.$$

This means that the function f can be weakly $(\gamma/2)$-approximated by a single parity function χ_S or its negation $-\chi_S$, where we use χ_S if $\hat{f}(S)$ is positive, and its negation otherwise. The corresponding frequency S can be found by the KM-algorithm with high probability in time $\mathrm{poly}(n, 1/\gamma)$, and the sign of $\hat{f}(S)$ can be easily determined by estimating $\hat{f}(S)$ within additive error less than γ. Thus, for any $\gamma > 0$ and every Boolean function f satisfying $||\hat{f}||_\infty \geq \gamma$ we can produce with high probability a weak $\Omega(\gamma)$-approximation of f in time $\mathrm{poly}(n, 1/\gamma)$, provided that we have access to the oracle f.

This reasoning can be generalized to arbitrary distributions D by using the crucial observation that the correlation $E_D[f\chi_S]$ between f and a parity χ_S with respect to D can be expressed as the correlation $E[f_D(x)\chi_S(x)]$ between the real-valued function $f_D(x) = 2^n D(x) f(x)$ and χ_S with respect to the uniform distribution. Thus, $E_D[f(x)\chi_S(x)] = E[f_D(x)\chi_S(x)]$ coincides with the Fourier coefficient $\hat{f}_D(S)$ of the function f_D, implying that

$$\Pr_D[f(x) \neq \chi_S(x)] \leq \frac{1 - \hat{f}_D(S)}{2}.$$

```
input: $\gamma, \varepsilon > 0$ and a weak learning algorithm $A$
output: a Boolean function $h$ approximating $f$

1. $i \leftarrow 0$;
2. while $|M_i| > \varepsilon 2^n$ do
       run $A$ to produce a weak $(\gamma, D_i)$-approximation $h_i$;
       $i \leftarrow i + 1$;
3. return $h = \mathrm{sgn}(\sum_{j=0}^{i} h_j(x))$
```

Figure 3: The IHA-boosting algorithm

If we now assume that $\|\hat{f_D}\|_\infty \geq \gamma$ (rather than $\|\hat{f}\|_\infty \geq \gamma$), then f can be weakly $(\gamma/2, D)$-approximated by a single parity function χ_S or its negation. Further, on input γ, the corresponding frequency S with $|\hat{f_D}(S)| = \Omega(\gamma)$ can be found by the algorithm of Lemma 4 using oracle access to f_D.

Lemma 9. *There is a randomized algorithm A such that for each distribution D and Boolean function f satisfying $\|\hat{f_D}\|_\infty \geq \gamma$ and $\|f_D\|_\infty \leq k$, $A(k, \gamma, \delta, f_D)$ outputs with probability $1 - \delta$ a weak $(\Omega(\gamma), D)$-approximation of f in time* $\mathrm{poly}(n, k, 1/\gamma, \log(1/\delta))$.

Boosting [41] is a well-known technique to transform a weak learner into a strong learner. The technique can be most easily described in the distribution-free setting, where we assume that the weak learner produces a weak (γ, D)-approximation of the target f for any distribution D. With respect to some fixed but unknown target distribution D, the boosting algorithm runs the weak learner several times with respect to different distributions D_i, which forces the weak learner to perform well on different regions of the instance space. The resulting weak (γ, D_i)-approximations are then combined in some way to produce a strong (ε, D)-approximation of f. Obviously, the oracle access to f required by the boosting algorithm depends on how the weak learner accesses the oracle. If the weak learner asks membership queries, then also the boosting algorithm needs to ask membership queries. If the weak learner needs only access to $EX(f, D_i)$, then it is usually possible to apply a filtering technique in order to simulate the $EX(f, D_i)$ oracle by asking queries to $EX(f, D)$. This means that each example $(x, f(x))$ drawn from $EX(f, D)$ is discarded by the boosting algorithm with a certain probability depending on $(x, f(x))$ and D_i, and only the remaining examples are passed on to the weak learner.

There are several boosting strategies which mainly differ in how the distributions D_i are defined, and in how the final hypothesis is obtained from the weak hypotheses. The *IHA-boosting algorithm* (see Figure 3) uses a particularly wellsuited boosting strategy which is based on a construction of a hard-core distribution due to Impagliazzo [18]. The boosting ability of this construction has been pointed out by Klivans and Servedio [26]. In order to achieve strong learning with respect to the uniform target distribution, the IHA-boosting algorithm uses the following distributions D_i. Suppose that the weak learning algorithm has already produced the hypotheses $h_0, \ldots, h_{i-1}$ for some $i \geq 0$, and let $h = \mathrm{sgn}(\sum_{j=0}^{i-1} h_j(x))$ denote the majority vote of these hypotheses. First consider the *margin*

$$N_i(x) = f(x) \sum_{j=0}^{i-1} h_j(x),$$

by which h agrees with the target f on an instance x. Note that h disagrees with f on x only if $N_i(x)$ is negative. Next we define a *measure* M_i on the instance space $\{0,1\}^n$ which assigns weight 0 to the instances with large margin, weight 1 to the instances with negative margin, and intermediate weights to instances with non-negative but small margin. More precisely,

$$M_i(x) = \begin{cases} 0, & N_i(x) \geq 1/\gamma, \\ 1, & N_i(x) \leq 0, \\ 1 - \gamma N_i(x), & \text{otherwise.} \end{cases}$$

Notice that $M_0(x) = 1$ for all x. The distribution D_i is now obtained by standardizing the measure M_i by the *weight* $|M_i| = \sum_x M_i(x)$ of M_i,

$$D_i(x) = \frac{M_i(x)}{|M_i|}.$$

Whenever the IHA-boosting algorithm is going to run the weak learner A, it first checks whether the measure M_i satisfies $|M_i| \leq \varepsilon 2^n$. Observe that the current majority vote h disagrees with f on x only if $N_i(x) \geq 0$. In this case $M_i(x) = 1$ and hence the approximation error of h can be bounded by

$$\Pr[h(x) \neq f(x)] \leq 2^{-n} \sum_x M_i(x) = 2^{-n}|M_i|.$$

Thus, the condition $|M_i| \leq \varepsilon 2^n$ guarantees that the boosting algorithm has found the desired ε-approximation of f.

Impagliazzo [18] showed that the abort condition $|M_i| \leq \varepsilon 2^n$ is met after at most $O(1/\gamma^2\varepsilon^2)$ runs of A. Since for all x we have that $D_i(x) \leq 1/|M_i|$, the ∞-norm of the distributions D_i is bounded by $1/|M_i|$ and hence the weak learner A is run only on distributions D_i satisfying $\|2^n D_i\|_\infty \leq 1/\varepsilon$.

An algorithmic difficulty is the computation of the exponentially large sum $|M_i|$ which is required to check the abort condition. This difficulty can be overcome by first expressing $2^{-n}|M_i|$ as the expected value $E[M_i(x)]$ for a uniformly chosen x. Then we only have to observe that a random example $(x, M_i(x))$ can be easily obtained from a random example $(x, f(x))$. Furthermore, $M_i(x)$ takes only values in the range $[0, 1]$. Hence, with high probability we can get an accurate estimate for $2^{-n}|M_i|$ by drawing a small sample from $EX(f)$. Now, by using an estimate rather than the exact value of $2^{-n}|M_i|$, it is easy to adjust the abort condition so that (1) the output h is still an ε-approximation of f and (2) the modified abort condition can still be met after at most $O(1/\gamma^2\varepsilon^2)$ runs of A with distributions D_i, where (3) each D_i still satisfies $\|2^n D_i\|_\infty = O(1/\varepsilon)$.

Now suppose that f is a Boolean function satisfying for *all* distributions D the bound

$$\|\hat{f}_D\|_\infty \geq \gamma.$$

The *Harmonic Sieve* for learning f, as suggested in [26], runs the IHA-boosting algorithm[1] based on the weak learner A provided by Lemma 9. Recall that A produces with sufficiently high probability a weak $(\Omega(\gamma), D_i)$-approximation of the target f in time $\text{poly}(n, k, 1/\gamma, \log(1/\delta))$, provided that A gets an upper bound k on the ∞-norm of f_{D_i} and has access to the oracle f_{D_i}. Further, recall that f_{D_i} is defined as $f_{D_i}(x) = 2^n D_i(x) f(x)$ and each distribution D_i satisfies $\|2^n D_i\|_\infty = O(1/\varepsilon)$. Hence, $\|f_{D_i}\|_\infty = O(1/\varepsilon)$ and we can easily provide A with the required bound k. The resulting running time of A becomes $\text{poly}(n, 1/\gamma, 1/\varepsilon, \log(1/\delta))$.

The remaining obstacle is the fact that the boosting algorithm cannot provide the weak learner with the exact values of $f_{D_i}(x)$. However, it can compute an accurate approximation of $f_{D_i}(x) = 2^n M_i(x)/|M_i|$ by using the already calculated estimate for $2^{-n}|M_i|$ together with the value $M_i(x)$. Note that the latter value can be exactly computed from $f(x)$ by using a single membership query to f. It can be shown that using a sufficiently accurate approximation of f_{D_i} does not have a significant impact on the learning ability of A (cf. [19]). Thus, with high probability, A indeed produces in each iteration a weak $(\Omega(\gamma), D_i)$-approximation of f which can be used by the boosting algorithm to produce an ε-approximation. The running time of the Harmonic Sieve is roughly $O(1/\gamma^2\varepsilon^2)$ times the time required for each simulation of A which is $\text{poly}(n, 1/\gamma, 1/\varepsilon, \log(1/\delta))$. Thus, the Harmonic Sieve achieves the following performance.

Theorem 10. *(cf.* [19]) *For each Boolean function f satisfying for all distributions D the bound $\|\hat{f}_D\|_\infty \geq \gamma$, the Harmonic Sieve on inputs ε, γ, δ and oracle access to f outputs with probability* $1 - \delta$ *an $O(\varepsilon)$-approximation of f in time* $\text{poly}(n, 1/\varepsilon, 1/\gamma, \log(1/\delta))$.

Jackson [19] showed that for every m-term DNF f and for all distributions D on $\{0, 1\}^n$ it holds that $\|\hat{f}_D\|_\infty = \max_S |E_D[f\chi_S]| \geq 1/(2m + 1)$. Hence, the

[1]In [19], Jackson used the F1 boosting algorithm of Freund [13].

Harmonic Sieve can be used to efficiently learn DNF formulas with membership queries.

Corollary 11. [19] *The class of m-term DNF formulas is learnable with membership queries in time* $\text{poly}(n, m, 1/\varepsilon, \log(1/\delta))$.

We notice that in the *random walk model* [1], the need for membership queries can be avoided by using the *Bounded Sieve* of Bshouty and Feldman [7]. The random walk model is a variant of the PAC model where the examples are generated by performing a random walk on the cube (hence, they are not independent). The idea is to search for large coefficients by performing a breadth-first search on the Boolean hypercube rather than a depth-first search on the binary tree. Using the crucial property that the Fourier spectrum of a DNF provides a large coefficient within the low-order spectrum, Bshouty et al. [8] showed that the Bounded Sieve can be used to efficiently learn DNF formulas in the random walk model. This property of DNF formulas will also play an important role in the next subsection.

3.4 Exhaustive Search

The main drawback of the Harmonic Sieve is its need for membership queries which are used by the underlying KM-algorithm to guide the search for large coefficients. The expensive use of membership queries can be avoided, if the low-order spectrum of the target contains a large coefficient. More precisely, let f be a Boolean function satisfying for each distribution D the bound

$$\max_{|S|\le t} |\hat{f}_D(S)| \ge \gamma.$$

Then f can be weakly $(\gamma/2, D)$-approximated by a single parity function χ_S or its negation where $|S| \le t$. Hence, it suffices to perform an exhaustive search over all frequencies S with $|S| \le t$; similar to the low-degree algorithm. This immediately yields the following weak learning result.

Lemma 12. [20] *There is a randomized algorithm A such that for each distribution D and Boolean function f satisfying* $\max_{|S|\le t} |\hat{f}_D(S)| \ge \gamma$, $A(t, \gamma, \delta, EX(f, D))$ *outputs with probability* $1 - \delta$ *a weak* $(\Omega(\gamma), D)$*-approximation of f in time* $\text{poly}(n^t, 1/\gamma, \log(1/\delta))$.

Applying the IHA boosting algorithm to the weak learning algorithm of Lemma 12, we can exploit the fact that the boosting algorithm only uses distributions D_i with $\|2^n D_i\|_\infty = O(1/\varepsilon)$. This yields the following strong learning result *without* membership queries.

Theorem 13. [20] *There is a randomized algorithm A such that for each Boolean function f satisfying for all distributions D with $\|2^n D\|_\infty = O(1/\varepsilon)$ the bound $\max_{|S|\le t} |\hat{f}_D(S)| \ge \gamma$, $A(t, \gamma, \varepsilon, \delta, EX(f))$ outputs with probability $1 - \delta$ an ε-approximation of f in time* $\mathrm{poly}(n^t, 1/\varepsilon, 1/\gamma, \log(1/\delta))$.

An MAC^0 circuit consists of a majority-gate over a polynomial number of AC^0 circuits. Jackson et al. [20] showed that for every distribution D and for every Boolean function f computable by an MAC^0 circuit of size M and depth d it holds that

$$\max_{|S|\le t} |\hat{f}_D(S)| = \Omega(1/Mn^t),$$

where $t = O(\log(M\|2^n D\|_\infty))^{d-1}$. In particular, $t = O(\log(M/\varepsilon))^{d-1}$ for every distribution D satisfying $\|2^n D\|_\infty \le 1/\varepsilon$. Thus we get the following improvement of Corollary 2.

Corollary 14. [20] *The class of* MAC^0 *circuits of size M and depth d is learnable in time*

$$\mathrm{poly}(n^{O(\log(M/\varepsilon))^{d-1}}, \log(1/\delta)).$$

By Corollary 14 it immediately follows that m-term DNF formulas are learnable in time $\mathrm{poly}(n^{O(\log(m/\varepsilon))}, \log(1/\delta))$ without membership queries. Let us remark that a similar result already has been obtained by Verbeurgt [44], though by using a different approach.

An algorithm similar to the one in Theorem 13 can also be applied in the model of *statistical queries* [23]. In this model it is possible to obtain $\hat{f}(S)$ within additive error τ by asking a single statistical query. The parameter τ is called the *tolerance* of the query. It can be shown that m-term DNF formulas are learnable with $n^{O(\log(m/\varepsilon))}$ statistical queries, provided that $\tau^{-1} = \mathrm{poly}(m/\varepsilon)$ [28]. This has been improved to $\tau^{-1} = O(m/\varepsilon)$ in [31]. Interestingly, learning m-term DNF formulas *requires* $n^{\Omega(\log(m))}$ statistical queries as long as the tolerance is sufficiently large [4].

3.5 Problems

Let us close this section by highlighting some important problems and suggestions for further research concerning the learnability of DNF formulas without membership queries. For the sake of clarity of exposition we omit the reference to the parameters ε and δ.

Clearly, the ultimate goal is to achieve the analogue of Jackson's learnability result for DNF formulas without using membership queries.

Problem 15. *Are m-term DNF formulas learnable in time* $\mathrm{poly}(n, m)$*?*

Less ambiguous, but still a major break-through would be a polynomial-time learning algorithm for DNF formulas with a non-constant number $m(n)$ of terms. In Section 4.3 we will present an algorithm which achieves this goal for the subclass of monotone DNF formulas with running-time $\text{poly}(n, (m \log n)^{\varphi(m)})$ for $\varphi(m) = \sqrt{m \log(m)}$.

Problem 16. *Is the class of m-term DNF formulas learnable in time* $\text{poly}(n, (m \log n)^{\varphi(m)})$, *where φ does not depend on n?*

In Section 3.4 we saw that m-term DNF formulas are learnable in time $\text{poly}(n^{\log m})$. This is the best known learning result for general m-term DNF without membership queries. So even an improvement to $\text{poly}(n^{(\log m)^{\alpha}})$ for some $\alpha < 1$ would be very interesting.

As a first step towards solving Problem 16, one might attack the easier problem of learning decision trees instead of DNF formulas.

Problem 17. *Is the class of decision trees with m nodes learnable in time* $\text{poly}(n, (m \log n)^{\varphi(m)})$, *where φ does not depend on n?*

As we will see in section 4.4, the subclass of monotone decision trees with m nodes is learnable in time $\text{poly}(n, m)$.

4 Monotone functions and influence

The sensitivity of a Boolean function f is a measure of how strongly $f(x)$ reacts to a change of its variables. It is closely related to the notion of influence of single variables on the value of f. Interestingly, the sensitivity (as well as the influence) can be expressed in terms of the Fourier coefficients of f yielding good approximations for functions having low sensitivity. This approach works especially well for monotone functions, since in this case, the influence values of the individual variables $x_1, \ldots, x_n$ constitute the Fourier spectrum on the singleton frequencies $\{x_1\}, \ldots, \{x_n\}$.

In this section we review some important learning results for monotone Boolean functions that are based on sensitivity arguments, including Bshouty and Tamon's [9] work on monotone functions, Servedio's learnability result for monotone DNF [40], and the very recent learning algorithm of O'Donnell and Servedio for monotone decision trees [39]. We start with a discussion of the influence and sensitivity of a Boolean function f.

Influence and sensitivity

The concept of influence of a variable on a Boolean function was introduced by Ben-Or and Linial [3]. Let f be a Boolean function on n variables $x_1, \ldots, x_n$. The *influence* $I_j(f)$ of x_j on f is defined as the probability that flipping the j-th bit in a uniformly at random chosen assignment $x = (x_1, \dots, x_n)$ changes the value of f. More formally,

$$I_j(f) = \Pr[f(x) \neq f(x \oplus e_j)],$$

where x is uniformly at random chosen from $\{0,1\}^n$ and e_j denotes the assignment $0^{j-1}10^{n-j-1}$. The *total influence* of f is defined as $I(f) = \sum_j I_j(f)$. It is easy to see that $I(f)$ equals the average sensitivity of f on all assignments $x \in \{0,1\}^n$, where the *sensitivity* of f on $x = (x_1, \dots, x_n)$ is defined as the number of bits in x whose flipping causes f to change its value. Note that $I(f)$ further coincides with the fraction of edges in the Boolean hypercube that connect assignments x and x' having different values under f. Let us consider the influence of some basic functions.

- The dictatorship function χ_i maps $(x_1, \dots, x_n) \mapsto (-1)^{x_i}$. Clearly, $I_j(\chi_i) = 1$ if $i = j$, and $I_j(\chi_i) = 0$ otherwise. Hence, the dictatorship function has total influence $I(\chi_i) = 1$.

- The influence of a single variable x_j on the parity function $\chi_{[n]}$ is $I_j(\chi_{[n]}) = 1$ and hence the total influence $I(\chi_{[n]})$ sums up to n.

- The influence of x_j on the majority function MAJ_n is

$$I_j(\mathrm{MAJ}_n) = \binom{n-1}{\lfloor n/2 \rfloor} \Big/ 2^{n-1},$$

  implying that $I(\mathrm{MAJ}_n) < \sqrt{2n/\pi}$ for $n \geq 2$. In fact, it is not hard to show (e.g., see [15]) that for any monotone n-ary Boolean function f, $I(f) \leq I(\mathrm{MAJ}_n)$ implying $I(f) < \sqrt{2n/\pi}$. We will prove a slightly weaker bound in Proposition 20.

- The influence of a *k-junta*, i.e., of a function f depending only on a fixed set R of at most k variables, is clearly bounded by $I(f) \leq k$, since all variables x_j with $j \notin R$ have influence $I_j(f) = 0$.

As has been observed in [22], the influence of x_j on a Boolean function f coincides with the weight of the Fourier spectrum on all frequencies containing j. In fact, since $\hat{f}_{\oplus y}(S) = \chi_S(y)\hat{f}(S)$ (cf. Equation (2) in Section 2), the function $f_j = (f - f_{\oplus e_j})/2$ has the coefficients

$$\hat{f}_j(S) = \frac{\hat{f}(S) - \hat{f}_{\oplus e_j}(S)}{2} = \frac{\hat{f}(S) - \chi_S(e_j)\hat{f}(S)}{2} = \begin{cases} \hat{f}(S), & j \in S \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

From Parseval's identity it follows that

$$I_j(f) = E[|f_j|] = E[f_j^2] = 2^{-n} \sum_x f_j(x)^2 = \sum_S \hat{f}_j(S)^2 = \sum_{S\,:\,j \in S} \hat{f}(S)^2.$$

Proposition 18. *For any Boolean function f,*

$$I_j(f) = \|f_j\|^2 = \sum_{S\,:\,j\in S} \hat{f}(S)^2.$$

Hence, the total influence is $I(f) = \sum_S |S|\hat{f}(S)^2$.

For monotone f, the influence $I_j(f)$ coincides with the Fourier coefficient $\hat{f}(j)$. Here we adopt the convention that f is called *monotone,* if flipping any 0-bit in x to 1 does not change the value of $f(x)$ from *true* to *false* (recall that *true* is represented by the number -1 and *false* by 1). For a bit $b \in \{0,1\}$ we use $f_{j,b}$ to denote the Boolean function $f_{j,b}(x) = f(x_1,\ldots,x_{j-1},b,x_{j+1},\ldots,n)$. Now it is easy to see that

$$|f - f_{\oplus e_j}| = (f_{j,0} - f_{j,1}) = (f - f_{\oplus e_j})\chi_j,$$

and hence, using (5), the influence of x_j evaluates to

$$I_j(f) = E\left[\frac{|f - f_{\oplus e_j}|}{2}\right] = \frac{E[f\chi_j - f_{\oplus e_j}\chi_j]}{2} = \frac{\hat{f}(j) - \hat{f}_{\oplus e_j}(j)}{2} = \hat{f}_j(j) = \hat{f}(j).$$

Proposition 19. *For any monotone Boolean function f,*

$$I_j(f) = \hat{f}(j).$$

Hence, the total influence is $I(f) = \sum_j \hat{f}(j)$.

Letting $R = \{j \in [n] \mid I_j(f) > 0\}$ be the set of relevant variables of f and denoting the number of relevant variables by k, it follows from Cauchy-Schwarz's inequality and Parseval's identity that

$$I(f)^2 = (\sum_{j\in R} I_j(f))^2 \le k\sum_{j\in R} I_j(f)^2 = k\sum_{j\in R}\hat{f}(j)^2 \le k.$$

Hence the total influence of a monotone function can be bounded as follows.

Proposition 20. *For any monotone k-junta f,*

$$I(f) \le \sqrt{k}.$$

4.1 Monotone Boolean functions

As we have seen, the low-degree algorithm succeeds on all targets having small weight on the high frequencies of their Fourier spectrum. As we will see in the proof of following proposition, the high frequency weight can be bounded in terms of the influence. Thus, a small bound on the influence guarantees a good performance of the low-degree algorithm.

Proposition 21. *Let f be a Boolean function satisfying $I(f) \leq l$. Then*

$$\sum_{|S| \geq l/\varepsilon} \hat{f}(S)^2 \leq \varepsilon.$$

Proof. We can bound the high frequency weight for any bound t as follows.

$$\sum_{|S| \geq t} \hat{f}(S)^2 \leq \sum_{|S| \geq t} |S| \hat{f}(S)^2/t \leq \sum_{S} |S| \hat{f}(S)^2/t = I(f)/t.$$

Hence, the claim follows by choosing $t = l/\varepsilon$. ■

By applying the low-degree algorithm, Proposition 21 immediately yields the following theorem.

Theorem 22. *There is a randomized algorithm A such that for each Boolean function f satisfying $I(f) \leq l$, $A(l, \varepsilon, \delta, EX(f))$ outputs with probability $1 - \delta$ an $O(\varepsilon)$-approximation for f in time* $\text{poly}(n^{l/\varepsilon}, \log(1/\delta))$.

By Proposition 20, for a monotone function f we have the bound $I(f) \leq \sqrt{n}$, which immediately yields the following learning result of Bshouty and Tamon.

Corollary 23. [9] *The class of monotone Boolean functions is learnable in time* $\text{poly}(n^{\sqrt{n}/\varepsilon}, \log(1/\delta))$.

In Section 4.4 we will present an algorithm for monotone Boolean functions running in time $\text{poly}(n, 2^{(l/\varepsilon)^2})$ rather than $\text{poly}(n^{l/\varepsilon})$ as in Theorem 22.

4.2 Monotone juntas

By Proposition 20 we know that the influence of a monotone k-junta is bounded by $\sqrt{k}$. Hence, the algorithm of Theorem 22 finds a low degree $O(\varepsilon)$-approximation g for f in time $\text{poly}(n^{\sqrt{k}/\varepsilon}, \log(1/\delta))$. Bshouty and Tamon [9] observed that by ignoring variables of sufficiently small influence, the running-time can be improved to $\text{poly}(k^{\sqrt{k}/\varepsilon}, \log(1/\delta))$.

For a bound $\theta \geq 0$ let

$$R(\theta) = \{j \in [n] \mid I_j(f) > \theta\}$$

denote the set of variables having influence greater than θ. Then it is easy to bound the Fourier weight of a k-junta on the frequencies containing at least one variable of small influence.

Proposition 24. *For any k-junta f,*

$$\sum_{S \not\subseteq R(\varepsilon/k)} \hat{f}(S)^2 \leq \varepsilon.$$

Proof. Since for a k-junta the number $|R|$ of relevant variables is bounded by k and since each variable x_j with $j \notin R(\theta)$ has influence $I_j(f) \leq \theta$, it follows that

$$\sum_{S \not\subseteq R(\theta)} \hat{f}(S)^2 \leq \sum_{j \notin R(\theta)} \sum_{S\,:\, j \in S} \hat{f}(S)^2 = \sum_{j \in R - R(\theta)} I_j(f) \leq \theta k.$$

Hence, the claim follows by choosing $\theta = \varepsilon/k$. ■

In the following we will frequently consider the collection of all frequencies S of order at most t containing only variables having influence greater than θ, which we denote by

$$G(\theta, t) = \{S \subseteq R(\theta) \mid |S| \leq t\}.$$

The following proposition shows that a k-junta f with influence $I(f) \leq l$ can be $O(\varepsilon)$-approximated by taking only the coefficients corresponding to frequencies inside $G(\varepsilon/k, l/\varepsilon)$.

Proposition 25. *Let f be a k-junta satisfying $I(f) \leq l$. Then*

$$\sum_{S \notin G(\varepsilon/k, l/\varepsilon)} \hat{f}(S)^2 \leq 2\varepsilon.$$

Proof. Using Propositions 21 and 24 it immediately follows that

$$\sum_{S \notin G(\varepsilon/k, l/\varepsilon)} \hat{f}(S)^2 \leq \sum_{S \not\subseteq R(\varepsilon/k)} \hat{f}(S)^2 + \sum_{|S| \geq l/\varepsilon} \hat{f}(S)^2 \leq 2\varepsilon.$$

■

If f is monotone, we can collect all variables having large influence by estimating the Fourier coefficients of all singleton frequencies. More precisely, in order to find all variables x_j having influence $I_j(f) \geq \theta$, we compute estimates a_j for the Fourier coefficients $\hat{f}(j)$ by drawing sufficiently many examples from the oracle $EX(f)$ and collect all variables x_j with $a_j \geq 3\theta/4$. Then, with high probability, we get all variables x_j with $I_j(f) \geq \theta$ and no variables x_j with $I_j(f) \leq \theta/2$. This algorithm has been called `Find-Variables` by Servedio [40].

Proposition 26. *For each monotone Boolean function f, `Find-Variables` on inputs θ, δ and access to $EX(f)$, outputs with probability $1 - \delta$ in time $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$ a set R^* with $R(\theta) \subseteq R^* \subseteq R(\theta/2)$.*

In order to compute an ε-approximation for a monotone k-junta f, we first use `Find-Variables` with $\theta = \varepsilon/k$ to obtain with high probability a variable set $R^* \subseteq R(\varepsilon/2k)$ containing all variables x_j with $I_j(f) \geq \varepsilon/k$. Since $R^* \subseteq R(\varepsilon/2k)$ implies that R^* contains only relevant variables, the size of $G^* = \{S \subseteq R^* \mid |S| \leq l/\varepsilon\}$ is polynomial in $k^{l/\varepsilon}$. Hence, we can apply the generalized low-degree algorithm to get the following learning result for monotone k-juntas having small influence.

Theorem 27. *There is a randomized algorithm A such that for each monotone k-junta f with $I(f) \leq l$, $A(k, l, \varepsilon, \delta, EX(f))$ outputs with probability* $1 - \delta$ *an* $O(\varepsilon)$-*approximation for f in time* $\text{poly}(n, k^{l/\varepsilon}, \log(1/\delta))$.

Corollary 28. [9] *The class of monotone k-juntas is learnable in time*

$$\text{poly}(n, k^{\sqrt{k}/\varepsilon}, \log(1/\delta)).$$

4.3 Monotone functions that are close to juntas

By refining the arguments used in the preceding subsection we will now see that the generalized low-degree algorithm also succeeds on monotone Boolean functions that are sufficiently close to a monotone k-junta [9]. As a consequence, the generalized low-degree algorithm (in conjunction with `Find-Variables`) becomes applicable to monotone m-term DNF formulas.

We call a function f an (ε, k)-*junta,* if f is ε-close to a k-junta. We first show that for an (ε, k)-junta, the number of variables x_j having influence $I_j(f) \geq \varepsilon$ is bounded by k.

Proposition 29. *For any (ε, k)-junta, $|R(\varepsilon)| \leq k$.*

Proof. Let h be a k-junta that is ε-close to f. Observe that for any variable x_j with $I_j(h) = 0$ it holds that $\hat{h}(S) = 0$ if $j \in S$. Hence,

$$I_j(f) = \sum_{S\,:\,j \in S} \hat{f}(S)^2 = \sum_{S\,:\,j \in S} (\hat{f}(S) - \hat{h}(S))^2 \leq \|f - h\|^2 \leq \varepsilon.$$

This shows that any variable in $R(\varepsilon)$ must be relevant for h. ∎

In the next proposition we bound the weight of the high order Fourier spectrum of f under the assumption that f is close to some function g with a small weight on this region.

Proposition 30. *For any function f that is ε-close to some Boolean function g with $\sum_{|S| \geq t} \hat{g}(S)^2 \leq \varepsilon$,*

$$\sum_{|S| \geq t} \hat{f}(S)^2 \leq 4\varepsilon.$$

Proof. Using the inequality $\hat{f}^2 = (\hat{f} - \hat{g} + \hat{g})^2 \le 2(\hat{f} - \hat{g})^2 + 2\hat{g}^2$ (Cauchy-Schwarz) it follows that

$$\sum_{|S|\ge t} \hat{f}(S)^2 \le 2 \sum_{|S|\ge t} (\hat{f}(S) - \hat{g}(S))^2 + 2 \sum_{|S|\ge t} \hat{g}(S)^2 \le 4\varepsilon.$$

■

Now assume that f is an $(\varepsilon/n, k)$-junta with the additional property that it is ε-close to some Boolean function g whose Fourier weight on the frequencies S with $|S| \ge t$ is bounded by ε. Then by using Proposition 24 with $k = n$ as well as Proposition 30, it follows that the frequency collection $G(\varepsilon/n, t)$ has the property

$$\sum_{S \notin G(\varepsilon/n,t)} \hat{f}(S)^2 \le \sum_{S \not\subseteq R(\varepsilon/n)} \hat{f}(S)^2 + \sum_{|S|\ge t} \hat{f}(S)^2 = O(\varepsilon).$$

This means that f can be $O(\varepsilon)$-approximated by using only Fourier coefficients corresponding to frequencies in $G(\varepsilon/n, t)$. Since by Proposition 29 the size of $R(\varepsilon/n)$ is bounded by k, it further follows that the size of $G(\varepsilon/n, t)$ is bounded by k^t. Thus we can use the algorithm `Find-Variables` to compute a superset R^* of $R(\varepsilon/2n)$ containing only variables in $R(\varepsilon/n)$ in order to get the following learning result.

Theorem 31. *There is a randomized algorithm A such that for each monotone $(\varepsilon/n, k)$-junta f that is ε-close to some Boolean function g with $\sum_{|S|\ge t} \hat{g}(S)^2 \le \varepsilon$, $A(k, t, \varepsilon, \delta, EX(f))$ outputs with probability* $1 - \delta$ *an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n, k^t, \log(1/\delta))$.

Since by Proposition 21, $I(g) \le l$ implies that the Fourier weight of g on the frequencies S with $|S| > l/\varepsilon$ is bounded by ε, we also obtain the following corollary.

Corollary 32. *There is a randomized algorithm A such that for each monotone $(\varepsilon/n, k)$-junta f that is ε-close to some Boolean function g with influence $I(g) \le l$, $A(k, l, \varepsilon, \delta, EX(f))$ outputs with probability* $1 - \delta$ *an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n, k^{l/\varepsilon}, \log(1/\delta))$.

Since for any $\varepsilon > 0$, an m-term DNF is an $(\varepsilon, m\log(m/\varepsilon))$-junta, Corollary 32 implies the following learning result for monotone m-term DNF, which is in fact a consequence of a stronger result from [9].

Corollary 33. [9] *The class of monotone m-term DNF formulas is learnable in time*

$$\mathrm{poly}(n, (m\log(n/\varepsilon))^{\sqrt{m\log(m/\varepsilon)}/\varepsilon}, \log(1/\delta)).$$

Hence, for $m = O((\log n)^2/(\log\log n)^3)$ and constant ε, monotone m-term DNF formulas are learnable in polynomial time.

Servedio [40] used a bound of Mansour [36] to show that any m-term DNF f is ε-close to a Boolean function g satisfying $\sum_{|S|>t} \hat{g}(S)^2 \leq \varepsilon$ for $t = O(\log(m/\varepsilon)\log(1/\varepsilon))$. By Theorem 31, this implies the following exponential improvement of Corollary 33.

Corollary 34. [40] *The class of monotone m-term DNF formulas is learnable in time*

$$\mathrm{poly}(n, (m\log(n/\varepsilon))^{\log(m/\varepsilon)\log(1/\varepsilon)}, \log(1/\delta)).$$

Hence, monotone $O(2^{\sqrt{\log n}})$-term DNF formulas are learnable in polynomial time.

4.4 Monotone functions with bounded influence

Since k-juntas have low influence, it is clear that all functions that are sufficiently close to some k-junta also must have low influence. As shown by Friedgut [14], also the converse is true: any Boolean function f having low influence must be close to a k-junta where k only depends on the influence and not on the arity of f. More precisely, if $I(f) \leq l$ then f is a $(\varepsilon, 2^{O(\varepsilon/l)})$-junta. Very recently, this relationship has been used by O'Donnell and Servedio [39] to demonstrate the applicability of the generalized low-degree algorithm to the class of monotone decision trees.

The proof of Friedgut's result yields the following approximability result for Boolean functions with bounded influence.

Proposition 35. [14] *For any Boolean function f with $I(f) \leq l$,*

$$\sum_{S \notin G(\theta,t)} \hat{f}(S)^2 = O(\varepsilon),$$

where $\theta = (\varepsilon 2^{-l/\varepsilon}/l)^3$ and $t = l/\varepsilon$.

Proof. For a given ε we want to derive a bound on θ such that at most an ε fraction of the weight of the Fourier spectrum of f is outside of the collection $G(\theta, t)$. Let $H = \{S \subseteq [n] \mid S \not\subseteq R(\theta) \wedge |S| < t\}$ and note that $S \notin G(\theta, t)$ implies that either $|S| \geq t$ or $S \in H$. Now it follows by Proposition 21 that

$$\sum_{S \notin G(\theta,t)} \hat{f}(S)^2 \leq \sum_{|S|\geq t} \hat{f}(S)^2 + \sum_{S\in H} \hat{f}(S)^2 \leq \varepsilon + \sum_{S\in H} \hat{f}(S)^2,$$

Hence, it suffices to choose θ small enough such that the Fourier weight of f on H is bounded by $O(\varepsilon)$. Using Beckner-Bonami (cf. inequality (4) in Section 2) and letting $H_j = \{S \subseteq [n] \mid j \in S \wedge |S| < t\}$ it follows for each position j that

$$\sum_{S\in H_j} 2^{-t} \hat{f}(S)^2 \leq \sum_{S\,:\, j\in S} 2^{-|S|} \hat{f}(S)^2 = \|T_{1/\sqrt{2}} f_j\|^2 \leq (\|f_j\|^2)^{4/3} = I_j(f)^{4/3},$$

implying that $\sum_{S\in H_j}\hat{f}(S)^2 \le 2^t I_j(f)^{4/3}$. Note that for each $S \in H$ there is some $j \notin R(\theta)$ such that $S \in H_j$. Hence, summing up and using the bounds $I(f) \le l$ and $I_j(f) \le \theta$ for all $j \notin R(\theta)$ we get

$$\sum_{S\in H}\hat{f}(S)^2 \le \sum_{j\notin R(\theta)}\sum_{S\in H_j}\hat{f}(S)^2 \le \sum_{j\notin R(\theta)} 2^t I_j(f)^{4/3} \le 2^t\theta^{1/3}\sum_j I_j(f) \le 2^t\theta^{1/3} l.$$

Thus, choosing $\theta = (\varepsilon 2^{-t}/l)^3$ yields the desired bound. Since $t = l/\varepsilon$, the size of $R(\theta)$ is bounded by $l/\theta = l^4 2^{3l/\varepsilon}/\varepsilon^3 = 2^{O(l/\varepsilon)}$ and it follows that the size of G is bounded by $2^{O(l/\varepsilon)^2}$. ∎

As an immediate consequence we can state the following learning result.

Theorem 36. *There is a randomized algorithm A such that for each monotone Boolean function f with $I(f) \le l$, $A(l,\varepsilon,\delta,EX(f))$ outputs with probability* $1-\delta$ *an $O(\varepsilon)$-approximation for f in time* $\mathrm{poly}(n, 2^{(l/\varepsilon)^2}, \log(1/\delta))$.

As shown by O'Donnell and Servedio [39], any Boolean function f computable by a decision tree of size m has the property that $\sum_i \hat{f}(i) \le \sqrt{\log m}$. If f is monotone, this implies that $I(f) = \sum_i \hat{f}(i) \le \sqrt{\log m}$.

Corollary 37. [39] *The class of monotone decision trees of size m is learnable in time*

$$\mathrm{poly}(n, m^{(1/\varepsilon)^2}, \log(1/\delta)).$$

Hence, for constant ε, monotone decision trees are learnable in polynomial time.

4.5 Problems

Let us conclude with some interesting problems concerning the learnability of monotone DNF formulas and juntas. As in Section 3.5 we omit the parameters ε and δ for clarity reasons.

Monotone DNF

In the light of O'Donnell and Servedio's efficient learning algorithm for monotone decision trees [39] an interesting question to ask is whether this result can be extended to monotone DNF formulas.

Problem 38. *Are monotone m-term DNF formulas learnable in time* $\mathrm{poly}(n,m)$*?*

It is further shown in [39] that the influence bound $I(f) \le \sqrt{\log m}$ cannot be extended to monotone m-term DNF formulas or even to functions f that are computable by *both* an m-term DNF as well as by an m-clause CNF. Thus, Problem 38 cannot be solved by solely relying on Theorem 36.

The currently best learning algorithm for monotone m-term DNF is the one of Corollary 34 due to Servedio [40] which runs in time $\mathrm{poly}(n, (m \log n)^{\varphi(m)})$ where $\varphi(m) = \log m$. It would be very interesting to improve this result to an algorithm having *fixed parameterized complexity* in the sense of Downey and Fellows [11].

Problem 39. *Are monotone m-term DNF formulas learnable in time* $\mathrm{poly}(n, \varphi(m))$*, where φ does not depend on n?*

Juntas

As we saw in Section 4.2, the analogue of Problem 39 for monotone k-juntas has been solved by Bshouty and Tamon [9] by providing an algorithm running in time $\mathrm{poly}(n, \varphi(k))$ for $\varphi(k) = k^{\sqrt{k}}$ (cf. Corollary 28). An immediate question is whether this can be extended to general juntas.

Problem 40. *Are k-juntas learnable in time* $\mathrm{poly}(n, \varphi(k))$*, where φ does not depend on n?*

We want to emphasize that Problem 40 is a very important question whose answer would have immediate applications to the learnability of DNF formulas (as every DNF formula is close to a junta) as well as to the important area of *feature selection* (cf. [5]). In fact, Mossel et al. [37] consider the problem of learning juntas as the single most important problem in uniform distribution learning. A slightly less ambiguous goal is to learn k-juntas in time $\mathrm{poly}(n, (k \log n)^{\varphi(k)})$. This would imply that juntas with a non-constant number $k(n)$ of relevant variables are learnable in polynomial time.

The currently best learning algorithm for k-juntas is due to Mossel et al. [37] and runs in time $O(n^{\alpha k})$ for some $\alpha < 0.7$. So even an improvement to an $\alpha < 0.5$ would be a significant progress.

Problem 41. *Are k-juntas learnable in time $O(n^{\alpha k})$ for some $\alpha < 0.5$?*

For the special case of symmetric juntas, Mossel et al. [37] used a result of von zur Gathen and Roche [45] to bound the running-time by $n^{\alpha k}$ for some $\alpha < 2/3$. The analogue of Problem 41 for symmetric juntas has been solved by Lipton et al. [34] by providing an algorithm with running-time $O(n^{\alpha k})$ for $\alpha = 3/31$. This bound has been subsequently improved to $O(n^{k/\log k})$ in [29].

References

[1] P. Bartlett, P. Fischer, and K.-U. Hoffgen. Exploiting random walks for learning. In *Proc. 7th Annual ACM Conference on Computational Learning Theory*, pages 318–327, 1994.

[2] W. Beckner. Inequalities in Fourier analysis. *Annals of Mathematics*, 102:159–182, 1975.

[3] M. Ben-Or and N. Linial. Collective coin flipping. In S. Micali, editor, *Randomness and Computation*, pages 91–115. Academic Press, 1989.

[4] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proc. 26th ACM Symposium on Theory of Computing*, pages 253–262, 1994.

[5] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[6] A. Bonami. Etude des coefficients fourier des fonctiones de $l^p(g)$. *Ann. Inst. Fourier*, 20(2):335–402, 1970.

[7] J. H. Bshouty and V. Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002.

[8] N. Bshouty, E. Mossel, R. O'Donnell, and R. Servedio. Learning DNF from random walks. *Journal of Computer and System Sciences*, 71(3):250–265, 2005.

[9] N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

[10] I. Dinur and E. Friedgut. Analytical methods in combinatorics and computer science. Lecure Notes, 2005. URL `http://www.cs.huji.ac.il/~analyt/content.html`.

[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[12] H. Dym and H.P. McKean. *Fourier Series and Integrals*. Academic Press, 1972.

[13] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[14] E. Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):27–36, 1998.

[15] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. In *Proc. Amer. Math. Soc.*, volume 124, pages 2993–3002, 1996.

[16] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st ACM Symposium on Theory of Computing*, 1989.

[17] J. Hastad. *Computational limitations of small-depth circuits*. MIT Press, 1987.

[18] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proc. 36th IEEE Symposium on the Foundations of Computer Science*, pages 538–545, 1995.

[19] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.

[20] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond AC^0. In *Proc. 34th ACM Symposium on Theory of Computing*, pages 776–784, 2002.

[21] J. Jackson and C. Tamon. Fourier analysis in machine learning. COLT 97 Tutorial, 1997. URL `http://learningtheory.org/tutorial/COLT97Fourier.ps`.

[22] J. Kahn, G. Kalai, and N. Linial. The influence of variables on Boolean functions. In *Proc. 29th IEEE Symposium on the Foundations of Computer Science*, pages 68–80, 1988.

[23] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

[24] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[25] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 372–381, 1993.

[26] A. Klivans and R. Servedio. Boosting and hard-core sets. In *Proc. 40th IEEE Symposium on the Foundations of Computer Science*, pages 624–633, 1999.

[27] A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004.

[28] J. Köbler and W. Lindner. A general dimension for approximately learning Boolean functions. In *Proc. 13th International Workshop on Algorithmic Learning Theory*, pages 139–148, 2002.

[29] M. Kolountzakis, E. Markakis, and A. Mehta. Learning symmetric juntas in time $n^{o(k)}$. Interface between Harmonic Analysis and Number Theory, 2005.

[30] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SICOMP*, 22(6):1331–1348, 1993.

[31] W. Lindner. Learning DNF by statistical and proper distance queries under the uniform distribution. In *Proc. 16th International Workshop on Algorithmic Learning Theory*, pages 198–210, 2005.

[32] N. Linial. Harmonic analysis and combinatorial applications. Lecure Notes, 2005. URL `http://www.cs.huji.ac.il/~nati/PAPERS/uw/`.

[33] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[34] R. Lipton, E. Markakis, A. Mehta, and N.K. Vishnoi. On the Fourier spectrum of symmetric Boolean functions with applications to learning symmetric juntas. In *Proc. 20th Annual IEEE Conference on Computational Complexity*, pages 112 – 119, 2005.

[35] Y. Mansour. Learning Boolean functions via the Fourier transform. In V.P. Roychodhury, K-Y. Siu, and A. Orlitsky, editors, *Theoretical Advances in Neural Computation and Learning*, pages 391–424. Kluwer Academic Publishers, 1994.

[36] Y. Mansour. An O($n^{\log\log n}$) learning algorithm for DNF under the uniform distribution. *Journal of Computer and System Sciences*, 50:543–550, 1995.

[37] E. Mossel, R. O'Donnell, and R. Servedio. Learning juntas. In *Proc. 35th Ann. ACM Symp. on the Theory of Computing, 2003.*, 2003.

[38] R. O'Donnell. *Computational Applications of Noise Sensitivity*. PhD thesis, Massachutes Institute of Technology, 2003.

[39] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. In *Proc. 21st Annual IEEE Conference on Computational Complexity*, 2006.

[40] R. Servedio. On learning monotone DNF under product distributions. *Inf. Comput.*, 193(1):57–74, 2004.

[41] R. Shapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[42] D. Stefankovic. Fourier transforms in computer science. Master's thesis, University of Chicago, 2002.

[43] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[44] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proc. 3rd Annual ACM Conference on Computational Learning Theory*, pages 314–326, 1990.

[45] J. von zur Gathen and J.R. Roche. Polynomials with two values. *Combinatorica*, 17 (3):345–362, 1997.

[46] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.

The Concurrency Column

BY

Luca Aceto

BRICS, Department of Computer Science
Aalborg University, 9220 Aalborg Ø, Denmark

Dept. of Computer Science, School of Science and Engineering
Reykjavik University, 103 Reykjavik, Iceland

luca@{cs.auc.dk,ru.is}, http://www.cs.auc.dk/~luca/BEATCS

In this installment of the Concurrency Column, Patricia Bouyer and Fabrice Chevalier offer a survey of results on the control of timed and hybrid systems. This is a very active area of research that is both the source of interesting theoretical problems and, in light of the ubiquituous presence of embedded systems, has the potential for wide practical applications.

I hope that readers of this survey will enjoy it and will be enticed to work on this difficult, but rewarding, area. I have no doubt that Fabrice and Patricia will continue to be at the forefront of the developments in this field of concurrency theory, and look forward to their future work on this topic.

On the Control of Timed and Hybrid Systems[*]

Patricia Bouyer and Fabrice Chevalier
LSV, CNRS & ENS de Cachan, France
{bouyer,chevalie}@lsv.ens-cachan.fr

Abstract

In this paper, we survey some of the results which have been obtained the last ten years on the control of hybrid and timed systems.

[*]Work partly supported by ACI Cortos, a program of the French ministry of research.

1 Introduction

Timed and hybrid systems. Timed automata are a well-established and widely used model for representing real-time systems. Since their definition by Alur & Dill in the 90's [3], many works have investigated this model, both from a theoretical and an algorithmic point of view. Several tools have been developed for model-checking timed automata [12, 27] and have been used for verifying industrial case studies.

Hybrid automata can be seen as an extension of timed automata, but they have in fact been defined and studied roughly at the same time [19]. Though most verification problems for this model are undecidable, many works are devoted to its study, by providing for instance decidable subclasses or approximation algorithms.

Control of timed and hybrid systems. To deal with *open* systems, *i.e.* systems interacting with an environment (which is the case for most embedded systems), model-checking may not be sufficient, and we better need to *control* (or guide) the system so that it satisfies the specification, whatever the environment does. More formally, the *control problem* asks, given a system $\mathcal{S}$ and a specification φ, whether there exists a controller C such that $\mathcal{S}$ guided by C satisfies φ (see [30, 31] for initial papers on the control of discrete event systems). Since the mid-90's, work on the control of real-time systems is flourishing (see all references mentioned along this paper).

Positioning of this survey. In the literature, several formalisations of the control problem have been proposed, some of them are based on a two-player game formulation where the "controller" plays against the "environment". Mostly, results in one framework can be translated into another framework, but care needs however to be taken. In this paper we focus on "control games", an asymmetric formulation where the "environment" player is somehow more powerful than the "controller" player (the controller has to fix his strategy, and this strategy has to be winning whatever the environment does). This is a framework related to the one considered for instance in [15, 16, 9, 8]. A very close framework based on control maps is considered also in [4, 21]. In the literature, we can find more "concurrent" (and thus symmetric) formulations where both players independently choose a delay and an action to be performed after that delay, and action corresponding to the shortest delay is done [1], or a joint play is obtained with these two choices [20].

Outline of the paper. In Section 2, we will present basic notions on timed systems. In Section 3, we define the problem of control for timed systems we will consider, and explain how the basic safety and reachability control problems can be solved for the class of timed automata. We also shortly explain for which other external specifications languages (like timed automata, temporal logics, etc.) we can solve the control problem. In the previous section, there is an (implicit) hypothesis that the controller has complete information on the system. This is a restrictive hypothesis, which we relax in Section 4: under a partial observability assumption, the control problem becomes very difficult, and even the simplest control problems (reachability and safety) become undecidable, except if we restrict the resources of the controller. In Section 5, we briefly give some results concerning the control of two subclasses of hybrid systems, namely rectangular hybrid automata and o-minimal hybrid automata. We finally give some conclusions and mention some current challenges in Section 6.

2 Timed Automata

2.1 Preliminaries

Timed words. Let $\mathbb{R}_{\geq 0}$ be the set of non-negative reals and $\mathbb{Q}_{\geq 0}$ be the set of non-negative rational numbers. Let Σ be a finite alphabet. A *timed word* over Σ is a (possibly infinite) sequence $\sigma = (a_1, \tau_1)(a_2, \tau_2)\ldots$ over $\Sigma \times \mathbb{R}_{\geq 0}$ such that $\tau_i \leq \tau_{i+1}$ for every $1 \leq i < |\sigma|$ (where $|\sigma|$ denotes the (possibly infinite) length of σ). If σ is infinite, it is *non-Zeno* if the sequence $\{\tau_i\}_{i \in \mathbb{N}}$ is unbounded.

Clocks, operations on clocks. We consider a finite set X of variables, called *clocks*. A *valuation* over X is a mapping $v : X \to \mathbb{R}_{\geq 0}$ which assigns to each clock a time value in $\mathbb{R}_{\geq 0}$. We note $\mathcal{V}_X$ (or $\mathcal{V}$ when it is clear from the context) the set of valuations over X. We use $\vec{0}$ to denote the valuation which sets each clock $x \in X$ to 0. If $t \in \mathbb{R}_{\geq 0}$, the valuation $v + t$ is defined as $(v + t)(x) = v(x) + t$ for all $x \in X$. If Y is a subset of X, the valuation $v[Y \leftarrow 0]$ is defined as: for each clock x, $(v[Y \leftarrow 0])(x) = 0$ if $x \in Y$ and $(v[Y \leftarrow 0])(x) = v(x)$ otherwise.

The set of *(clock) constraints* (or *guards*) over a set of clocks X, denoted $\mathcal{G}(X)$, is given by the syntax "$g \ ::= \ x \sim c \ \mid \ g \wedge g$" where $x \in X$, $c \in \mathbb{Q}_{\geq 0}$ and $\sim$ is one of the comparison operators $<, \leq, =, \geq,$ or $>$. We write $v \models g$ if the valuation v satisfies the clock constraint g, and is given by $v \models x \sim c$ if $v(x) \sim c$ and $v \models g_1 \wedge g_2$ if $v \models g_1$ and $v \models g_2$. The set of valuations over X which satisfy a guard $g \in \mathcal{G}(X)$ is denoted by $[\![g]\!]_X$, or simply $[\![g]\!]$ when X is clear from the context.

Timed automata [2, 3]. A *timed automaton* (TA for short) is a 6-tuple $\mathcal{A} = (\Sigma, X, Q, q_0, \longrightarrow, Inv)$ where Σ is a finite alphabet of actions, X is a finite set of clocks, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\longrightarrow \subseteq Q \times \mathcal{G}(X) \times$

$\Sigma \times 2^X \times Q$ is a finite set of transitions, and $Inv : Q \rightarrow \mathcal{G}(X)$ assigns an invariant to every state. The timed automaton $\mathcal{A}$ is said to be *deterministic* if (q, g_1, a, Y_1, q_1), $(q, g_2, a, Y_2, q_2) \in \longrightarrow$ implies $[\![g_1]\!] \cap [\![g_2]\!] = \emptyset$. The class of deterministic timed automata is denoted DTA.

A configuration of $\mathcal{A}$ is a pair (q, v) where $q \in Q$ and v si a valuation over the set of clocks X. The timed automaton $\mathcal{A}$ defines a timed transition system, made of timed and discrete transitions. A *timed transition* in $\mathcal{A}$ is a transition of the form $(q, v) \xrightarrow{t} (q, v+t)$ where for every $0 \leq t' \leq t$, $v+t' \models Inv(q)$. A *discrete transition* in $\mathcal{A}$ is a transition of the form $(q, v) \xrightarrow{a} (q', v')$ when there exists $(q, g, a, Y, q') \in \longrightarrow$ such that $v \models g \wedge Inv(q)$, $v' = v[Y \leftarrow 0]$, and $v' \models Inv(q')$.

A run of $\mathcal{A}$ starting in (q_1, v_1) is a finite or infinite sequence of transitions $\rho = (q_1, v_1) \xrightarrow{t_1} (q'_1, v'_1) \xrightarrow{a_1} (q_2, v_2) \xrightarrow{t_2} \cdots$, which alternates between timed and discrete transitions. We will sometimes equivalently write it $\rho = (q_1, v_1) \xrightarrow{a_1, t_1} (q_2, v_2) \xrightarrow{a_2, t_2} \cdots$. We write $tw(\rho)$ the timed word associated to the run ρ, that is the finite or infinite sequence $(a_1, t_1)(a_2, t_2) \cdots$. If ρ is finite and ends in (q_n, v_n) we write $\mathsf{Last}(\rho) = (q_n, v_n)$. We write $\mathsf{Runs}(\mathcal{A})$ (resp. $\mathsf{Runs}_f(\mathcal{A})$) the set of runs (resp. finite runs) in $\mathcal{A}$. We say that $a \in \Sigma$ is *enabled* in (q, v) if there exists (q', v') such that $(q, v) \xrightarrow{a} (q', v')$. We say that λ (a symbol used to express time elapsing) is enabled in (q, v) if there exists (q', v') and $t > 0$ such that $(q, v) \xrightarrow{t} (q', v')$. We write $\mathsf{Enabled}((q, v))$ the subset of $\Sigma \cup \{\lambda\}$ of actions or λ enabled in (q, v).

The scrupulous reader may have noticed that we did not include an accepting condition to timed automata. This is for simplicity, but when necessary (for instance in Subsection 3.3), we will assume that there is an accepting condition in timed automata, for instance a set F of accepting states for finite words, or a set of R of repeated states for defining a Büchi condition for infinite words (and we could consider more general accepting conditions like Muller, Rabin or parity conditions). Acceptance of a timed word is defined classically and we write $L(\mathcal{A})$ the set of timed words accepted by $\mathcal{A}$.

2.2 Region automaton

The *region automaton* $\mathcal{R}(\mathcal{A})$ of a timed automaton $\mathcal{A}$ is a finite automaton, which abstracts timed behaviours of a timed automaton into (untimed) behaviours. This abstraction has been proposed by Alur & Dill in [2, 3] for deciding language emptiness of timed automata. Indeed, each state of the region automaton is a class[1] of an equivalence relation $\equiv_{\mathcal{A}}$ (of finite index) over configurations which is a *time-abstract bisimulation*: if $(q_1, v_1) \equiv_{\mathcal{A}} (q_2, v_2)$ and $(q_1, v_1) \xrightarrow{a} (q'_1, v'_1)$ with $a \in \Sigma$, then there exists $(q_2, v_2) \xrightarrow{a} (q'_2, v'_2)$ such that $(q'_1, v'_1) \equiv_{\mathcal{A}} (q'_2, v'_2)$; if $(q_1, v_1) \equiv_{\mathcal{A}}$

[1]called a *region*.

(q_2, v_2) and $(q_1, v_1) \xrightarrow{t_1} (q'_1, v'_1)$ for some $t_1 > 0$, then there exists $t_2 > 0$ such that $(q_2, v_2) \xrightarrow{t_2} (q'_2, v'_2)$ and $(q'_1, v'_1) \equiv_{\mathcal{A}} (q'_2, v'_2)$. We will not enter into more details and better refer to [3] for the description of the region automaton construction. However it is worth mentioning that this construction is the core of numerous decidability results for the model of timed automata.

3 Control of Timed Automata

3.1 The control problem

Several formulations of the control problem can be found in the literature. It is sometimes defined as a game between a *controller* and a (possibly nasty) *environment*. The formulation we consider in this paper is an asymmetric game where the environment is more powerful than the controller.

We define the control problem for a timed system given as a timed automaton, but it could be easily generalized to other kinds of systems. A *plant* is a timed automaton $\mathcal{A} = (\Sigma, X, Q, q_0, \longrightarrow, Inv)$ where the alphabet Σ is partitioned into two subsets Σ_c and Σ_u corresponding respectively to controllable and uncontrollable actions. Intuitively, the controller will be able to perform controllable actions, whereas the environment will be able to perform uncontrollable actions.

A *controller strategy* (or simply a *strategy*) is a partial function f from the set $\mathsf{Runs}_f(\mathcal{A})$ to $2^{\Sigma_c \cup \{\lambda\}}$ such that for every finite run ρ such that $f(\rho)$ is defined, $f(\rho) \subseteq \mathsf{Enabled}(\mathsf{Last}(\rho))$ and $f(\rho) \neq \emptyset$. The strategy f tells precisely what the controller has to do: if $f(\rho) \subseteq \Sigma_c$, then a discrete controllable action of $f(\rho)$ has to be done, whereas if $\lambda \in f(\rho)$, then it is possible to delay (or a discrete action of $f(\rho)$ can be done as well). Note that a strategy is not deterministic as it may propose a set of actions in $\Sigma_c \cup \{\lambda\}$. In the literature, several other notions of strategies can be found.

A run $\rho = (q_1, v_1) \xrightarrow{a_1, t_1} (q_2, v_2) \xrightarrow{a_2, t_2} \cdots$ is said *compatible with a strategy* f if for all i, writing $\rho_i = (q_1, v_1) \xrightarrow{a_1, t_1} \cdots \xrightarrow{a_{i-1}, t_{i-1}} (q_i, v_i)$ we have that for all $0 \leq t' \leq t_i$, $\lambda \in f(\rho_i \xrightarrow{t'} (q_i, v_i + t'))$, and if $a_i \in \Sigma_c$, $a_i \in f(\rho_i \xrightarrow{t_i} (q_i, v_i + t_i))$. A *maximal run w.r.t a strategy* f (or simply *maximal run* when f is clear from the context) is either an infinite run or a run which satisfies: for all $t \geq 0$, if $\rho \xrightarrow{t} (q, v)$ then $\lambda \in f(\rho \xrightarrow{t} (q, v))$.

A strategy f is said *realizable* if for every finite run ρ such that $\lambda \in f(\rho)$, there exists $\delta > 0$ such that for all $0 \leq t < \delta$ if $\rho \xrightarrow{t} (q, v)$ then $\lambda \in f(\rho \xrightarrow{t} (q, v))$. This notion of realizability has been defined in [8] to avoid strategies which have no compatible run, but is relevant for most control problems.

Given a plant $\mathcal{A}$, a *specification* $\mathcal{S}$ for $\mathcal{A}$ is a subset of $\mathsf{Runs}(\mathcal{A})$. Intuitively it corresponds to the desired behaviours of the plant. In the following, we will consider special cases of specifications such as reachability objectives, safety objectives, or external specifications.

If $\mathcal{A}$ is a plant and $\mathcal{S}$ a specification for $\mathcal{A}$, a strategy f is *winning from a configuration (q,v)* if all maximal runs starting in (q, v) compatible with f are in the set $\mathcal{S}$. A configuration (q, v) is said *winning* if there is a realizable winning strategy from (q, v).

We can now formally define the control problem:

Problem (Control problem). Given a plant $\mathcal{A}$, a specification $\mathcal{S}$ and an initial configuration (q, v), determine if there is a realizable winning strategy from (q, v).

A natural further question is to (effectively) construct winning strategies, if one exists. In the rest of the paper, we will not devote much time to that subject, though it is sometimes a non-trivial one.

3.2 Reachability and safety control

In this subsection, we consider two natural types of specifications: reachability and safety properties. Intuitively, for a reachability specification, the goal for the controller is to reach a set of good states, whereas for a safety specification, the controller has to avoid a set of bad states.

Let $\mathcal{A} = (\Sigma, X, Q, q_0, \longrightarrow, Inv)$ be a plant, and let **Good** and **Bad** be two subsets of Q, we define the two following specifications:

$$\begin{cases} \mathcal{S}_{\mathbf{Good}} = \{\rho = (q_1, v_1) \xrightarrow{a_1,t_1} (q_2, v_2) \xrightarrow{a_2,t_2} \cdots \mid \exists i\ q_i \in \mathbf{Good}\} \\ \mathcal{S}_{\mathbf{Bad}} = \{\rho = (q_1, v_1) \xrightarrow{a_1,t_1} (q_2, v_2) \xrightarrow{a_2,t_2} \cdots \mid \forall i\ q_i \notin \mathbf{Bad}\} \end{cases}$$

Problem (Reachability control problem). Given a plant $\mathcal{A}$, a set of states **Good** and an initial configuration (q, v), determine if there is a realizable winning strategy from (q, v) for the specification $\mathcal{S}_{\mathbf{Good}}$.

Problem (Safety control problem). Given a plant $\mathcal{A}$, a set of states **Bad** and an initial configuration (q, v), determine if there is a realizable winning strategy from (q, v) for the specification $\mathcal{S}_{\mathbf{Bad}}$.

We now explain how to solve reachability and safety control problems in the timed framework [4]. A way of computing winning states for these specifications is to compute the *attractor* of goal states by iterating a *controllable predecessor* operator. This is a classical method in the theory of classical (untimed) games for computing winning states [18].

Let $\mathcal{A}$ be a plant. We define the controllable and uncontrollable discrete predecessors of a set of configurations $A \subseteq Q \times \mathcal{V}$ as follows:

$$\mathsf{cPred}(A) = \left\{ (q,v) \in Q \times \mathcal{V} \;\middle|\; \begin{array}{l} \exists c \in \Sigma_c,\ \text{c is enabled in } (q,v), \\ \text{and } \forall (q',v') \in Q \times \mathcal{V}, \\ (q,v) \xrightarrow{c} (q',v') \Rightarrow (q',v') \in A \end{array} \right\}$$

$$\mathsf{uPred}(A) = \left\{ (q,v) \in Q \times \mathcal{V} \;\middle|\; \begin{array}{l} \exists u \in \Sigma_u,\ \exists (q',v') \in Q \times \mathcal{V} \text{ s.t.} \\ (q,v) \xrightarrow{u} (q',v') \text{ and } (q',v') \in A \end{array} \right\}$$

The set $\mathsf{cPred}(A)$ is the set of configurations from which we can enforce a configuration of A by doing a controllable action. The set $\mathsf{uPred}(A)$ is the set of configurations from which the environment can do an uncontrollable action which leads to a configuration in A. In the untimed (finite-state) framework, these two operators are sufficient to define the set of configurations from which we can enforce the set A in one step, this is $\pi(A) = \mathsf{cPred}(A) \setminus \mathsf{uPred}(\overline{A})$. Then, starting from the set of configurations which are good, and iterating the operator π, we can compute the set of configurations from which we can enforce the set of states **Good**. Similarly (or dually), we can also compute the set of configurations from which we can avoid the set of states **Bad**. Note that in the untimed framework, these two problems are dual.

In the timed framework, these two discrete controllable and uncontrollable predecessors are not sufficient, and we need to define a time controllable predecessor operator of a set A of configurations: a state (q,v) is in $\pi(A)$ if and only if (1) it is possible to let t time units elapse for some $t \geq 0$ and use a controllable action to reach A and no uncontrollable action played before or at t leads outside A; or (2) A can be reached by just letting time elapse and no uncontrollable action leads outside A. Formally the operator π is defined as follows:

$$\pi(A) = \left\{ (q,v) \in Q \times \mathcal{V} \;\middle|\; \begin{array}{l} \quad \exists t\ (q,v) \xrightarrow{t} (q',v'),\ (q',v') \in \mathsf{cPred}(A), \\ \quad \text{and } \mathsf{Post}_{[0,t]}(q,v) \cap \mathsf{uPred}(\overline{A}) = \emptyset; \\ \text{or } \exists t\ \mathsf{Post}_{[t,+\infty)}(q,v) \subseteq A, \\ \quad \text{and } \mathsf{Post}_{[0,+\infty)}(q,v) \cap \mathsf{uPred}(\overline{A}) = \emptyset \end{array} \right\}$$

where $\mathsf{Post}_I(q,v) = \{(q, v+t) \in Inv(q) \mid t \in I\}$ with I interval of $\mathbb{R}_{\geq 0}$.

We now contruct an increasing and a decreasing version of π to solve respectively reachability and safety control: $\pi_{\text{reach}}(A) = A \cup \pi(A)$ and $\pi_{\text{safe}}(A) = A \cap \pi(A)$. We note $\pi^*_{\text{reach}}(\mathbf{Good}) = \lim_{k \to +\infty} \pi^k_{\text{reach}}(\mathbf{Good})$ and $\pi^*_{\text{safe}}(\mathbf{Bad}) = \lim_{k \to +\infty} \pi^k_{\text{safe}}(\mathbf{Bad})$.

Proposition 1 ([4]). *Let $\mathcal{A}$ be a plant, let* **Good** *(resp.* **Bad***) a set of good (resp. bad) states. The set of winning states for the reachability specification $\mathcal{S}_{\textbf{Good}}$ is $\pi^*_{reach}(\textbf{Good})$, whereas the set of winning states for the safety specification $\mathcal{S}_{\textbf{Bad}}$ is $\pi^*_{safe}(\textbf{Bad})$.*

From this characterization of the set of winning states, we can deduce an algorithm for computing the set of winning states for reachability and safety specifications: indeed it is easy to show that when A is a union of regions (see Subsection 2.2), then $\pi(A)$ is a union of region and can be effectively computed. So the fixed points $\pi^*_{\text{reach}}(\textbf{Good})$ and $\pi^*_{\text{safe}}(\textbf{Bad})$ of π_{reach} and π_{safe} respectively can be computed after a finite number of iterations: the set of winning states can thus be effectively computed.

Example. We develop a short example to illustrate how the controllable predecessor operator acts. We assume $\Sigma_c = \{c\}$, $\Sigma_u = \{u\}$, and the following plant with **Good** $= \{q_3\}$:

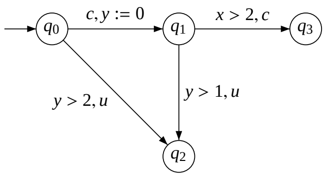


Figure 1: A plant to control

The computation of the fixed point is the following (we note G the set of good configurations we want to enforce, *i.e.* $G = \{(q_3, (x, y)) \mid x \geq 0 \text{ and } y \geq 0\}$):

$$\begin{cases} \pi_{reach}(G) & = & G \cup \{(q_1, (x,y)) \mid y \leq 1 \text{ and } x - y > 1\} \\ \pi^2_{reach}(G) & = & \pi_{reach}(G) \cup \{(q_0, (x,y)) \mid y \leq 2 \text{ and } y - x < 1\} \\ \pi^*_{reach}(G) & = & \pi^2_{reach}(G) \end{cases}$$

The set of states from which the controller can enforce state q_3 is thus $\pi^*_{reach}(G)$, as described above. Thus, the initial configuration $(q_0, \vec{0})$ is winning. □

From the above fixed point computation, we can extract winning strategies (like in the untimed framework [18]). However, strategies which are extracted that way may not be realizable: this is for instance the case in the previous example where the extracted strategy says "λ" on $(q_1, (x = 2, y))$ (with $y < 1$) and "c" on $(q_1, (x > 2, y))$ (with $y \leq 1$). It is then necessary to make the strategy realizable [8].[2]

[2]Note that the new realizable strategy might not be expressible as a timed automaton, even if the former was definable as a timed automaton.

Moreover by adapting the PSPACE-hardness proof of the reachability problem in timed automata, one can show that reachability and safety control of timed automata are EXPTIME-hard [21]. As the above-mentioned fixed points can be computed in exponential time (because there is an exponential number of regions [3]), we have the following theorem:

Theorem 2. *Reachability and safety control problems in timed automata are EXPTIME-complete.*

3.3 External specifications

In the previous subsection we have investigated the control of timed automata for reachability and safety specifications; these are internal specifications as the winning condition is given on states of the plant itself. The control problem for various external specifications has also been considered.

External specifications given as timed automata. A natural external specification is one given by a timed automaton. Let $\mathcal{A}$ be a plant and $\mathcal{B}$ a timed automaton with an accepting condition for infinite words. We can see $\mathcal{B}$ as a specification for the plant $\mathcal{A}$ by defining $\mathcal{S}_{\mathcal{B}} = \{\rho \in \mathsf{Runs}(\mathcal{A}) \mid tw(\rho) \in L(\mathcal{B})\}$.[3]

For this kind of specifications, the following decidability results have been proved:

Theorem 3 ([15]). *The control problem for specifications given as timed automata over infinite words is undecidable. The control problem for specifications given as deterministic timed automata over infinite words is decidable and can be solved in 2EXPTIME.*

The control problem for specifications given as (non-deterministic) timed automata is undecidable as inclusion of timed automata can be reduced to that problem. However a natural assumption when synthesizing timed systems is to restrict the power of the controller by looking for a controller which is a timed automaton using a fixed number of clocks and a fixed set of constants. In that case, we say that we fix the *granularity* of the controller. This assumption is done for instance for solving the satisfiability problem of the logic L_ν [26] or for various control problems [15, 9, 6].

A granularity is a triple $\mu = (k, m, K)$ where k, m, K are integers. A timed automaton is said of granularity μ if it uses k clocks and constants mentioned in its constraints are of the form $\frac{i}{m}$ with $0 \le i \le K$. Let $\mathcal{A}$ be a plant, a strategy f for

[3]Note that negative specifications like $\mathcal{S}^{-}_{\mathcal{B}} = \{\rho \in \mathsf{Runs}(\mathcal{A}) \mid tw(\rho) \notin L(\mathcal{B})\}$ have also been considered in [15].

$\mathcal{A}$ is said *μ-granular* if it can be represented by a deterministic timed automaton $\mathcal{B}$ of granularity μ. Formally if ρ is a run of $\mathcal{A}$ compatible with strategy f and ρ' is the unique corresponding run in $\mathcal{B}$ then $f(\rho) = \mathsf{Enabled}(\mathsf{Last}(\rho')) \cap (\Sigma_c \cup \{\lambda\})$.

The DTA$_\mu$-control problem for a specification given as a timed automaton over infinite words then asks, given a plant $\mathcal{A}$, a granularity μ, a timed automata over infinite words $\mathcal{B}$ and an initial configuration (q, v), whether there is a μ-granular realizable winning strategy from (q, v) for the specification $\mathcal{S}_\mathcal{B}$.

Theorem 4 ([15]). *The DTA$_\mu$-control problem for specifications given as timed automata over infinite words is 2EXPTIME-complete.*

This result can be proved by reducing this control problem to a parity game over a finite automaton (this automaton is obtained using some region construction).

External specifications given as formulas of (timed) temporal logics. Various (timed) temporal logics have been used as specification languages for control problems. We first focus on linear-time (timed) temporal logics. If ϕ is a formula of some linear-time (timed) temporal logic, we write $\mathcal{S}_\phi = \{\rho \in \mathsf{Runs}(\mathcal{A}) \mid tw(\rho) \in L(\phi)\}$ for the specification defined by ϕ (where $L(\phi)$ is the set of models of ϕ).

We assume the reader is familiar with the linear-time temporal logic LTL, and better refer to [29] for the definition of this logic. The control problem for specifications given as LTL formulas has been studied in [16]:

Theorem 5 ([16]). *The control problem over timed automata for specifications given as* LTL *formulas is 2EXPTIME-complete.*

The technique used to prove this result relies on the construction of a tree automaton (based on regions) which accepts all winning strategies.

The logic MTL [23] is a timed extension of LTL with time constraints on modalities. For instance, in this logic, we can write bounded response time properties like $\Box(p \rightarrow \Diamond_{\leq 5} q)$ which says that every time p holds, q has to hold within a time window of 5 time units. Similarly to specifications given as timed automata, the control problem is undecidable for specifications given as MTL formulas but becomes decidable if we restrict to μ-granular strategies. However in this case the complexity is much higher:

Theorem 6 ([6]). *The control problem for* MTL*-specifications is undecidable. The DTA$_\mu$-control problem for* MTL*-specifications is decidable and has non-primitive recursive complexity.*

It is worth reminding that already model-checking of MTL is non-primitive recursive [28], and that the halting problem of a lossy-channel system can be simulated by an MTL model-checking problem. Roughly, using the power of the controller, we can simulate with a control problem the halting problem of a perfect channel system, which leads to undecidability. The decidability when fixing the granularity of the controller is not based on regions, but on a better-quasi-order.

Branching-time (timed) logics have also been considered in the literature. For lack of space, and because it does not perfectly fit our definition of specifications,, we do not enter into details and better point out the corresponding references: [16, 17]. Finally, games for specifications given as formulas of the μ-calculus have been studied in the rather different framework of symmetric timed games [1].

4 Partial Observability

In the previous section we have supposed that the controller has perfect information on the plant: at any time, the controller knows in which state the plant is. In this section, we consider the more general problem of control under partial observability: the controller has only partial information about the plant and should control it whatever is the behaviour (observable or not) of the environment.

In this section we suppose that Σ_u is partionned into two subsets Σ_u^{obs} and Σ_u^{unobs}. The actions of Σ_u^{obs} are uncontrollable but observable, whereas the actions of Σ_u^{unobs} are not observable (and thus not controllable). The controllable actions of Σ_c remains observable.

We define a consistent strategy for the controller as a strategy which depends only on observations of actions in Σ_c and Σ_u^{obs}. Let π_{obs} be the projection of timed words on $\Sigma_c \cup \Sigma_u^{obs}$ (which is defined in a natural way by erasing actions in Σ_u^{unobs} and their corresponding dates), a strategy is said *consistent* if for all runs ρ and ρ' such that $\pi_{obs}(tw(\rho)) = \pi_{obs}(tw(\rho'))$, then $f(\rho)$ and $f(\rho')$ are simultaneously (un-)defined, and when they are defined, $f(\rho) = f(\rho')$.

The control problem under partial observability is thus the following:

Problem (Control problem under partial observability). Given a plant $\mathcal{A}$, a specification $\mathcal{S}$ and an initial configuration (q, v), determine if there is a consistent realizable winning strategy from (q, v).

If we consider specifications given as timed automata, the control under partial observability is undecidable even for deterministic specifications. On the other side, the decidability proof for the DTA$_\mu$ control problem can be extended to partial observability:

Theorem 7 ([9]). *The control problem under partial observability for specifications given as deterministic timed automata over infinite words is undecid-*

able. The control problem under partial observability for specifications given as deterministic timed automata over infinite words is decidable and 2EXPTIME-complete.

Indeed, techniques used in [9] can be easily extended to show that reachability control under partial observability is undecidable. However it cannot be applied to safety control under partial observability. We present here an original construction which shows that safety control under partial observability is undecidable when we consider non-Zeno controllers, *i.e.* controllers which generate only non-Zeno behaviours (the fact that the controller has to be non-Zeno will be encoded in the specification).

Given a plant $\mathcal{A}$ and a set **Bad** of bad states we consider the specification $\mathcal{S}_{\mathbf{Bad}}^{\neg Zeno} = \mathcal{S}_{\mathbf{Bad}} \cap \{\rho \mid tw(\rho) \text{ is non-Zeno}\}$.[4] The safety control problem under partial observability for non-Zeno controllers is given a plant $\mathcal{A}$, a set of states **Bad** and an initial configuration (q, v), determine if there is a realizable winning strategy from (q, v) for the specification $\mathcal{S}_{\mathbf{Bad}}^{\neg Zeno}$.

Theorem 8. *The safety control problem under partial observability for non-Zeno controllers is undecidable.*

The proof consists in reducing the halting problem of a 2-counter machine and uses roughly the same encoding as the undecidability of the universality of timed automata [3]. A configuration of a 2-counter machine with n states is encoded by a timed word over the alphabet $\{b_1, \cdots, b_n, a_1, a_2, a_3\}$. We will encode the first configuration of the 2-counter machine within the time interval $[0, 1[$, and the time interval $[i, i+1[$ will contain the encoding of the i^{th} configuration of the execution of the 2-counter machine. If the 2-counter machine is in state j and counter values are c and d then the corresponding timed word should contain the actions $b_j a_1^c a_2^d a_3^e$ between time i and $i+1$ (the use of a_3 is explained later). To express for example that the value of counter 1 does not change between the i^{th} and the $(i+1)^{\text{th}}$ configurations we require that every a_1 of the interval $[i, i+1[$ has a matching a_1 one time unit later and that there is no a_1 in $[i+1, i+2[$ without a matching a_1 one time unit earlier. Similarly, to express that the first counter has decreased by one, we check that all a_1's apart the last one in the interval $[i, i+1[$ has a matching one time unit later in $[i+1, i+2[$, and every a_1 in the interval $[i+1, i+2[$ is matched one time unit before by another a_1.

The use of action a_3 is as follows: we require that the first configuration is encoded by $b_1 a_3^e$ for some $e \in \mathbb{N}$; then, after each configuration the number of a_3 must be decreased by one.

The plant $\mathcal{A}$ is the universal timed automaton. The set of controllable actions is $\Sigma_c = \{b_1, \cdots, b_n, a_1, a_2, a_3\}$: the controller will play an encoding of the execution

[4]recall that $\mathcal{S}_{\mathbf{Bad}}$ is the set of runs which avoid **Bad**.

of the 2-counter machine freely. We use a single unobservable action (we take $\Sigma_u^{obs} = \emptyset$ and $\Sigma_u^{unobs} = \{u\}$) which can be used by the environment to check whether the encoding played by the controller is correct. Non-deterministically and in an unobservable way, the environment launches a *test* to check that the controller has simulated the 2-counter machine correctly (that is it checks that all actions are matched correctly one time unit later or earlier, as explained above).

If the environment discovers that the encoding is incorrect it goes to a bad state and the controller loses. So the controller has to play a correct encoding for winning. Moreover if the number of a_3 played during an interval $[i, i+1[$ is null and the goal state of the 2-counter machine has not been reached, then the plant also goes into a bad state. So to control the plant properly, during the first time unit the controller must play at least n a_3 where n is the number of steps needed to reach q. Then it just has to play a correct encoding leading to q to win the control game.

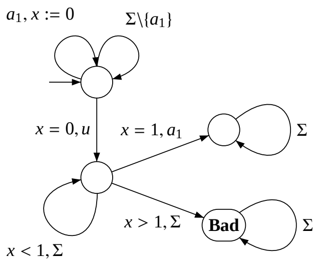


Figure 2: Environment checking that every a_1 is matched one t.u. later by an a_1

On Figure 2, we give an example of how the environment can check that every a_1 is matched one time unit later by another a_1. Note that this construction works only because u is not observable: as the controller cannot know when it is played it has to always simulate correctly the 2-counter machine.

Note that this proof shows that both control by general strategies and control by strategies defined by DTA (with non-fixed granularity) are undecidable. In fact to show that control under partial observability by general strategies a slighty simpler proof not involving a_3 actions can be done.

Recently, another framework for control under partial observability has been developed [14], where observations are based on the visited states. Until now, this work applies only to finite-state automata, and discrete-time timed (and even hybrid) automata.

5 Control of Hybrid Systems

Hybrid automata are an extension of timed automata in which variables are not clocks but are more general: they can for instance follow rules given by differential equations. This model, though immediately undecidable, is widely used and studied (this is the topic of the conference HSCC which takes place every year since 1998). The literature on that subject is substantial, we can thus not give an overview of all results on that subject, we thus select two kinds of models for which decidability results can be obtained.

5.1 Rectangular hybrid automata

The model of rectangular hybrid automata is an extension of the model of timed automata. It thus extends finite automata with real-valued variables, the enabling condition for each discrete move is a cartesian product of intervals, and the first derivative of each continuous variable is bounded by constants from below and above. Checking reachability properties in that model is undecidable unless we assume that they are initialized [22], which means that every transition changing the slope of a variable has to reset it.

In that context, the control problem for safety specifications is undecidable for rectangular timed automata [22]. Several simpler control problems have thus been considered.

Sampling control. In this framework, the controller performs actions every time unit (or every δ if δ is the sampling rate of the controller). When the sampling rate is known in advance, the safety sampling control problem for rectangular hybrid automata is decidable [21]. When the sampling rate is not known *a priori*, the problems becomes undecidable [11]. Recently a new notion of sampling has been proposed [24], and it would be interesting to check whether the control problem can become decidable in this (more natural) framework.

Time-abstract restriction. In these games, when the strategy is to wait, it is not possible to bound the time which is waited, and the environment chooses when the next discrete action happens. In this framework, the strategy to wait is viewed as a discrete action, and this somehow "*untimes*" the system. Under this time-abstract restriction, rectangular hybrid games are decidable [13], even for specifications given as LTL formulas [20].

5.2 O-minimal hybrid automata

O-minimal systems [25] are hybrid systems which have very rich continuous dynamics (for instance polynomial and exponential functions can be used), but have limited discrete behaviours (at each discrete step, all variables have to be reset).

The control of such systems is not always decidable as o-minimal structures (and thus reachability) are not necessarily decidable. However if we restrict to decidable structures the (safety and reachability) control of o-minimal hybrid systems is decidable [7].

The technique of the proof is standard as it uses a computation of controllable predecessors over a symbolic representation of the state-space. However there is a major difference with the case of timed automata: in o-minimal hybrid systems, time-abstract bisimulation is not the right tool to compute the set of winning states. A finer bisimulation called *suffix-partition* has to be used to prove decidability. Moreover if the system is controllable, a strategy can be defined in the underlying structure.

6 Conclusion

The control of timed and hybrid has been the core of much research these last ten years. In this paper, we have focused on a formulation of the control problem which is an asymmetric two-player game between the controller and the environment. We have given several results concerning the control problem, from the simplest reachability and safety control problems of timed automata to more involved control problems like the control for external specifications, under partial observability, or for more general systems like subclasses of hybrid automata.

These last years, an extension of timed automata with costs has been studied, which can be used for modeling resource consumption in timed systems. These automata are simple linear hybrid automata with a single hybrid variable (which is an "observer" variable), and can used in a formulation of timed games with an optimization criterion on the cost. Those kinds of games have direct applications for modeling for instance scheduling problems, and have been recently much studied. We refer to [5] for a recent survey on this model.

In 2005, the tool Uppaal TiGA has been developed, which solves the safety control problems of timed automata, symbolically and in a forward manner [10].

References

[1] L. d. Alfaro, M. Faella, Th. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *Proc. 14th International Conference on*

Concurrency Theory (CONCUR'03), volume 2761 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2003.

[2] R. Alur and D. Dill. Automata for modeling real-time systems. In *Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[3] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[4] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier Science, 1998.

[5] P. Bouyer. Weighted timed automata: Model-checking and games. In *Proc. 22nd Conf. Mathematical Foundations of Programming Semantics (MFPS XXII)*, volume 158 of *Electronic Notes in Theoretical Computer Science*, pages 3–17. Elsevier, 2006. Invited paper.

[6] P. Bouyer, L. Bozzelli, and F. Chevalier. Controller synthesis for MTL specifications. Submitted, 2006.

[7] P. Bouyer, Th. Brihaye, and F. Chevalier. Control in o-minimal hybrid systems. In *Proc. 21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*. IEEE Computer Society Press, 2006. To appear.

[8] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *Proc. 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2004.

[9] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *Proc. 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 180–192. Springer, 2003.

[10] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *Proc. 16th International Conference on Concurrency Theory (CONCUR'2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2005.

[11] F. Cassez, Th. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th International Workshop on Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

[12] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In *Proc. Hybrid Systems III: Verification and Control (1995)*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 1996.

[13] L. de Alfaro, Th. A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *Proc. 12th International Conference on Concurrency Theory (CONCUR'01)*, volume 2154 of *Lecture Notes in Computer Science*, pages 536–550. Springer, 2001.

[14] M. De Wulf, L. Doyen, and J.-F. Raskin. A lattice theory for solving games of imperfect information. In *Proc. 8th International Workshop on Hybrid Systems: Computation and Control (HSCC'06)*, volume 3927 of *LNCS*, pages 153–168. Springer, 2006.

[15] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *Lecture Notes in Computer Science*, pages 571–582. Springer, 2002.

[16] M. Faella, S. La Torre, and A. Murano. Automata-theoretic decision of timed games. In *Proc. 3rd International Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI'02)*, volume 2294 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2002.

[17] M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *Proc. 17th Annual Symposium on Logic in Computer Science (LICS'02)*, pages 167–176. IEEE Computer Society Press, 2002.

[18] E. Grädel, W. Thomas, and Th. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[19] Th. A. Henzinger. The theory of hybrid automata. In *Proc. 11th Annual Symposim on Logic in Computer Science (LICS'96)*, pages 278–292. IEEE Computer Society Press, 1996.

[20] Th. A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 1999.

[21] Th. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.

[22] Th. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998.

[23] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[24] P. Krčál and R. Pelánek. On sampled semantics of timed systems. In *Proc. 25th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'05)*, volume 3821 of *Lecture Notes in Computer Science*, pages 310–321. Springer, 2005.

[25] G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, 2000.

[26] F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic – and back. In *Proc. 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 529–539. Springer, 1995.

[27] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[28] J. Ouaknine and J. B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Annual Symposium on Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Computer Society Press, 2005.

[29] A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Computer Society Press, 1977.

[30] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symposium on Principles of Programming Languages (POPL'89)*, pages 179–190. ACM, 1989.

[31] P. Ramadge and W. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, 1989.

THE DISTRIBUTED COMPUTING COLUMN

BY

MARIO MAVRONICOLAS

Department of Computer Science, University of Cyprus
75 Kallipoleos St., CY-1678 Nicosia, Cyprus
mavronic@cs.ucy.ac.cy

A GAME ON A DISTRIBUTED NETWORK[*]

Vicky Papadopoulou

Department of Computer Science

University of Cyprus

P.O. Box 20537, Nicosia CY-1678, Cyprus

viki@cs.ucy.ac.cy

Abstract

Consider a distributed information network with harmful procedures called *attackers* (e.g., viruses); each attacker uses a probability distribution to choose a node of the network to damage. Opponent to the attackers is the *system protector* scanning and cleaning from attackers some part of the network (e.g., an edge or a simple path), which it chooses independently using another probability distribution. Each attacker wishes to maximize the probability of escaping its cleaning by the system protector; towards a conflicting objective, the system protector aims at maximizing the expected number of cleaned attackers. In [8, 9], we model this network scenario as a non-cooperative strategic game on graphs. We focus on two basic cases for the protector; where it may choose a single edge or a simple path of the network. The two games obtained are called as the *Path* and the *Edge* model, respectively. For these games, we are interested in the associated *Nash equilibria,* where no network entity can unilaterally improve its local objective. For the Edge model we obtain the following results:

[*]This work was partially supported by the IST Programs of the European Union under contract numbers IST-2001-33116 (FLAGS) and IST-2004-001907 (DELIS).

- No instance of the model possesses a pure Nash equilibrium.
- Every mixed Nash equilibrium enjoys a graph-theoretic structure, which enables a (typically exponential) algorithm to compute it.
- We coin a natural subclass of mixed Nash equilibria, which we call *matching Nash equilibria,* for this game on graphs. Matching Nash equilibria are defined using structural parameters of graphs
  - We derive a characterization of graphs possessing matching Nash equilibria. The characterization enables a linear time algorithm to compute a matching Nash equilibrium on any such graph.
  - Bipartite graphs and trees are shown to satisfy the characterization; we derive polynomial time algorithms that compute matching Nash equilibria on corresponding instances of the game.
- We proceed with other graph families. Utilizing graph-theoretic arguments and the characterization of mixed NE proved before, we compute, in polynomial time, mixed Nash equilibria on corresponding graph instances. The graph families considered are regular graphs, graphs with, polynomial time computable, r-regular factors and graphs with perfect matchings.
- We define the *social cost* of the game to be the expected number of attackers catch by the protector. We prove that the corresponding *Price of Anarchy* in any mixed Nash equilibria of the Edge model is upper and lower bounded by a linear function of the number of vertices of the graph.

Finally, we consider the more generalized variation of the problem considered, captured by the Path model. We prove that the problem of existence of a pure Nash equilibrium is $\mathcal{NP}$-complete for this model.

1 Introduction

Motivation and Framework. Although *Network Security* has been always considered to be a critical issue in networks, the recent huge growth of public networks (e.g. the Internet) made it even more very important [15]. This work considers a dimension of this area, related to the protection of a system from harmful entities (e.g. viruses, worms, trojan horses, eavesdroppers [4]). Consider an information network where the nodes of the network are insecure and vulnerable to infection by *attackers* such as, viruses, Trojan horses, eavesdroppers. In particular, at any time, a number of harmful entities is known (or an upper bound of this number) to be present in the network. A *protector*, i.e. system security software, is available in the system but it can guarantee security only to a limited part of the network, such as a simple path or a single link of it, which it may choose using a probability distribution. Such limitations result from money and system performance costs caused in order to purchase a global security software or by the reduced efficiency or usability of a protected node. Each harmful entity targets a location (i.e. a node) of the network via a probability distribution; the node is damaged unless it is cleaned by the system security software. Apparently, the harmful entities and the system security software have conflicting objectives. The system security software seeks to protect the network as much as possible, while the harmful entities wish to avoid being caught by the software so that they be able to damage the network. Thus, the system security software seeks to maximize the expected number of viruses it catches, while each harmful entity seeks to maximize the probability it escapes from the system security software.

Naturally, we model this scenario as a non-cooperative multi-player strategic game played on a graph with two kinds of players: the *vertex players* representing the harmful entities, and the *edge* or the *path player* representing each one of the above two cases for the system security software considered; where it can choose a simple path or a single edge of the network, respectively. The corresponding games are called the *Path* and the *Edge* model, respectively. In both cases, the Individual Cost of each player is the quantity to be maximized by the corresponding entity. We are interested in the *Nash equilibria* [11, 12] associated with these games, where no player can unilaterally improve its Individual Cost by switching to a more advantageous probability distribution.

Summary of Results. Here we overview the most important results of [8, 9]. Our study is mainly focus on the Edge model where our results are summarized as follows:

- We prove that the model posses no pure Nash equilibrium (Theorem 3.1).
- We then proceed to study mixed Nash equilibria (mixed NE) of the Edge model. We provide a graph-theoretic characterization of mixed NE (Theorem 3.2). Roughly speaking, the characterization yields that the support of the edge player and the vertex players are an edge cover and a vertex cover of the graph and a subgraph of the graph, respectively. Given the supports, the characterization provides a system of equalities and inequalities to be satisfied by the probabilities of the players. Unfortunately, this characterization only implies an exponential time algorithm for the general case.
- We introduce *matching* Nash equilibria, which are a natural subclass of mixed Nash equilibria with a graph-theoretic definition (Definition 4.1). Roughly speaking, the supports of vertex players in a matching Nash equilibrium form together an independent set of the graph, while each vertex

in the supports of the vertex players is incident to only one edge from the support of the edge player.

- We provide a characterization of graphs admitting a *matching* Nash equilibrium (Theorem 4.4). We prove that a *matching* Nash equilibrium can be computed in linear time for any graph satisfying the characterization once a *suitable* independent set is given for the graph.

- We consider bipartite graphs for which we show that they satisfy the characterization of *matching* Nash equilibria; hence, they always have one (Theorem 5.4). More importantly, we prove that a *matching* Nash equilibrium can be computed in polynomial time for bipartite graphs (Theorem 5.5).

- Next, we proceed with other families of graphs. Combining the characterization of mixed Nash equilibria proved before with suitable graph-theoretic properties of each class addressed, we compute polynomial time mixed NE for each of them. These graph families include, trees, regular graphs, graphs that can be partitioned into vertex disjoint regular subgraphs, graphs with perfect matchings (Theorems 6.5, 6.6, 6.7, 6.9, respectively). Note that trees are also bipartite graphs. Thus, the algorithm for bipartite graphs can apply on them as well. However, the algorithm for trees provided, computes *matched* Nash equilibria in in significantly less time that the algorithm of bipartite graphs. This is achieved via suitable exploration of the special structure of a tree.

- We measure the system performance with respect to the problem considered utilizing the notion of the *social cost* [6]. Here, it is defined to be the number of attackers catch by the protector. We compute upper and lower bounds of the social cost in any mixed Nash equilibria of the Edge model. Using these bounds, we show that the corresponding Price of Anarchy is upper and lower bounded by a linear function of the number of vertices of the graph (Theorem 7.2).

Finally, we consider a more generalized case of the problem considered, represented by the Path model. We prove that the problem of existence of pure Nash equilibria in this model is $\mathcal{NP}$-complete (Theorem 8.2). This result opposes interestingly with the corresponding non-existence result of the Edge model, proved before and indicates some fascinating dimensions of the yet unexplored research area considered here.

Significance and Related Work. Our work joins the booming area of *Algorithmic Game Theory*. At the same time, it contributes in the subfield of *Network*

Security, related to the protection of a network from harmful entities (e.g. viruses, worms, malicious procedures, or eavesdroppers [4]). This work is the *first* work (with an exception of [2]) to model *network security problems* as strategic game and study its associated Nash equilibria. In particular, [2] is a part of a relevant research line related on *Interdependent Security* games [5]. In such a game, a large number of players must make individual investment decisions related to security, in which the ultimate safety of each participant may depend in a complex way on the actions of the entire population. Another related work is that of [4], studying the feasibility and computational complexity of two privacy tasks in distributed environments with *mobile eavesdroppers*; of distributed database maintenance and message transmission. A mobile eavesdropper is a computationally unbounded adversary that move its bugging equipment within the system.

This work is one of the only few works highlighting a fruitful interaction between *Game Theory* and *Graph Theory*. In [2], the authors consider inoculation strategies for victims of viruses and establishes connections with variants of the Graph Partition problem. In [1], the authors study a two-players game on a graph, establish connections with the k-server problem and provide an approximate solution for the simple network design problem.

Our results contribute towards answering the general question of Papadimitriou [14] about the complexity of Nash equilibria for our special game. We believe that our *matching* Nash equilibria (and/or extensions of them) will find further applications in other network games and establish themselves as a candidate Nash equilibrium for polynomial time computation in other settings as well.

2 Framework

Throughout, we consider an undirected graph $G(V, E)$, with $|V(G)| = n$ and $|E(G)| = m$. Given a set of vertices $X \subseteq V$, the graph $G \backslash X$ is obtained by removing from G all vertices of X and their incident edges. A graph H, is an *induced* subgraph of G, if $V(H) \subseteq V(G)$ and $(u, v) \in E(H)$, whenever $(u, v) \in E(G)$. For any vertex $v \in V(G)$, denote $Neigh(v) = \{u : (u, v) \in E(G)\}$, the set of neighboring vertices of v. For a set of vertices $X \subseteq V$, denote $Neigh(X) = \{u \notin X : (u, v) \in E(G)$ for some $v \in X\}$. Denote $\Delta(v) = |Neigh(v)|$ the degree of vertex v in G and $\Delta(G) = \max_{v \in V} |Neigh(v)|$ the maximum degree of G. A *simple* path, P, is a path of G with no repeated vertices, i.e. $P = \{v_1, \cdots, v_i \cdots, v_k\}$, where $1 \leq i \leq k \leq n$, $v_i \in V$, $(v_i, v_{i+1}) \in E(G)$ and each $v_i \in V$ appears at most once in P. Denote $\mathcal{P}(G)$ the set of all possible paths in G. For a tree graph T denote $root \in V$, the root of the tree and $leaves(T)$ the leaves of the tree T. For any $v \in V(T)$, denote $parent(v)$ the parent of v in the tree and $children(v)$ its children in the tree T. For any $A \subseteq V$, let $parents(A) := \{u \in V : u = parent(v),\ v \in A\}$. For all above properties of a graph G, when there is no confusion, we omit G.

2.1 The model

Definition 2.1. *An information network is represented as an undirected graph $G(V, E)$. The vertices represent the network hosts and the edges represent the communication links. For $\mathsf{M} = \{\mathsf{P}, \mathsf{E}\}$, we define a non-cooperative game $\Pi_{\mathsf{M}}(G) = \langle \mathcal{N}, \{S_i\}_{i \in \mathcal{N}}, \{\mathsf{IC}\}_{i \in \mathcal{N}} \rangle$ as follows:*

- *The set of players is $\mathcal{N} = \mathcal{N}_{vp} \cup \mathcal{N}_p$, where $\mathcal{N}_{vp}$ is a finite set of* vertex *players vp_i, $i \geq 1$, $p = \{pp, ep\}$ and $\mathcal{N}_p$ is a singleton set of a player p which is either (i) the* path *player and $p = pp$ or (ii) the* edge *player and $p = ep$, in the case where $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively.*

- *The strategy set S_i of each player vp_i, $i \in \mathcal{N}_{vp}$, is V; the strategy set S_p of the player p is either (i) the set of paths of G, $\mathcal{P}(G)$ or (i) E, when $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively. Thus, the strategy set $\mathcal{S}$ of the game is $\left(\underset{i \in \mathcal{N}_{vp}}{\times} S_i \right) \times S_p$ and equals to $|V|^{|\mathcal{N}_{vp}|} \times |\mathcal{P}(G)|$ or $|V|^{|\mathcal{N}_{vp}|} \times |E|$, when $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively.*

- *Take any* strategy profile $\vec{\mathbf{s}} = \langle s_1, \ldots, s_{|\mathcal{N}_{vp}|}, s_p \rangle \in \mathcal{S}$, *also called a* configuration.

  - *The* Individual Cost *of vertex player vp_i is a function $\mathsf{IC}_i : \mathcal{S} \to \{0, 1\}$ such that $\mathsf{IC}_i(\vec{\mathbf{s}}) = \begin{cases} 0, & s_i \in s_p \\ 1, & s_i \notin s_p \end{cases}$; intuitively, vp_i receives* 1 *if it is not caught by the player p, and* 0 *otherwise.*

  - *The* Individual Cost *of the player p is a function $\mathsf{IC}_p : \mathcal{S} \to \mathbb{N}$ such that $\mathsf{IC}_p(\vec{\mathbf{s}}) = |\{s_i : s_i \in s_p\}|$.*

We call the games obtained as the Path *or the* Edge *model, for the case where $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively.*

The configuration $\vec{\mathbf{s}}$ is a *pure Nash equilibrium* [11, 12] (abbreviated as *pure NE*) if for each player $i \in \mathcal{N}$, it maximizes IC_i over all configurations $\vec{\mathbf{t}}$ that differ from $\vec{\mathbf{s}}$ only with respect to the strategy of player i.

We consider *mixed strategies* for the Edge model. In the rest of the paper, unless explicitly mentioned, when referring to mixed strategies, these apply on the Edge model. A *mixed strategy* for player $i \in \mathcal{N}$ is a probability distribution over its strategy set S_i; thus, a mixed strategy for a vertex player (resp., edge player) is a probability distribution over vertices (resp., over edges) of G. A *mixed strategy profile* $\vec{\mathbf{s}}$ is a collection of mixed strategies, one for each player. Denote

$P_{\vec{s}}(ep, e)$ the probability that edge player ep chooses edge $e \in E(G)$ in $\vec{s}$; denote $P_{\vec{s}}(vp_i, v)$ the probability that player vp_i chooses vertex $v \in V$ in $\vec{s}$. Note $\sum_{v \in V} P_{\vec{s}}(vp_i, v) = 1$ for each player vp_i; similarly, $\sum_{e \in E} P_{\vec{s}}(ep, e) = 1$. Denote $P_{\vec{s}}(vp, v) = \sum_{i \in \mathcal{N}_{vp}} P_{\vec{s}}(vp_i, v)$ the probability that vertex v is chosen by some vertex player in $\vec{s}$.

The *support* of a player $i \in \mathcal{N}$ in the configuration $\vec{s}$, denoted $D_{\vec{s}}(i)$, is the set of pure strategies in its strategy set to which i assigns strictly positive probability in $\vec{s}$. Denote $D_{\vec{s}}(vp) = \bigcup_{i \in \mathcal{N}_{vp}} D_{\vec{s}}(i)$; so, $D_{\vec{s}}(vp)$ contains all pure strategies (that is, vertices) to which some vertex player assigns strictly positive probability. Let also $ENeigh_{\vec{s}}(v) = \{(u, v) \in E : (u, v) \in D_{\vec{s}}(ep)\}$; that is $ENeigh_{\vec{s}}(v)$ contains all edges incident to v that are included in the support of the edge player in $\vec{s}$. Given a mixed strategy profile $\vec{s}$, we denote $(\vec{s}_{-x}, [y])$ a configuration obtained by $\vec{s}$, where all but player x play as in $\vec{s}$ and player x plays the pure strategy y.

A mixed strategic profile $\vec{s}$ induces an *Expected Individual Cost* IC_i for each player $i \in \mathcal{N}$, which is the expectation, according to $\vec{s}$, of its corresponding Individual Cost (defined previously for pure strategy profiles). The mixed strategy profile $\vec{s}$ is a *mixed Nash equilibrium* [11, 12] (abbreviated as mixed NE) if for each player $i \in \mathcal{N}$, it maximizes IC_i over all configurations $\vec{t}$ that differ from $\vec{s}$ only with respect to the mixed strategy of player i. We denote such a strategy profile as $\vec{s}^{\,*}$. Denote $BR_{\vec{s}}(x)$ the set of *best response (pure) strategies* of player x in a mixed strategy profile $\vec{s}$, that is, $\mathsf{IC}_x(\vec{s}_{-x}, y) \geq \mathsf{IC}_x(\vec{s}_{-x}, y')$, $\forall\ y \in BR_{\vec{s}}(x)$ and $y' \notin BR_{\vec{s}}(x)$, $y' \in S_x$, where S_x is the strategy set of player x (see also [13], chapter 3). A *fully* mixed strategy profile is one in which each player plays with positive probability all strategies of its strategy set.

For the rest of this section, fix a mixed strategy profile $\vec{s}$. For each vertex $v \in V$, denote $Hit(v)$ the event that the edge player hits vertex v. So, the probability (according to $\vec{s}$) of $Hit(v)$ is $P_{\vec{s}}(Hit(v)) = \sum_{e \in ENeigh(v)} P_{\vec{s}}(ep, e)$. Define the minimum hitting probability $P_{\vec{s}}$ as $\min_v P_{\vec{s}}(Hit(v))$. For each vertex $v \in V$, denote $m_{\vec{s}}(v)$ the expected number of vertex players choosing v (according to $\vec{s}$). For each edge $e = (u, v) \in E$, denote $m_{\vec{s}}(e)$ the expected number of vertex players choosing either u or v; so, $m_{\vec{s}}(e) = m_{\vec{s}}(u) + m_{\vec{s}}(v)$. It is easy to see that for each vertex $v \in V$, $m_{\vec{s}}(v) = \sum_{i \in \mathcal{N}_{vp}} P_{\vec{s}}(vp_i, v)$. Define the maximum expected number of vertex players choosing e in $\vec{s}$ as $\max_e m_{\vec{s}}(e)$.

We proceed to calculate the Expected Individual Cost. Clearly, for the vertex player $vp_i \in \mathcal{N}_{vp}$,

$$\begin{aligned}
\mathsf{IC}_i(\vec{s}) &= \sum_{v \in V(G)} P_{\vec{s}}(vp_i, v) \cdot (1 - P_{\vec{s}}(Hit(v)) \\
&= \sum_{v \in V(G)} \left(P_{\vec{s}}(vp_i, v) \cdot (1 - \sum_{e \in ENeigh(v)} P_{\vec{s}}(ep, e) \right) \qquad (1)
\end{aligned}$$

For the edge player *ep*,

$$\begin{aligned}\mathsf{IC}_{ep}(\vec{\mathbf{s}}) &= \sum_{e=(u,v)\in E(G)} P_{\vec{\mathbf{s}}}(ep,e)\cdot(m_{\vec{\mathbf{s}}}(u)+m_{\vec{\mathbf{s}}}(v)) \\ &= \sum_{e=(u,v)\in E(G)} \left(P_{\vec{\mathbf{s}}}(ep,e)\cdot\Big(\sum_{i\in N_{vp}} P_{\vec{\mathbf{s}}}(vp_i,u)+P_{\vec{\mathbf{s}}}(vp_i,v)\Big)\right) \qquad (2)\end{aligned}$$

Social Cost and Price of Anarchy. We utilize the notion of *social cost* [6] for evaluating the system performance related to the problem considered. A natural such measurement is the number of attackers catch by the system protector; a maximization of this quantity maximizes system's performance with respect to its safety from harmful entities. We therefore define,

Definition 2.2. *For model* M*,* $\mathsf{M} = \{\mathsf{P},\mathsf{E}\}$*, we define the* social cost *of configuration* $\vec{\mathbf{s}}$ *on instance* $\Pi_{\mathsf{M}}(G)$*,* $\mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{\mathbf{s}})$*, to be the sum of vertex players of* $\Pi_{\mathsf{M}}(G)$ *arrested in* $\vec{\mathbf{s}}$*. That is,* $\mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{\mathbf{s}}) = \mathsf{IC}_p(\vec{\mathbf{s}})$*, where* $p = pp$ *or* $p = ep$ *when* $\mathsf{M} = \mathsf{P}$ *or* $\mathsf{M} = \mathsf{E}$*, respectively. The system wishes to maximize the social cost.*

Definition 2.3. *For model* M*,* $\mathsf{M} = \{\mathsf{P},\mathsf{E}\}$*, we define the* price of anarchy*,* $\mathsf{r}(\mathsf{M})$ *to be,*

$$\mathsf{r}(\mathsf{M}) = \max_{\Pi_{\mathsf{M}}(G),\vec{s}^{\,*}} \frac{\max_{\vec{\mathbf{s}}\in\mathcal{S}} \mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{\mathbf{s}})}{\mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{s}^{\,*})}$$

2.2 Background from Graph Theory

Throughout this section, we consider the (undirected) graph $G = G(V,E)$.
$G(V,E)$ is *bipartite* if its vertex set V can be partitioned as $V = V_1 \cup V_2$ such that each edge $e = (u,v) \in E$ has one of its vertices in V_1 and the other in V_2. Such a graph is often referred to as a V_1,V_2-bigraph. Fix a set of vertices $S \subseteq V$. The graph G is an S*-expander* if for every set $X \subseteq S$, $|X| \leq |Neigh_G(X)|$. For an integer r, graph G is *r-regular* if $\Delta(v) = r$, $\forall v \in V$.

A *factor* of a graph G is a sugraph $G_r \neq G$ such that $V(G_r) = V(G)$. An *r-regular factor* of G is a factor of it (not necessarily connected) which is also an r-regular graph. A *hamiltonian* path of a graph G is a simple path containing all vertices of G. A set $M \subseteq E$ is a *matching* of G if no two edges in M share a vertex. Given a matching M, say that set $S \subseteq V$ is *matched into* $V\backslash S$ in M if for every vertex $v \in S$, there is an edge $(v,u) \in M$ and $u \in V\backslash S$. A *vertex cover* of G is a set $V' \subseteq V$ such that for every edge $(u,v) \in E$ either $u \in V'$ or $v \in V'$. An *edge cover* of G is a set $E' \subseteq E$ such that for every vertex $v \in V$, there is an edge $(v,u) \in E'$. Say that an edge $(u,v) \in E$ (resp., a vertex $v \in V$) is *covered* by the vertex cover V' (resp., the edge cover E') if either $u \in V'$ or $v \in V'$ (resp., if there is an edge $(u, v) \in E'$). A set $IS \subseteq V$ is an *independent set* of G if for all vertices $u, v \in IS$, $(u, v) \notin E$. Clearly, $IS \subseteq V$ is an independent set of G if and only if the set $VC = V \backslash IS$ is a vertex cover of G.

We will use the consequence of Hall's Theorem [3, Chapter 6] on the marriage problem.

Proposition 2.4 (Marriage's Theorem). *A graph G has a matching M in which set $X \subseteq V$ is matched into $V \backslash X$ in M if and only if for each subset $S \subseteq X$, $|Neigh(S)| \geq |S|$.*

Note that the problem of finding a perfect matching of a graph (if there exists one) is equivalent to the problem of finding an 1-regular factor of the graph. The problem of finding a maximum matching of any graph can be solved in polynomial time [10]. Furthermore, a 2-regular factor of a graph (if there exists one) can be computed in polynomial time, via Tutte's reduction [16]; see also [7] for a survey in cycle covers problems of various sizes. By the above observations we get that there exists an exponential number of graphs that have polynomial time computable r-regular factors.

3 Nash Equilibria

All following sections, except the last one, are devoted to the Edge model. For pure Nash equilibria of the Edge model, in [8] we prove:

Theorem 3.1. *If G contains more than one edges, then $\Pi_{\mathsf{E}}(G)$ has no pure Nash Equilibrium.*

Proof. Consider any graph G with at least two edges and any configuration $\vec{\mathbf{s}}$ of $\Pi_{\mathsf{E}}(G)$. Let e the edge selected by the edge player in $\vec{\mathbf{s}}$. Since G contains more than one edges, there exists an $e' \in E$) not selected by the edge player in $\vec{\mathbf{s}}$, such that e and e' contain at least one different endpoint, assume u. If there is at least one vertex player located on e, it will prefer to alternate to u so that not to get arrested by the edge player and gain more. Thus, this case can not be a pure NE. Otherwise, no vertex player is located on edge e. This implies an individual cost of 0 for the edge player which the player can unilaterally improve by selecting any edge containing at least one vertex player. Thus, this case also can not be a pure NE for the instance, concluding that $\vec{\mathbf{s}}$ is not a pure NE. ■

Characterization of Mixed Nash Equilibria. Next we present a characterization of mixed Nash equilibria of the Edge model, proved in [8].

Theorem 3.2. (**Characterization of Mixed NE**) *A mixed strategy profile* $\vec{\mathbf{s}}$ *is a Nash equilibrium for any* $\Pi_E(G)$ *if and only if:*

1. $D_{\vec{\mathbf{s}}}(ep)$ *is an* edge cover *of* G *and* $D_{\vec{\mathbf{s}}}(vp)$ *is a* vertex cover *of the graph obtained by* $D_{\vec{\mathbf{s}}}(ep)$.

2. *The probability distribution of the edge player over* E*, is such that, (a)* $P_{\vec{\mathbf{s}}}(Hit(v)) = P_{\vec{\mathbf{s}}}(Hit(u)) = \min_v P_{\vec{\mathbf{s}}}(Hit(v))$, $\forall\, u, v \in D_{\vec{\mathbf{s}}}(vp)$ *and (b)* $\sum_{e \in D_{\vec{\mathbf{s}}}(ep)} P_{\vec{\mathbf{s}}}(ep, e) = 1$.

3. *The probability distributions of the vertex players over* V *are such that, (a)* $m_{\vec{\mathbf{s}}}(e_1) = m_{\vec{\mathbf{s}}}(e_2) = \max_e m_{\vec{\mathbf{s}}}(e)$, $\forall\, e_1, e_2 \in D_{\vec{\mathbf{s}}}(ep)$ *and*
(b) $\sum_{v \in V(D_{\vec{\mathbf{s}}}(ep))} m_{\vec{\mathbf{s}}}(v) = \nu$.

Remark 3.3. *Note that the characterization does not implies a polynomial time algorithm for computing a mixed Nash equilibrium, since it involves solving a mixed integer programming problem.*

In [9], we also provide an estimation on the payoffs of the vertex players in any Nash equilibrium of the Edge model.

Claim 3.4. *For any* $\Pi_E(G)$*, a mixed NE,* $\vec{s}^{\,*}$*, satisfies* $\mathsf{IC}_i(\vec{s}^{\,*}) = \mathsf{IC}_j(\vec{s}^{\,*})$ *and* $1 - \frac{2}{|D_{\vec{s}^{\,*}}(vp)|} \leq \mathsf{IC}_i(\vec{s}^{\,*}) \leq 1 - \frac{1}{|D_{\vec{s}^{\,*}}(vp)|}$, $\forall i, j \in \mathcal{N}_{vp}$.

4 Matching Nash Equilibria

In [8] we introduce a family of configurations of the Edge model, called *matching*. Such configurations are shown to lead to mixed NE, called *matching* mixed NE. First, we provide a characterization for the existence of a *matching* mixed NE, shown in [8]. Using this characterization, we provide a polynomial time algorithm for the computation of *matching* Nash equilibria for any instance $\Pi_E(G)$ of the problem, where the graph G satisfies the characterization. We remark applicability of the algorithm for a quite broad family of graphs, that of *bipartite graphs* (section 5).

Intuition behind Matching Nash equilibria. The obvious difficulty of solving the system of Theorem 3.2 directs us in trying to investigate the existence of some polynomially computable solutions of the system, corresponding to mixed NE of the game. To which configuration should we consider as *easy to compute*, we utilized the following way of thinking. A first observation is that finding a configuration that satisfies condition **2** of Theorem 3.2 seems the most difficult constrain (among the three conditions) to be fulfilled. This is so because it contains the largest number of variables ($P_{\vec{s}}(ep, e)$, $\forall\ e \in E$) among the three conditions and each equation of it might involve up to $\Delta(G)$ such variables. Thus, let us consider the subtask of the system of computing function $P_{\vec{s}}(\cdot)$, $\forall e \in E$. Consider the case where the equations of condition **2.(a)** are *independent*, that is for each variable e, $P_{\vec{s}}(ep, e)$ appears in only one equation of condition **2.(a)**. Obviously, in this case the task becomes less difficult. Note that in such case, $D_{\vec{s}}(vp)$ constitutes an independent set of G. Moreover, when furthermore, each vertex of $D_{\vec{s}}(vp)$ is incident only in one edge of $D_{\vec{s}}(ep)$, then each equation of condition **2.(a)** contains only one variable, making the satisfaction of the condition even less difficult. Based on these thoughts, in [8], we define the following family of configurations which, as we show, can lead to mixed NE for the game. In the sequel, we investigate their existence and their polynomial time computation.

Definition 4.1. *A* **matching configuration** $\vec{s}$ *of* $\Pi_E(G)$ *satisfies:* **(1)** $D_{\vec{s}}(vp)$ *is an independent set of G and* **(2)** *each vertex v of* $D_{\vec{s}}(vp)$ *is incident to only one edge of* $D_{\vec{s}}(ep)$.

Claim 4.2. *[8] For any graph G, if in* $\Pi_E(G)$ *there exists a* matching *configuration which additionally satisfies condition* **1** *of Theorem 3.2, then there exists probability distributions for the vertex players and the edge player such that the resulting configuration is a mixed Nash equilibrium for* $\Pi_E(G)$*. These distributions can be computed in polynomial time.*

In the proof of the Claim, in [8], we consider any configuration $\vec{s}$ as stated by the Claim (assuming that there exists one) and the following probability distributions of the vertex players and the edge player on $\vec{s}$:

$$\begin{array}{l} \forall e \in D_{\vec{s}}(ep),\ P_{\vec{s}}(ep, e) := 1/|D_{\vec{s}}(ep)|, \\ \forall e' \in E,\ e' \notin D_{\vec{s}}(ep),\ P_{\vec{s}}(ep, e') := 0 \end{array} \tag{3}$$

$$\begin{array}{l} \forall\ i \in \mathcal{N}_{vp},\ \forall\ v \in D_{\vec{s}}(vp),\ P_{\vec{s}}(vp_i, v) := \frac{1}{|D_{\vec{s}}(vp)|}, \\ \forall u \in V,\ u \notin D_{\vec{s}}(vp),\ P_{\vec{s}}(vp_i, u) := 0 \end{array} \tag{4}$$

Then, it is shown that $\vec{s}$ satisfies all conditions of Theorem 3.2, thus it is a mixed NE.

Definition 4.3. *A* matching *configuration which additionally satisfies condition* **1** *of Theorem 3.2 is called a* **matching mixed NE**.

Furthermore, in [8], we characterize graphs that admit *matching* Nash equilibria.

Theorem 4.4. *For any graph G,* $\Pi_E(G)$ *contains a* matching *mixed Nash equilibrium if and only if the vertices of the graph G can be partitioned into two sets IS, VC* ($VC \cup IS = V$ *and* $VC \cap IS = \emptyset$)*, such that IS is an independent set of G (equivalently, VC is a vertex cover of the graph) and G is a VC-expander graph.*

Proof. We first prove that if G has an independent set IS and the graph G is a VC-expander graph, where $VC = V \backslash IS$, then $\Pi_E(G)$ contains a matching *mixed NE.* By the definition of a VC-expander graph, it holds that $Neigh(VC') \geq VC'$, for all $VC' \subseteq VC$. Thus, by the Marriage's Theorem 2.4, G has a matching M such that each vertex $u \in VC$ is matched into $V \backslash VC$ in M; that is there exists an edge $e = (u, v) \in M$, where $v \in V \backslash VC = IS$. Partition IS into two sets IS_1, IS_2, where set IS_1 consists of vertices $v \in IS$ for which there exists an $e = (u, v) \in M$ and $u \in VC$. Let IS_2 the remaining vertices of the set, i.e. $IS_2 = \{v \in IS : \forall\, u \in VC,\ (u, v) \notin M\}$.

Now, recall that there is no edge between any two vertices of set IS, since it is independent set, by assumption. Henceforth, since G is a connected graph, $\forall\ u \in IS_2 \subseteq IS$, there exists $e = (u, v) \in E$ and moreover $v \in V \backslash IS = VC$. Now, construct set $M_1 \subseteq E$ consisting of all those edges. That is, initially set $M := \emptyset$ and then for each $v \in IS_2$, add one edge $(u, v) \in E$ in M_1. Note that, by the construction of the set M_1, each edge of it is incident to only one vertex of IS_2. Next, construct the following configuration $\vec{\mathbf{s}}$ of $\Pi_E(G)$: Set $D_{\vec{\mathbf{s}}}(vp) := IS$ and $D_{\vec{\mathbf{s}}}(ep) := M \cup M_1$.

We first show that that $\vec{\mathbf{s}}$ is a *matching* configuration. Condition (1) of a matching configuration is fulfilled because $D_{\vec{\mathbf{s}}}(vp)(= IS)$ is an independent set. We show that condition (2) of a *matching* configuration is fulfilled. Each vertex of set IS belongs either to IS_1 or to IS_2. By definition, each vertex of IS_1 is incident to only one edge of M and each vertex of IS_2 is incident to no edge in M. Moreover, by the construction of set M_1, each vertex of IS_2 is incident to exactly one edge of M_1. Thus, each vertex $v \in D_{\vec{\mathbf{s}}}(vp)(= IS)$ is incident to only one edge of $D_{\vec{\mathbf{s}}}(ep)(= M \cup M_1)$, i.e. condition (2) holds as well. Henceforth, $\vec{\mathbf{s}}$ is a *matching* configuration.

We next show that condition **1** of Theorem 3.2 is satisfied by $\vec{\mathbf{s}}$. We first show that $D_{\vec{\mathbf{s}}}(ep)$ is an edge cover of G. This is true because (i) set $M \subseteq D_{\vec{\mathbf{s}}}(ep)$ covers all vertices of set VC and IS_1, by its construction and (ii) set $M_1 \subseteq D_{\vec{\mathbf{s}}}(ep)$ covers all vertices of set IS_2, which are the remaining vertices of G not covered by set M, also by its construction. We next show that $D_{\vec{\mathbf{s}}}(vp)$ is a vertex cover of the subgraph of G obtained by set $D_{\vec{\mathbf{s}}}(ep)$. By the definition of sets IS_1, $IS_2 \subseteq IS$, any edge $e \in M$ is covered by a vertex of set IS_1 and each edge $e \in M_1$ is covered by a vertex of set IS_2. Since $D_{\vec{\mathbf{s}}}(ep) = M \cup M_1$, we get that all edges of the set are covered by $D_{\vec{\mathbf{s}}}(vp) = IS_1 \cup IS_2$. This result combined with the above observation on $D_{\vec{\mathbf{s}}}(ep)$ concludes that condition **1** of Theorem 3.2 is satisfied by $\vec{\mathbf{s}}$. Henceforth, by Claim 4.2, it can lead to a *matching* mixed NE of $\Pi_E(G)$.

We proceed to show that if G contains a matching *mixed NE, assume $\vec{\mathbf{s}}$, then G has an independent set IS and the graph G is a VC-expander graph, where $VC = V \backslash IS$.* Define sets $IS = D_{\vec{\mathbf{s}}}(vp)$ and $VC = V \backslash IS$. We show that these sets satisfy the above requirements for G. Note first that, set IS is an independent of

G since $D_{\vec{\mathbf{s}}}(vp)$ is an independent set of G by condition (1) of the definition of a *matching* configuration.

We next show G contains a matching M such that each vertex of VC is matched into $V \backslash VC$ in M. Since $D_{\vec{\mathbf{s}}}(ep)$ is an edge cover of G (condition **1** of a mixed NE of Theorem 3.2), for each $v \in VC$, there exists an edge $(u, v) \in D_{\vec{\mathbf{s}}}(ep)$. Note that for edge (u, v), it holds that $v \in IS$, since otherwise IS would not be a vertex cover of $D_{\vec{\mathbf{s}}}(ep)$ (Condition **1** of a mixed NE). Now, construct a set $M \subseteq E$ consisting of all those edges. That is, That is, initially set $M := \emptyset$ and then for each $v \in VC$, add one edge $(u, v) \in D_{\vec{\mathbf{s}}}(ep)$ in M. By the construction of set M and condition (2) of a *matching* mixed NE, we get that M is a matching of G and that each vertex of VC is matched into $V \backslash VC$ in M. Thus, by the Marriage's Theorem 2.4, we get that $Neigh(VC') \geq VC'$, for all $VC' \subseteq VC$ and so G is a VC-expander and condition (2) of a matching configuration also holds in $\vec{\mathbf{s}}$.

■

An example of a graph G with a *matching* mixed NE $\vec{\mathbf{s}}$ is illustrated in Figure 1. Set $D_{\vec{\mathbf{s}}}(ep)$ is denoted by bold edges and set $D_{\vec{\mathbf{s}}}(vp)(= IS)$ (as in Theorem 4.4) by vertices with an asterisk, $*$. We remark that *not* all graphs have a *matching* mixed NE; any odd cycle is such graph; this is so because for every edge cover EC of the graph (corresponding to $D_{\vec{\mathbf{s}}}(ep)$), there is no set $VC \subseteq V$ (corresponding to $D_{\vec{\mathbf{s}}}(vp)$) such that VC is a vertex cover of the graph induced by EC and VC is also an independent set of G. See Figure 1(b) for an example.

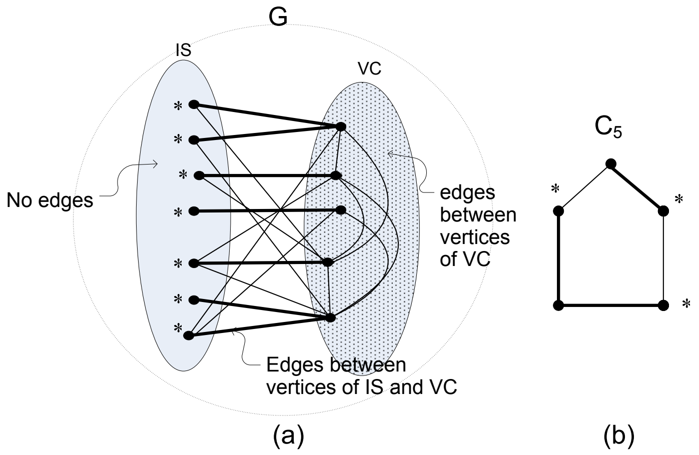

G
IS
VC
No edges
edges between vertices of VC
Edges between vertices of IS and VC
C5
(a)
(b)

Figure 1: Examples of graphs (a) with and (b) without *matching* mixed Nash equilibrium.

4.1 A Polynomial Time Algorithm

The previous Theorems and Lemmas enabled us to develop a polynomial time algorithm for finding *matching* mixed NE for any $\Pi_E(G)$, where G is a graph satisfying the requirements of Theorem 4.4. The algorithm is described in pseudocode in Figure 2.

Theorem 4.5. *[8] Algorithm $A(\Pi_E(G), IS, VC)$ computes a* matching *mixed Nash equilibrium for $\Pi_E(G)$ in linear time $O(n)$.*

5 Bipartite Graphs

In this section we overview the basic results of [8] on the investigation the existence and polynomial time computation of *matching* mixed Nash equilibria for any $\Pi_E(G)$, for which G is a bipartite graph. We first provide some useful Lemmas and Theorems on important properties of bipartite graphs.

Lemma 5.1. *[8] In any bipartite graph G there exists a matching M and a vertex cover VC such that* **(1)** *every edge in M contains exactly one vertex of VC and* **(2)** *every vertex in VC is contained in exactly one edge of M.*

Remark 5.2. *The statement of the Lemma does not hold for all graphs; any odd cycle graph is an example of its falseness (See Figure 1(b)). The falseness of the Lemma in a general graph consists in that the statement $(*1)$ in its proof is false; condition (ii) required for proving $*1$ is not true.*

By the above Lemma 5.1, we can prove that,

Lemma 5.3. *[8] Any X, Y-bigraph graph G can be partitioned into two sets IS, VC ($IS \cup VC = V$ and $IS \cap VC = \emptyset$) such that VC is a vertex cover of G (equivalently, IS is an independent set of G) and G is a VC-expander graph.*

Lemma 5.3 and Theorem 4.4 imply:

Theorem 5.4. *[8] Any $\Pi_E(G)$ for which G is a connected bipartite graph, contains a* matching *mixed Nash equilibrium.*

On the light of above results it is not difficult to show that,

Theorem 5.5. *[8] For any $\Pi_E(G)$, for which G is a bipartite graph, a* matching *mixed Nash equilibrium of $\Pi_E(G)$ can be computed in polynomial time,* $\max\{O(m\sqrt{n}), O(n^{2.5}/\sqrt{\log n})\}$*, using Algorithm A.*

6 Mixed Nash Equilibria in Various Graphs

Here, we overview on polynomial computable Nash equilibria of the Edge model on some practical families of graphs, such as trees, regular graphs, graphs that can be partitioned into vertex disjoint regular subgraphs, graphs with perfect matchings, showed in [9].

6.1 Trees

In Figure 3 we present in pseudocode an algorithm, called $\mathsf{Trees}(\Pi_{\mathsf{E}}(T))$, for computing mixed NE for trees graph instances. The analysis following shows that the algorithm computes a *matched* NE of the instance in linear time $O(n)$. Observe that trees are bipartite graphs, thus by Theorem 5.5 a matched mixed NE of $\Pi_{\mathsf{E}}(T)$ can be computed in time $O(n^{2.5}/\sqrt{\log n})$ via algorithm $A(\Pi_{\mathsf{E}}(G), IS, VC)$ (section 4). Thus, algorithm $\mathsf{Trees}(\Pi_{\mathsf{E}}(T))$ presented next consists a more efficient algorithm that A for computing matched NE for the case where the graph of the instance is a tree.

The proof of correctness of the Algorithm is obtained via a series of Claims proved in [9].

Claim 6.1. *Set VC, computed by Algorithm* $\mathsf{Trees}(\Pi_{\mathsf{E}}(T))$*, is an independent set of* T.

Claim 6.2. *Set EC is an edge cover of* T *and VC is a vertex cover of the graph obtained by set EC.*

Claim 6.3. *Each vertex of* IS *is incident to exactly one edge of* EC.

By Claims 6.1 and 6.3 we prove,

Lemma 6.4. *Configuration* $\vec{s}^{\,t}$ *computed by algorithm* $\mathsf{Trees}(\Pi_{\mathsf{E}}(T))$ *is a* matching *mixed NE.*

By the previous Lemma, combined with Claim 4.2, in the same work it is shown that,

Theorem 6.5. *For any* $\Pi_{\mathsf{E}}(T)$*, where* T *is a tree graph, algorithm* $\mathsf{Trees}(\Pi_{\mathsf{E}}(T))$ *computes a mixed NE in polynomial time* $O(n)$.

6.2 Regular and Polynomially Computable r-factor graphs

Theorem 6.6. *[9] For any* $\Pi_{\mathsf{E}}(G)$ *for which* G *is an* r*-regular graph, a mixed NE can be computed in constant time* $O(1)$.

In the proof of the Theorem, the following configuration $\vec{s}^{\,r}$ on $\Pi_{\mathsf{E}}(G)$ is constructed:

$$\begin{aligned}&\text{For } any\ i \in \mathcal{N}_{vp},\ P_{\vec{s}^{\,r}}(vp_i, v) := \tfrac{1}{n},\ \forall v \in V(G) \text{ and then set, } \vec{s}^{\,r}_j := \vec{s}^{\,r}_i,\\ &\forall j \neq i, j \in \mathcal{N}_{vp}\ .\ Set\ \ P_{\vec{s}^{\,r}}(ep, e) := \tfrac{1}{m},\ \forall e \in E.\end{aligned} \tag{5}$$

Then, its is shown that $\vec{s}^{\,r}$ is a mixed NE for $\Pi_{\mathsf{E}}(G)$.

The above result can be extended to graphs containing polynomially computable r-regular factors subgraphs.

Corollary 6.7. *For any $\Pi_{\mathsf{E}}(G)$ for which G is contains an r-regular factor subgraph, a mixed NE can be computed in polynomial time $O(T(G))$, where $O(T(G))$ is the time needed for the computation of G_r from G.*

Observation 6.8. *For any $\Pi_{\mathsf{E}}(G)$ for which G is a* 2-regular factor *graph, a mixed NE can be computed in polynomial time, $O(T(G))$, where $O(T(G))$ is the (polynomial) time needed for computing G_2.*

6.3 Perfect Graphs

Theorem 6.9. *[9] For any $\Pi_{\mathsf{E}}(G)$ for which G has a perfect matching, a mixed NE can be computed in linear time, $O(\sqrt{n} \cdot m)$.*

In the proof of the Theorem, first, a perfect matching M of G is computed. Then, the following configuration $\vec{s}^{\,p}$ on $\Pi_{\mathsf{E}}(G)$ is constructed:

$$\begin{aligned}&\text{For } any\ i \in \mathcal{N}_{vp},\ P_{\vec{s}^{\,p}}(vp_i, v) := \tfrac{1}{n},\ \forall v \in V(G) \text{ and set } \vec{s}^{\,p}_j := \vec{s}^{\,p}_i,\\ &\forall j \neq i, j \in \mathcal{N}_{vp}\ .\ \text{Set } P_{\vec{s}^{\,p}}(ep, e) := \tfrac{1}{|M|},\ \forall e \in E.\end{aligned} \tag{6}$$

Then, it is shown, that both kinds of players, the vertex players and the edge player are satisfied in $\vec{s}^{\,p}$. Thus it is a mixed NE for $\Pi_{\mathsf{E}}(G)$.

7 The Price of Anarchy

In this section we overview on the basic results of [9] on the Social Cost and Price of Anarchy of the Edge model.

Lemma 7.1. *For any $\Pi_{\mathsf{E}}(G)$ and an associated mixed NE $\vec{s}^{\,*}$, the social cost $\mathsf{SC}(\Pi_{\mathsf{E}}(G), \vec{s}^{\,*})$ is upper and lower bounded as follows:*

$$\max\left\{\frac{\nu}{|D_{\vec{s}^{\,*}}(ep)|}, \frac{\nu}{|V(D_{\vec{s}^{\,*}}(vp))|}\right\} \ \leq\ \mathsf{SC}(\Pi_{\mathsf{E}}(G), \vec{s}^{\,*}) \leq \frac{\Delta(D_{\vec{s}^{\,*}}(ep)) \cdot \nu}{|D_{\vec{s}^{\,*}}(ep)|} \tag{7}$$

These bounds are tight.

Theorem 7.2. *The Price of Anarchy for the Edge model is $\frac{n}{2} \leq \mathsf{r}(\mathsf{E}) \leq n$.*

8 The Path Model

In the last section, we take a glimpse on the Path model. In [9], we provide the following characterization of pure Nash Equilibria in the Path model.

Theorem 8.1. *For any graph G, $\Pi_{\mathsf{P}}(G)$ has a pure NE if and only if G contains a hamiltonian path.*

This characterization implies the following result regarding the existence of pure NE.

Corollary 8.2. *The problem of deciding whether there exists a pure NE for any $\Pi_{\mathsf{P}}(G)$ is $\mathcal{NP}$-complete.*

References

[1] N. Alon, R. M. Karp, D. Peleg and D. West, "A Graph-Theoretic Game and its Application to the k-Server Problem", *SIAM Journal on Computing*, Vol 24, No 1, pp. 78-100, February 1995.

[2] J. Aspnes, K. Chang and A. Yampolskiy, "Inoculation Strategies for Victims of Viruses and the Sum-of-Squares Problem", *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 43-52, January 2005.

[3] A. S. Asratian, D. Tristan and M. J. Häggkvist, *Bipartite Graphs and Their Applications*, Cambridge Tracts in Mathematics, 131, 1998.

[4] M. Franklin, P. Alto, Z. Galil and Moti Yung, "Eavesdropping Games: a Graph-Theoretic Approach to Privacy in Distributed Systems", *Journal of the ACM*, Vol 47, No 2, pp. 225-243, March 2000.

[5] M. Kearns and L. Ortiz, "Algorithms for Interdependent Security Games", *Proceedings of the 17th Annual Conference on Neural Information Processing Systems*, December 2003.

[6] E. Koutsoupias and C. H. Papadimitriou, "Worst-Case Equilibria", *In Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 404–413, Springer-Verlag, March 1999.

[7] B. Manthey, "On Approximating Restricted Cycle Covers", Technical Report *arXiv:cs.CC/0504038 v2* , June 10, 2005.

[8] M. Mavronicolas, V. Papadopoulou, A. Philippou, P. Spirakis, "A Network Game with Attacker and Protector Entities", *In the Proceedings of the 16th Annual International Symposium on Algorithms and Computation*, 2005.

[9] M. Mavronicolas, V. Papadopoulou, A. Philippou, P. Spirakis, "A Graph-Theoretic Network Security Game", *In the Proceedings of the 1st Workshop on Internet and Network Economics*, 2005.

[10] S. Micali and V.V. Vazirani, "An $O(V^{1/2}E)$ Algorithm for Finding Maximum Matching in General Graphs", *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science*, pp. 17-27, 1980.

[11] J. F. Nash, "Equilibrium Points in n-Person Games", *Proceedings of the National Acanemy of Sciences of the United States of America*, Vol 36, pp 48-49, 1950.

[12] J. F. Nash, "Non cooperative Games", *Annals of Mathematics*, Vol 54, No 2, pp. 286-295, 1951.

[13] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, MIT Press, 1994.

[14] C. H. Papadimitriou, "Algorithms, Games, and the Internet", *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pp. 749-753, June 2001.

[15] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Third Edition, Prentice Hall, 2003.

[16] W. T. Tutte, "A Short Proof of the Factor Theorem for Finite Graphs", *Canadian Journal of Mathematics*, Vol 6, pp. 347-352, 1954.

Algorithm $A(\Pi_E(G), IS, VC)$

INPUT: A game $\Pi_E(G)$ and a partition of $V(G)$ into sets IS, $VC = V\backslash IS$, such that IS is an independent set of G and G is a VC-expander graph.
OUTPUT: A mixed NE $\vec{s}$ for $\Pi_E(G)$.

```
1. Compute a set $M \subseteq E$ as follows:
   (a) Initialization: Set $M := \emptyset$, $Matched := \emptyset$ (currently matched vertices in $M$), $Unmatched := VC$ (currently unmatched vertices of $VC$ in $M$), $Unused := IS$, $i := 1$, $G_i := G$ and $M_1 := \emptyset$.
   (b) While $Unmatched \neq \emptyset$ Do:
       i. Consider a $u \in Unmatched$.
       ii. Find a $v \in Unused$ such that $(u, v) \in E_i$. Set $M := M \cup (u, v)$, $Unused := Unused \backslash \{v\}$.
       iii. Prepare next iteration: Set $i := i + 1$, $Matched := Matched \cup \{u\}$, $Unmatched := Unmatched \backslash \{u\}$, $G_i := G_{i-1} \backslash u \backslash v$.
2. Partition set $IS$ into two sets $IS_1$, $IS_2$ as follows: $IS_1 := \{u \in IS : \exists (u, v) \in M\}$ and $IS_2 := IS \backslash IS_1$. Note that $IS_2 := \{u \in IS : \forall v \in VC, \nexists (u, v) \in M\}$.
   Compute set $M_1$ as follows: $\forall u \in IS_2$, set $M_1 := M_1 \cup (u, v)$, for any $(u, v) \in E$, $v \in VC$.
3. Define a configuration $\vec{s}$ with the following support: $D_{\vec{s}}(vp) := IS$, $D_{\vec{s}}(ep) := M \cup M_1$.
4. Determine the probabilities distributions of the vertex players and the e.p. of configuration $\vec{s}$ using equations (3) and (4) of Claim 4.2.
```

Figure 2: Algorithm $A(\Pi_E(G), IS, VC)$.

```
Algorithm Trees($\Pi_E(T)$)

  1. Initialization: $VC := \emptyset$, $EC := \emptyset$, $r := 1$, $T_r := T$.

  2. Repeat until $T_r == \emptyset$

     (a) Find the leaves of the tree $T_r$, $leaves(T_r)$.
     (b) Set $VC := VC \cup leaves(T_r)$.
     (c) For each $v \in leaves(T_r)$ do:
             If $parent_{T_r}(v) \neq \emptyset$, then $EC := EC \cup \{(v, parent_{T_r}(v)))\}$,
             else $EC := EC \cup \{(v, u)\}$, for any $u \in children_T(v)$.
     (d) Update tree: $T_{r+1} := T_r \backslash leaves(T_r) \backslash parents(leaves(T_r))$.
         Set $r := r + 1$.

  3. Define a configuration $\vec{s}^{\,t}$ with the following support:
     For any $i \in \mathcal{N}_{vp}$, set $D_{\vec{s}^{\,t}}(vp_i) := VC$ and $D_{\vec{s}^{\,t}}(ep) := EC$. Then
     set $D_{\vec{s}^{\,t}}(vp_j) := D_{\vec{s}^{\,t}}(vp_i)$, $\forall\ j \neq i,\ j \in \mathcal{N}_{vp}$.

  4. Determine the probabilities distributions of players in $\vec{s}^{\,t}$ as
     follows:
     $ep$ : $\forall\ e \in D_{\vec{s}^{\,t}}(ep)$, set $P_{\vec{s}^{\,t}}(ep, e) := 1/|EC|$. Also, $\forall\ e' \in E(T)$,
     $e' \notin D_{\vec{s}^{\,t}}(ep)$, set $P_{\vec{s}^{\,t}}(ep, e') := 0$.

     For any $vp_i$, $i \in \mathcal{N}_{vp}$ : $\forall\ v \in D_{\vec{s}^{\,t}}(vp_i)$, set $P_{\vec{s}^{\,t}}(vp_i, v) := \frac{1}{|VC|}$.
     Also, $\forall\ u \notin D_{\vec{s}^{\,t}}(vp_i)$, set $P_{\vec{s}^{\,t}}(vp_i, u) := 0$. Then set $\vec{s}^{\,t}_j = \vec{s}^{\,t}_i$,
     $\forall\ j \neq i,\ j \in \mathcal{N}_{vp}$.
```

Figure 3: Algorithm Trees($\Pi_E(T)$).

The Formal Language Theory Column

by

Arto Salomaa

Turku Centre for Computer Science
University of Turku
Lemminkäisenkatu 14, 20520 Turku, Finland
asalomaa@it.utu.fi

Enumeration of Formal Languages

Michael Domaratzki
Jodrey School of Computer Science
Acadia University
Wolfville, NS B4P 2R6 Canada
mike.domaratzki@acadiau.ca

Abstract

We survey recent results on the enumeration of formal languages. In particular, we consider enumeration of regular languages accepted by deterministic and nondeterministic finite automata with n states, regular languages generated by regular expressions of a fixed length, and ω-regular languages accepted by Müller automata. We also survey the uncomputability of enumeration of context-free languages and more general structures.

1 Introduction

Given a set of objects, enumeration asks "how many distinct objects are there?" Easy examples of enumeration problems are "how many binary sequences of length n are there?" (2^n) and "how many distinct subsets of size m can we take from a set of n elements?" ($\binom{n}{m}$). A sampling of other classical topics for enumeration familiar to computer scientists include graphs ("how many non-isomorphic graphs on n vertices are there?"), trees, primitive and Lyndon words [33, A000031], and monotone Boolean functions (this enumeration problem is known as Dedekind's problem [33, A000372]). With over 100 000 sequences, the Encyclopedia of Integer Sequences [33] contains a wealth of examples related to enumeration.

The enumeration of structures in formal language theory is a topic that has been considered for almost fifty years, and the objects in formal language theory yield many interesting enumeration problems. In this survey, we consider recent results on the enumeration of formal languages. Many of these results concern enumeration of finite automata, but we also consider enumeration of regular expressions, context-free languages, and more general results.

Why enumeration? There are several compelling reasons for studying the enumeration of formal languages beyond the intrinsic research challenge. In particular, research on enumeration is closely linked to problems of random generation of automata [3], average case complexity [27] and establishing lower bounds by counting arguments (see, e.g., Domaratzki *et al.* [11, Thm. 2.5] for an example from formal language theory). Results on enumeration are useful in varied locations when, for one reason or another, the number of regular languages of a given size is required. A recent example is given Gramlich and Schnitger [14], who use bounds on the number of regular languages accepted by NFAs with n states in proving inapproximability results for finding minimal NFAs for a given DFA.

2 Enumeration and Formal Language Theory

Given an infinite set of objects $\mathcal{S}$, enumeration asks the question "how many distinct objects are there in $\mathcal{S}$ of size n"? The goal of enumeration is to express this quantity exactly as a function of n, typically in a closed form. Asymptotics for these functions are typically also of interest to researchers, for comparative purposes.

There are some important assumptions in enumeration problems. The two most crucial—especially in relation to formal language enumeration—are the *measurement* and the idea of *equivalence*. First, we must have a measure on $\mathcal{S}$ such that the number of objects of size n is finite for all n. Clearly, without a measure satisfying these requirements, asking enumeration questions doesn't make sense. When enumerating structures in formal language theory, there are often several different descriptional complexity measures available. This gives many, often unique research questions for the same structure, as we see in this survey. Secondly, our enumeration problem asks about the number of *distinct* objects of size n. Thus, we must have a concept of which objects are and are not equivalent.

Typically, in classical areas like graph theory, the notion of equivalence means *isomorphic*: two directed graphs are equivalent if they are isomorphic. In formal language theory, we still use the notion of isomorphic—when discussing the uniqueness of minimal DFAs, for instance. However, when dealing with devices which generate or accept languages, our primary notion of equivalence is usually equality of the languages they generate or accept. Thus, we focus on this concept of equivalence in this survey: two language devices are equivalent if they accept or generate the same language.

3 Preliminaries

We assume the reader is familiar with the basic notions of formal language theory, in particular, the concepts of deterministic and nondeterministic finite automata (DFAs and NFAs), regular expressions, regular languages, context-free grammars (CFGs) and context-free languages (CFLs). See, for example, Rozenberg and Salomaa [32] for an introduction to concepts used in this survey.

We will employ a few descriptional complexity measures of regular languages below. The (deterministic) state complexity of a regular language L is the minimal number of states in any DFA accepting L. See Yu [36, 37] for surveys of results on state complexity. The nondeterministic state complexity [17] of a regular language L is, as expected, the minimal number of states in any NFA accepting L.

4 Early Results

Since nearly the inception of the study of formal languages, there has been interest in enumeration problems relating to automata. For a list of references and background, we refer the reader to Domaratzki *et al.* [9], where it is noted that the problem was considered at least as early as 1959, and in 1960, Harary listed enumeration of automata as an unsolved problem in graph enumeration. Harrison [16] wrote "A census of finite automata" in 1965, which provided enumeration results using group-theoretic means. Many other papers also attacked the enumeration of automata, including strongly-connected, initially-connected[1] and minimal automata. Much research was independently conducted in the Soviet Union and in the West.

Most early research focuses on enumerating automata by considering them to be distinct if they are non-isomorphic, and little attention is given to the languages

[1]Recall that a DFA is *initially-connected* if, for each state q, there is a word w such that $\delta(q_0, w) = q$, and similarly for an NFA. Initially-connected automata are also called *accessible* in the literature.

accepted by the automata. Some of the work on enumeration of minimal automata does begin to address the number of languages accepted by DFAs. To see this, let $f_k(n)$ be the number of pairwise non-isomorphic minimal DFAs with n states over a k-letter alphabet and $g_k(n)$ be the number of distinct regular languages accepted by DFAs with n states over a k-letter alphabet. Then we note that $g_k(n) = \sum_{i=1}^{n} f_k(i)$ [9, Prop. 1]. Thus, in what follows, we are generally looking for bounds on $g_k(n)$, but it will be sufficient to obtain bounds on $f_k(n)$.

For research on enumeration of minimal finite automata, we mention here in particular the less well-known work of Narushima [24, 25, 26], who developed new methods, namely inclusion-exclusion properties on semi-lattices, for enumeration of minimal automata. These techniques appear to have never been exploited to enumerate formal languages (in particular, the relationship between Narushima's methods and methods for enumerating initially-connected automata does not appear to have been studied) and the inclusion-exclusion principles do not appear to have ever been employed to give any asymptotic analysis of the number of minimal automata with n states.

There is also other work on minimal automata which we do not cover in this survey. The work of Korshunov [18, 19] (a survey in Russian is also available [20]) enumerates minimal automata. However, as noted in Domaratzki *et al.* [9], the automata studied by Korshunov lack a distinguished initial state. Korshunov also studies initially-connected automata (in which an initial state is given [20, Ch. 4]), however, it does not appear that the work was broadened to study initially-connected minimal automata.

5 Enumeration by State Complexity

Renewed interest in the enumeration of formal languages can be traced to the work of Nicaud which investigated average state complexity of operations on regular languages [27]. In order to examine the average case complexity of these operations, an exact characterization of all distinct automata with n states is required. Nicaud gives such a characterization for unary regular languages and, as a by-product, also gives an asymptotic enumeration of unary regular languages. Recall that $f_k(n)$ denotes the number of pairwise non-isomorphic minimal DFAs with n states over a k-letter alphabet and $g_k(n)$ denotes the number of distinct regular languages accepted by DFAs with n states over a k-letter alphabet. The following result is due to Nicaud [27]:

Theorem 1. *The function $f_1(n)$ satisfies $f_1(n) \sim n2^{n-1}$.*

This result of Nicaud was considered by Domaratzki *et al.* [9]. In particular, the asymptotic bound on $f_1(n)$ can be further refined, and an asymptotic bound on

$g_1(n)$ can also be given [9]:

Corollary 2. *The following asymptotic bound holds:* $g_1(n) = 2^n(n - \alpha + O(n2^{-n/2}))$ *where* α *is a constant with approximate value* 1.382714455402.

The value of α in Theorem 2 is given by a sum involving the Möbius function [9]. Domaratzki *et al.* also examine the behaviour of the function $f_k(n)$ for $k \geq 2$. These results depend on the following theorem due to Liskovets [23] and, independently, Robinson [31]. Let $C_k(n)$ be the number of initially-connected DFAs (without final states) on n states over an alphabet of size k.

Theorem 3. *Let* $n, k \geq 2$. *The function* $C_k(n)$ *satisfies the following recurrence:*

$$C_k(n) = n^{nk} - \sum_{i=1}^{n-1} \binom{n-1}{i-1} C_k(i) n^{(n-i)k}. \tag{1}$$

The asymptotics of $C_k(n)$ are are given by Robinson [31]:

$$C_k(n) = n^{kn} \gamma_k^{n(1+o(1))}, \tag{2}$$

where γ_k is a constant depending only on k, the size of the alphabet. Korshunov [20, p. 50] also gives precise results in this area. Using Theorem 3, Domaratzki *et al.* [9] give asymptotic bounds on $f_k(n)$ for $k \geq 2$:

Theorem 4. *The function* $f_k(n)$ *is bounded below by a function which is asymptotically* $(k - o(1))n2^{n-1}n^{(k-1)n}$ *and bounded above by* $2^n C_k(n)/(n-1)!$.

Thus, considering the estimates of (2), the upper and lower bounds in Theorem 4 differ by a factor of $(\gamma_k e)^n$. For $k = 2$ this is approximately 2.27^n [9].

Reis *et al.* [30] have also considered enumeration of automata and in particular, initially-connected DFAs. By proposing a canonical, compact string representation for initially-connected DFAs, Reis *et al.* give an alternate formula for $C_k(n)$ [30].

Theorem 5. *The function* $C_k(n)$ *satisfies the following formula:*

$$C_k(n) = \sum_{b_1=1}^{k} \sum_{b_2=1}^{2k-b_1} \sum_{b_3=1}^{3k-b_1-b_2} \cdots \sum_{b_{n-1}=1}^{k(n-1)-\sum_{\ell=1}^{n-2} b_\ell} \prod_{j=1}^{n} j^{b_j - 1}. \tag{3}$$

Finally, we consider the recent work by Bassino and Nicaud [1], who also study the enumeration of non-isomorphic initially-connected DFAs. Recall that the Stirling numbers of the second kind, denoted here by $S_2(n, m)$, are defined by $S_2(0, 0) = 1$, $S_2(n, 0) = 0$ for all $n \geq 1$ and, for all $n, m \geq 1$,

$$S_2(n, m) = mS_2(n-1, m) + S_2(n-1, m-1).$$

The main enumerative result of Bassino and Nicaud gives bounds on $C_k(n)$ [1]:

Theorem 6. *Let $n, k \geq 1$. The following asymptotic bound holds:*

$$C_k(n) \in \Theta\left(nS_2(kn, n)\right). \tag{4}$$

We note that Theorem 6 is obtained from exact bounds on $C_k(n)$. Bassino and Nicaud also reinterpret a result of Korshunov using Stirling numbers of the second kind. Note that Theorems 3, 5 and 6 all do not account for the choices of final states. Thus, each of these quantities can be multiplied by a factor of 2^n, as is done in the upper bound of Theorem 4.

5.1 Enumeration by Nondeterministic State Complexity

Despite the long history of enumeration of finite automata, and the central importance of nondeterminism in automata theory, there does not appear to have been any consideration of the enumeration of nondeterministic finite automata or of regular languages by their nondeterministic state complexity until very recently. Estimates of this quantity have appeared in at least one instance (in 1997 by Pomerance *et al.* [29], which we note below), but the first study of the enumeration problem appears to be by Domaratzki *et al.* [9]. Let $G_k(n)$ denote the number of distinct regular languages accepted by NFAs with n states over a k-letter alphabet. We first consider the unary case [9]:

Theorem 7. *The function $G_1(n)$ satisfies the inequality $G_1(n) \geq 2^{n+(2.295-o(1))\sqrt{\frac{n}{\log n}}}$.*

Theorem 7 is given by languages which are accepted by NFAs in Chrobak normal form [4]. A non-trivial upper bound on $G_1(n)$ is given by Pomerance *et al.* [29]:

Theorem 8. *There are $O(n/(\log n))^n$ distinct unary languages accepted by NFAs with n states.*

For larger alphabets, the following bounds are known [9]:

Theorem 9. *For $k \geq 2$, we have $n2^{(k-1)n^2} \leq G_k(n) \leq (2n-1)2^{kn^2} + 1$.*

One fact worth noting is that the upper bound in Theorem 9 does not enforce that the NFAs are initially-connected. In fact, it can be shown that there are asymptotically 2^{kn^2} initially-connected NFAs on n states over a k-letter alphabet with a fixed initial state and no final states [9, Thm. 12]. This result is derived from analyzing the recurrence analogous to (2) for NFAs.

5.2 Enumeration of Finite Languages by State Complexity

We now turn to enumeration of finite languages. Recently, finite languages have received an increasing amount of attention. The state complexity of language operations acting on finite languages is almost as well-studied as that of regular languages (the survey of Yu [37] also covers the case where the languages are finite). Further, the relationship between the state complexity and the longest word in a finite language has been recently studied [2].

Let $f'_k(n)$ denote the number of pairwise non-isomorphic DFAs with n states over a k-letter alphabet which accept finite languages. For finite unary languages, enumeration is trivial: the number of finite unary languages accepted by a DFA with n states is exactly 2^{n-1}. For larger alphabets, the problem has been studied by Domaratzki *et al.* [9], Domaratzki [7] and Liskovets [22].

For arbitrary alphabets, a lower bound may be given by an explicit construction [9, Thm. 15]:

Theorem 10. *For $k, n \geq 2$, $f'_k(n) \geq 2^{n-2}((n-1)!)^{k-1}$.*

Domaratzki [7] gives an improved lower bound on the number of finite languages accepted by DFAs with n states over a binary alphabet. In particular, the following bound is given by explicitly constructing large sets of finite languages all accepted by DFAs with n states [7]:

Theorem 11. *For all $n \geq 5$, $f'_2(n) \geq \frac{(2n-3)!}{(n-2)!} c_1^{n-2}$ for some constant $c_1 \simeq 1.0669467$.*

An upper bound on the number of finite languages accepted by DFAs with n states over a binary alphabet is possible by giving another combinatorial interpretation to the classical Genocchi numbers. The Genocchi numbers G_{2n} for $n \geq 1$ can be defined in terms of the following generating function (see Sloane [33, A001469] for further references):

$$\frac{2t}{e^t + 1} = t + \sum_{n \geq 1} (-1)^n G_{2n} \frac{t^{2n}}{(2n)!}.$$

In particular, we have the following result [6]:

Theorem 12. *For all $n \geq 2$, $f'_2(n) \leq 2^{n-2} G_{2n}$.*

Theorem 12 can be extended to alphabets of size k using an generalization of the Genocchi numbers due to Han [15].

Enumeration of finite languages has also been considered by Liskovets [22] by enumerating acyclic unlabelled DFAs. Using two approaches previously developed, Liskovets gives an exact enumeration of unlabelled DFAs accepting finite languages.

Let $a_k(n,r)$ be the recurrence defined by

$$a_k(n,r) = \sum_{t=0}^{n-1} \binom{n}{t} (-1)^{n-t-1} (t+r)^{k(n-t)} a_k(t,r)$$

for $n, r \geq 1$ and $a_k(0,r) = 1$ for all $r \geq 0$. The recurrence $a_k(n,r)$ enumerates DFAs which are called *quasi-acyclic* by Liskovets, but is primarily an auxiliary recurrence for the following result [22]:

Theorem 13. *Let $c_k(n)$ be the function defined by $c_k(1) = 1$ and*

$$\sum_{t=1}^{n} \binom{n-1}{t-1} a_k(n-t, t+1) \cdot c_k(t) = a_k(n,1)$$

for $n \geq 2$. Then $c_k(n)$ gives the number of labelled, initially-connected acyclic DFAs on n states over a k-letter alphabet.

As Liskovets notes, the number of *unlabelled* initially connected acyclic DFAs is given by the quantity $c_k(n)/(n-1)!$. The above bounds can be further improved by considering only DFAs with a unique so-called pre-dead state (the pre-dead state is the state for which all of its transitions enter the dead state). Though Liskovets does not give asymptotics for $c_k(n)$, numerical evidence suggests it gives a good upper bound on the number of finite languages accepted by DFAs with at most n states.

5.3 Enumeration of Finite Languages by Nondeterministic State Complexity

For enumeration of finite languages by nondeterministic state complexity, let $G'_k(n)$ denote the number of finite languages over a k-letter alphabet with nondeterministic state complexity n. We have the following result [9].

Theorem 14. *We have $G'_1(n) = 2^n$, and for all $k \geq 2$ and $n \geq 2$,*

$$2^{(k-1)n(n-1)/2} \leq G'_k(n) \leq 2^{n-1+kn(n-1)/2}.$$

5.4 Enumeration by ⊕-State Complexity

Recently, van Zijl [35] has considered enumeration problems for ⊕-DFAs and ⊕-NFAs. A *symmetric difference NFA* (or ⊕-NFA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where each component is the same as a traditional NFA. However, we extend δ to a function $\delta : Q \times \Sigma^* \to 2^Q$ as follows:

$$\begin{array}{rcl} \delta(q,\epsilon) & = & \{q\} \quad \forall q \in Q \\ \delta(q,aw) & = & \bigoplus_{q' \in \delta(q,a)} \delta(q',w) \quad \forall q \in Q, w \in \Sigma^*, a \in \Sigma. \end{array}$$

Here, $\oplus$ is the symmetric difference operation on sets: $X_1 \oplus X_2 = (X_1 \backslash X_2) \cup (X_2 \backslash X_1)$. Thus, $\oplus$-NFAs are obtained from traditional NFAs by extending the transition function to words by using symmetric difference instead of union. A $\oplus$-DFA is any DFA obtained by applying the subset construction to a $\oplus$-NFA.

van Zijl considers enumeration of regular languages by the number of states in the $\oplus$-NFA and $\oplus$-DFA simultaneously. This problem has been considered for traditional NFAs and DFAs by Domaratzki *et al.* [9]. Let φ be the Euler totient function. We have the following result [35, Thm. 10]:

Theorem 15. *For all $n \geq 1$, there are at least $\frac{2^n}{n}\varphi(2^n-1)$ distinct regular languages over a binary alphabet such that each is accepted by an n-state $\oplus$-NFA, and the minimal $\oplus$-DFA for each has $2^n - 1$ states.*

6 Enumeration by Regular Expression Size

Lee and Shallit have recently investigated the enumeration of regular languages by regular expression size [21]. This follows previous work, most recently by Ellul *et al.* [13], on the study of regular expression size as a descriptional complexity measure for regular languages. The work of Ellul *et al.* [13] includes investigations of trade-offs between regular expression size and automata size and the effect of operations on regular expression size.

The study of the descriptional complexity of regular expressions requires us to be precise about our measure of the length of a regular expression. For instance, Lee and Shallit [21] and Ellul *et al.* [13] consider the following three measures:

(a) The *ordinary length* of a regular expression, that is, the number of symbols in the regular expression, including parentheses, ϵ and $\emptyset$.

(b) The *reverse polish* length, which is the length of the equivalent expression written in reverse polish (postfix) notation.

(c) The *alphabetic length*, which counts only letters from the alphabet Σ, and ignores all operators, occurrences of ϵ and $\emptyset$, and parentheses.

Ellul *et al.* [13] note that each of these lengths is linearly related to each other, provided the expressions do not contain some basic forms of redundancy (such redundancy-avoiding expressions are called *irreducible* by Ellul *et al.* [13], where we refer the reader for more details).

The techniques of Lee and Shallit are themselves worth mentioning. The first step is constructing a CFG G such that $L(G)$ generates the language of all valid regular expressions over an alphabet Σ (i.e., $L(G)$ consists of words over the alphabet $\Sigma \cup \{(,), \emptyset, \epsilon, +, *\}$, each of which is a valid regular expression). Using the Chomsky-Shützenberger Theorem, G can be translated to a system of linear equations which (implicitly) give the number of regular expressions of a given length. Lee and Shallit then use Gröbner bases to obtain a generating function for the number of regular expressions of length n. This technique enumerates all valid regular expressions, which treats regular expressions as being distinct if they differ as words generated by the grammar G.

In the following, $S_k(n)$ denotes the number of valid regular expressions of ordinary length n over a k-letter alphabet. The following result is due to Lee and Shallit [21]:

Theorem 16. *The function $S_k(n)$ satisfies $S_k(n) \sim c_k \alpha_k^n n^{-3/2}$, for some constant c_k, where $\alpha_1 = 6.1552665$ and $\alpha_2 = 7.2700161767$.*

Clearly, $S_k(n)$ is an upper bound on the number of distinct regular languages generated by a regular expression of length n over a k-letter alphabet, denoted by $R_k(n)$. By further refining the grammars used to generate regular expressions to reduce the number of repeated regular expressions, Lee and Shallit give improved upper bounds on $R_k(n)$:

Theorem 17. *The function $R_k(n)$ satisfies the upper bounds in Table 1, where the length of the regular expressions is ordinary length.*

k	1	2	3	4	5	6
$R_k(n)$	$O(2.9090^n)$	$O(4.2198^n)$	$O(5.3182^n)$	$O(6.4068^n)$	$O(7.4736^n)$	$O(8.5261^n)$

Table 1: Upper bounds on $R_k(n)$ for $1 \leq k \leq 6$.

Lee and Shallit also give upper bounds for $R_k(n)$ using reverse polish and alphabetic length, as well as establish lower bounds on $R_k(n)$ [21]:

Theorem 18. *The function $R_k(n)$ satisfies the lower bounds in Table 2, where the length of the regular expressions is ordinary length.*

k	1	2	3	4	5	6
$R_k(n)$	$\Omega(1.3247^n)$	$\Omega(2.7799^n)$	$\Omega(3.9582^n)$	$\Omega(5.0629^n)$	$\Omega(6.1319^n)$	$\Omega(7.1804^n)$

Table 2: Lower bounds on $R_k(n)$ for $1 \leq k \leq 6$.

Again, lower bounds for reverse polish and alphabetic length are also given. The bounds in Table 2 are obtained by explicitly constructing large sets of distinct regular expressions of the given length. We note that Lee and Shallit also give bounds on the number of star-free and finite languages accepted by regular expressions of a given length.

7 Enumeration of ω-regular Languages

Finite automata recognizing infinite words are a classic model of study in the field of formal language theory. For an introduction to automata on infinite words see, e.g., Pin and Perrin [28] or Thomas [34]. A one-way infinite word w over the alphabet Σ is a mapping $w : \mathbb{N} \to \Sigma$. Denote $w_i = w(i)$. We view w as a word which has a starting point w_1 and proceeds to the right $w = w_1 w_2 w_3 w_4 \cdots$. The set of all one-way infinite words is denoted Σ^ω.

One model for accepting the ω-regular languages (i.e., the sets of one-way infinite words recognized by a regular expression involving the operator X^ω) are Müller automata. A (deterministic) Müller automaton M is given by a 5-tuple $M = (Q, \Sigma, \delta, q_0, \mathcal{F})$ where Q is a finite set of states, Σ is the alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state and $\mathcal{F} \subseteq 2^Q$ is the acceptance table. For any infinite word $w \in \Sigma^\omega$, w is accepted by a Müller automaton M if, when starting in the initial state, the set of states visited by w infinitely often is an element of $\mathcal{F}$. As usual, the language accepted by M is the set of all words accepted by M.

Let $f_k^{(\omega)}(n)$ be the number of distinct ω-regular languages accepted by a deterministic Müller automata with n states over a k-letter alphabet. Domaratzki [5] has given upper and lower bounds on the number of ω-regular languages accepted by Müller automata:

Theorem 19. *For all $k \geq 2$, there exists a constant γ_k depending only on k such that for all $n \geq 3$, the following bound hold:*

$$f_k^{(\omega)}(n) \leq \frac{n^{kn} \gamma_k^{n(1+o(1))} 2^{2^n - n - 1}}{(n-1)!} \cdot \sum_{m=0}^{k} \binom{n}{m}.$$

Further, for all $n > k \geq 2$,

$$f_k^{(\omega)}(n) \geq 2^{2^{n - \lfloor n/k \rfloor - 1} - 1}.$$

Note that the constant γ_k in Theorem 19 is the same as the constant in (2). The upper bound of Theorem 19 is interesting, since it relies on the fact that some of the 2^{2^n} possible acceptance tables, called *strongly inadmissible* acceptance tables, are not valid for any possible assignment of transition functions [5].

8 Enumeration of Context-Free Languages

Domaratzki *et al.* [10] have recently considered enumeration questions for context-free languages. The main stumbling block to counting the number of context-free languages of a given size is the fact that deciding if two context-free grammars are equivalent (i.e., generate the same language) is undecidable. However, this does not preclude that the enumeration of context-free languages of size n is computable as a function of n. But in fact, it does turn out that the function counting the number of CFLs of a given size is uncomputable.

In the following theorem [10], we restrict our attention to descriptional complexity measures that are *well-behaved*. By well-behaved, we mean that the total number of CFGs of any given size is finite. We note that, for instance, the minimal number of nonterminals in any CFG generating a CFL is not a well-behaved descriptional complexity measure, since all finite CFLs are generated by CFGs with one nonterminal.

Theorem 20. *If $c(n)$ is the number of CFLs of size n (for any well-behaved, computable descriptional complexity measure), then $c(n)$ is uncomputable.*

However, despite the fact that the number of CFLs of size n is uncomputable, we can still approximate this quantity. For instance, it can be shown that the number of CFLs generated by a CFG in Chomsky Normal Form with at most n nonterminals over a fixed sized alphabet is $2^{\Theta(n^3)}$ [10, Thm. 7].

8.1 Related Enumeration Results

Theorem 20 can be extended to give a general result on the uncomputability of enumerative functions. In what follows, let X be a recursive language, d be a computable and well-behaved descriptional complexity measure, R be an equivalence relation on X and $g_R(n)$ denote the number of equivalence classes on the elements of measure n in X. Let Σ_k, Δ_k be levels in the arithmetic hierarchy. We have the following result [10]:

Theorem 21. *For any equivalence relation R on X that is complete for Σ_k or for Δ_k, the corresponding function $g_R(n)$ is not computable.*

For instance, Domaratzki *et al.* [10] note the following applications of Theorem 21:

- The number of distinct rational relations defined by nondeterministic finite transducers with n states is uncomputable.
- The number of distinct recursively enumerable languages recognized by Turing machines of size n is uncomputable.

However, not all equivalence relations are captured by Theorem 21. We mention some interesting open problems in this area in Section 9.

9 Open Problems

Enumeration of formal languages has several areas of investigation which are open. We mention some open problems which seem particularly interesting.

We first note some asymptotic bounds that we think might be easily improved. Enumeration of regular languages by nondeterministic state complexity is a very natural problem that has not received much attention. The bounds for many of these problems are likely to be able to be improved. We mention in particular the number of unary regular languages accepted by NFAs with n states as one such open problem. The current best known upper bound is logarithmically $n \log(cn) - n \log\log(n)$ while the best known lower bound is logarithmically $n + (c - o(1))\sqrt{\frac{n}{\log n}}$.

Enumeration of automata accepting ω-regular languages is an interesting area which has received only minimal attention. The unique mode of acceptance for Müller automata presents an interesting enumeration problem, and some results have been obtained by Domaratzki [5]. However, tight bounds have not been obtained, and enumeration of Büchi automata has not been considered. The acceptance mode of Büchi automata yield a distinct notion of equivalence and it would be interesting to give asymptotics for the number of ω-regular languages accepted by Büchi automata with n states.

Theorem 21 gives a general result for proving that several enumerative functions are uncomputable. However, the result is not applicable in all cases. For instance, the following problem is open [10]: Is the number of regular languages generated by CFGs of a fixed size computable?

Recently, measuring the descriptional complexity of regular languages by the minimal number of transitions required by an NFA to recognize a language has received increased attention. This raises the natural question: how many regular languages can be accepted by NFAs with at most n transitions? Gramlich and Schnitger give an upper bound on the number of binary regular languages accepted by an NFA with n transitions: they show that this quantity is at most n^{8n+2} [14]. We can also adapt the result of Liskovets [23] and Robinson [31] of Theorem 3 (see also Domaratzki *et al.* [9] for the case of NFAs) for enumerating the number of labelled, initially-connected NFAs over n states with m transitions. In particular, if $T_k(n,m)$ is the number of initially-connected NFAs with n states and m transitions over a k-letter alphabet (without final states), we can easily show that $T_k(n,m)$ satisfies the following recurrence:

$$\begin{aligned}
T_k(1,1) &= k, \\
T_k(n,m) &= 0 \quad \text{if } n \geq m+2 \text{ and} \\
T_k(n,m) &= \binom{kn^2}{m} - \binom{kn(n-1)}{m} - \sum_{i=1}^{n-1}\sum_{j=1}^{m}\binom{n-1}{i-1}T_k(i,j)\binom{kn(n-i)}{(m-j)}.
\end{aligned}$$

However, tight asymptotic bounds for enumerating regular languages by the number of transitions are unknown.

Enumeration by other descriptional complexity measures is also an area for future research. For instance, the measure of radius [12, 8] has been implicitly studied in relation to the enumeration of finite languages [7] and as descriptional complexity measure [2]. Further, simultaneous enumeration by several descriptional complexity measures has only received some attention in the literature [9, 35]. We feel that there are many interesting avenues of research in the area of enumeration of formal languages.

Finally, we note that explicitly computing values of the functions described here is often challenging for even small values of n. As an example, we note that the values of $G_1(n)$ (the number of unary regular languages accepted by NFAs with n states) is known only for values of $n \leq 6$.

10 Conclusions

Enumeration problems in formal language theory have many applications, and also presents interesting challenges relating to our understanding of the structure of language devices, especially distinctness and minimality. The recent work surveyed here shows that results in enumeration of formal languages often yield enlightening results that further our knowledge of the theory of formal languages in general. Though these fundamental questions have been examined for many years, interesting challenges still remain.

References

[1] Bassino, F., and Nicaud, C. Enumeration and random generation of accessible automata. *Submitted for publication* (2006). Retrieved from `http://www-igm.univ-mlv.fr/~bassino/publications/tcs06.ps`, April, 2006.

[2] Câmpeanu, C., and Ho, W. The maximum state complexity for finite languages. *Journal of Automata, Languages and Combinatorics 9*, 2–3 (2004), 189–202.

[3] Champarnaud, J.-M., and Paranthoën, T. Random generation of DFAs. *Theoretical Computer Science 330*, 2 (2005), 221–235.

[4] Chrobak, M. Finite automata and unary languages. *Theoretical Computer Science 47* (1986), 149–158.

[5] Domaratzki, M. On enumeration of Müller automata. In *Developments in Language Theory: 7th International Conference* (2003), Z. Ésik and Z. Fülöp, Eds., vol. 2710 of *Lecture Notes in Computer Science*, pp. 254–265.

[6] Domaratzki, M. Combinatorial interpretations of a generalization of the Genocchi numbers. *Journal of Integer Sequences 7* (2004), Article 04.3.6.

[7] Domaratzki, M. Improved bounds on the number of automata accepting finite languages. *International Journal of Foundations of Computer Science 15*, 1 (2004), 143–161.

[8] Domaratzki, M., Ellul, K., Shallit, J., and Wang, M.-W. Non-uniqueness and radius of cyclic unary NFAs. *International Journal of Foundations of Computer Science 16*, 5 (2004), 883–896.

[9] Domaratzki, M., Kisman, D., and Shallit, J. On the number of distinct languages accepted by finite automata with *n* states. *Journal of Automata, Languages and Combinatorics 7*, 4 (2002), 469–486.

[10] Domaratzki, M., Okhotin, A., and Shallit, J. Enumeration of context-free languages and related structures. In *Seventh International Workshop on Descriptional Complexity of Formal Systems: Proceedings* (2005), C. Mereghetti, B. Palano, G. Pighizzini, and D. Wotschke, Eds., pp. 85–96.

[11] Domaratzki, M., Pighizzini, G., and Shallit, J. Simulating finite automata with context-free grammars. *Information Processing Letters 84* (2002), 339–344.

[12] Ellul, K. Descriptional complexity measures of regular languages. Master's thesis, University of Waterloo, 2002.

[13] Ellul, K., Krawetz, B., Shallit, J., and Wang, M.-W. Regular expressions: New results and open problems. *Journal of Automata, Languages and Combinatorics 9*, 2–3 (2004), 233–256.

[14] Gramlich, G., and Schnitger, G. Minimizing NFAs and regular expressions. In *STACS 2005* (2005), V. Diekert and B. Durand, Eds., vol. 3404 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 399–411.

[15] Han, G.-N. Escaliers évalués et nombers classiques. *Publ. IRMA Strasbourg, Actes 24e Séminaire Lotharingien* (1993), 77–85.

[16] Harrison, M. A census of finite automata. *Canadian journal of mathematics 17* (1965), 100–113.

[17] Holzer, M., and Kutrib, M. Nondeterministic descriptional complexity of regular languages. *International Journal of Foundations of Computer Science 14*, 6 (2003), 1087–1102.

[18] Korshunov, A. Asymptotic estimates of the number of finite automata. *Cybernetics 3*, 2 (1967), 12–19.

[19] Korshunov, A. A survey of certain trends in automata theory (in Russian). *Diskretnyi Analiz 25* (1974), 19–55.

[20] Korshunov, A. Enumeration of finite automata (in Russian). *Problemy Kibernetiki 34* (1978), 5–82,272.

[21] Lee, J., and Shallit, J. Enumerating regular expressions and their languages. In *Implementations and Application of Automata* (2005), M. Domaratzki, A. Okhotin, K. Salomaa, and S. Yu, Eds., vol. 3317 of *Lecture Notes in Computer Science*, pp. 2–22.

[22] Liksovets, V. Exact enumeration of acyclic deterministic automata. *Discrete Applied Mathematics 154*, 3 (2006), 537–551.

[23] Liskovets, V. The number of connected initial automata. *Cybernetics 5* (1969), 259–262.

[24] Narushima, H. Principle of inclusion-exclusion on semilattices. *Journal of Combinatorial Theory, Series A 17* (1974), 196–203.

[25] Narushima, H. A survey of enumerative combinatorial theory and a problem. *Proceedings of the Faculty of Science of Tokai University 14* (1979), 1–10.

[26] Narushima, H. Principle of inclusion-exclusion on partially ordered sets. *Discrete Mathematics 42* (1982), 243–250.

[27] Nicaud, C. Average state complexity of operations on unary automata. In *Proc. 24th Symposium, Mathematical Foundations of Computer Science 1999* (1999), M. Kutylowski, L. Pacholski, and T. Wierzbicki, Eds., vol. 1672 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 231–240.

[28] Perrin, D., and Pin, J.-E. *Infinite Words*. Elsevier, 2004.

[29] Pomerance, C., Robson, J., and Shallit, J. Automaticity II: Descriptional complexity in the unary case. *Theoretical Computer Science 180* (1997), 181–201.

[30] Reis, R., Moreira, N., and Almeida, M. On the representation of finite automata. In *Seventh International Workshop on Descriptional Complexity of Formal Systems: Proceedings* (2005), C. Mereghetti, B. Palano, G. Pighizzini, and D. Wotschke, Eds., pp. 269–276.

[31] Robinson, R. W. Counting strongly connected finite automata. In *Graph Theory with Applications to Algorithms and Computer Science* (1985), Y. Alavi, G. Chartrand, L. Lesniak, D. Lick, and C. Wall, Eds., pp. 671–685.

[32] Rozenberg, G., and Salomaa, A., Eds. *Handbook of Formal Languages*. Springer-Verlag, 1997.

[33] Sloane, N. *On-line Encyclopedia of Integer Sequences*. Available Electronically at `http://research.att.com/~njas/sequences`, 2006.

[34] Thomas, W. Automata on infinite objects. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Elsevier, 1990, pp. 133–192.

[35] van Zijl, L. Magic numbers for symmetric difference NFAs. *International Journal of Foundations of Computer Science 16*, 5 (2005), 1027–1038.

[36] Yu, S. State complexity of the regular languages. *Journal of Automata, Languages and Combinatorics 6* (2001), 221–234.

[37] Yu, S. State complexity of finite and infinite regular languages. *Bulletin of the European Association of Theoretical Computer Science 76* (2002), 142–152.

The Formal Specification Column

BY

HARTMUT EHRIG

Technical University of Berlin, Department of Computer Science
Franklinstraße 28/29, D-10587 Berlin, Germany
ehrig@cs.tu-berlin.de

Report on ACCAT Workshop at ETAPS 2006: Applied and Computational Category Theory

by Hartmut Ehrig
Technische Universität Berlin, Germany

Category Theory is a well-known powerful mathematical modeling language with a wide area of applications in mathematics and computer science, including especially the semantical foundations of topics in software science and development. Since about 30 years there have been workshops including these topics. More recently, the ACCAT group established by J. Pfalzgraf at Linz and Salzburg has begun to study interesting applications of category theory in Geometry, Neurobiology, Cognitive Sciences, and Artificial Intelligence. It is the intention of this ACCAT workshop to bring together leading researchers in these areas with those in Software Science and Development in order to transfer categorical concepts and theories in both directions. The ACCAT 2006 workshop was organized by Jochen Pfalzgraf and Hartmut Ehrig and took place on March 26, 2006 as satellite of ETAPS 2006 at the Vienna University of Technology. The organizers are representatives of categorical methods for several areas like Geometry, Neurobiology, Cognitive Sciences, and Artificial Intelligence on one hand and Software Science and Development on the other hand. Categorical methods are already well-established for the semantical foundation of type theory (Cartesian closed categories), data type specification frameworks (institutions) and graph transformation (adhesive high level replacement categories), which are most relevant for ETAPS. The organizers have invited leading senior and promising junior researchers for giving invited lectures at the ACCAT workshop which promises to lead to interesting discussions concerning transfer of categorical methods between the areas mentioned above.

After a short opening statement by the organizers Jochen Pfalzgraf continued with an interesting overview of the ACCAT origins. Julia Padberg reported about the integration of two important categorical frameworks, the generic component concept for system modelling and adhesive HLR systems, which is the subject of her habilitation thesis completed recently. After the presentation of a functional framework for constraint normal logic programming by Fernando Orejas we learned about a category theory in Brazil from Ciara Aparecida dos Santos Leal, especially a categorical view of structural complexity.

Andrzej Tarlecki and Till Mossakowski gave a very nice overview about institutions, abstract model theory, heterogeneous specification and the heterogeneous tool set. Motivated by homotopy theory in topology J. Rosicky reported about factorization systems and classification problems. Jose Meseguer discussed in his presentation different aspects of theory morphisms in membership equational logic.

The new topic of adhesive categories and adhesive high-level replacement systems was used by Ulrike Prange to present general constructions and properties of such categories and Andres Corradini discussed how to use this framework for a categorical semantics of concurrency generalizing that of graph transformation systems in the double pushout approach. Since Jiri Adamek and D. Dubois could not attend ACCAT we had enough time for long discussions of the presentations.

ACCAT will be continued next year as satellite workshop of ETAPS 2007 in Braga, Portugal.

The Logic in Computer Science Column

by

Yuri Gurevich
Microsoft Research
One Microsoft Way, Redmond WA 98052, USA
gurevich@microsoft.com

Henkin quantifiers: logic, games, and computation[*]

Merlijn Sevenster
ILLC, Universiteit van Amsterdam
sevenstr@science.uva.nl

Abstract

In this paper, we study the game-theoretic and computational repercussions of Henkin's partially ordered quantifiers [19]. After defining a game-theoretic semantics for these objects, we observe that tuning the parameter of absentmindedness gives rise to quantifier prefixes studied in [28]. In the interest of computation, we characterize the complexity class $\mathrm{P}_{\shortparallel}^{\mathrm{NP}}$ in terms of partially ordered quantifiers, by means of a proof different from Gottlob's [17]. We conclude with some research questions at the interface of logic, game theory, and complexity theory.

[*]We gratefully acknowledge Johan van Benthem, Peter van Emde Boas, Yuri Gurevich, Marcin Mostowski, Eric Pacuit, Gabriel Sandu, Tero Tulenheimo, and the anonymous reviewer. This paper was finalized at Stanford University, whom we thank for hosting us as a visiting scholar.

1 Setting the stage

Henkin's partially ordered quantifier prefixes were initially introduced as a mathematical exercise [19], but have ever since been the subject of lively discussion in various disciplines. In linguistics, Hintikka [20] and Barwise [4] argued that partially ordered or *branching* quantifiers should be added to the linguist's toolbox to give certain natural language expressions their correct logical form. Sandu and Hintikka [21, 23, 33] imported the idea of a partial dependence relation between quantifiers in first-order logic, resulting in *Independence Friendly logic*. Independence Friendly logic has congenially been given a semantics in terms of *games with imperfect information*. The partiality of information—i.e., imperfect information—present in these games can be seen to reflect the partial ordering of the quantifiers.

In this paper we aim to show some of the repercussions of Henkin's exercise from a game-theoretic and (finite) model-theoretic angle. Game theory has penetrated logic successfully, providing an interactive and goal-oriented viewpoint on concepts in logic. The game-theoretic viewpoint allows us to compare logics in terms of the interaction, goals and knowledge.

In Section 2, we introduce logics with Henkin quantifiers and recall their model-theoretic behavior.

In Section 3, we show that Henkin quantifiers are played by agents with a limited number of memory cells, whereas first-order logic is played by Eloise enjoying an infallible faculty of memory.

In Section 4, we give a finite model-theoretic account of Henkin quantifiers. Finite model theory is the model-theoretic face of complexity theory, and provides a neat algorithmic view on Henkin quantifiers.

Section 5 concludes the paper.

2 Logic

Henkin's novelty in the theory of quantification is nowadays known under the header of *Henkin quantifier*. A Henkin quantifier is a two-dimensional object of the form

$$\begin{pmatrix} \forall x_{11} & \dots & \forall x_{1k} & \exists y_1 \\ \vdots & \ddots & \vdots & \vdots \\ \forall x_{n1} & \dots & \forall x_{nk} & \exists y_n \end{pmatrix}, \tag{1}$$

where $\mathbf{x}_i = x_{i1}, \dots, x_{ik}$. Henceforth, a string of variables is referred to by using the obvious symbol boldfaced.

With every Henkin quantifier (1) is associated its *dimensions* n and k. In the interest of space, we abbreviate the Henkin quantifier (1) as $\mathsf{H}^k_n\mathbf{xy}$. If no confusion threatens, we may also skip the variables and the integers indicating the dimensions, and simply write H^k_n and H. To identify a Henkin quantifier without referring to its dimensions, we write $\mathsf{H}_{(i)}$.

The two-dimensional way of representation aims to convey that the variable y_i depends on $\mathbf{x}_i$ and on $\mathbf{x}_i$ only. This is formalized by means of the notion of *Skolem function*, that underlies its semantics. Let $\models$ be the satisfaction relation properly defined for the formula $\phi(\mathbf{x}, \mathbf{y})$ on the structure $\mathfrak{A}$. Then, $\models$ is extended in the following way: $\mathfrak{A} \models \mathsf{H}^k_n\ \phi(\mathbf{x}, \mathbf{y})$ iff there exist k-ary functions $f_1, \dots, f_n$ on the universe of $\mathfrak{A}$ such that

$$\mathfrak{A} \models \forall \mathbf{x}_1 \dots \forall \mathbf{x}_n\ \phi(\mathbf{x}_1, \dots, \mathbf{x}_n, f_1(\mathbf{x}_1), \dots, f_n(\mathbf{x}_n)).$$

Note that the quantifier $\mathsf{H}^0_1 x$ has the same semantics as the quantifier $\exists x$, and that H^0_i is elementary definable for every $i \geq 1$.

In this paper, we are interested in two logics featuring Henkin quantifiers. The first one, denoted **H**, contains all strings of the form $\mathsf{H}^k_n\ \psi$, where ψ is first-order and n and k are arbitrary integers. The second logic's formulae are generated by the following grammar:

$$\phi \quad ::= \quad \psi \mid \neg\phi \mid \phi \vee \phi \mid \exists x\ \phi \mid \mathsf{H}^k_n\ \phi,$$

where ψ is first-order and n and k are arbitrary integers. Let us refer to the latter logic by $\mathbf{H}^*$.

For a thorough introduction to the logics **H** and $\mathbf{H}^*$ and their model-theoretic behavior, we refer the reader to [27].

The set of free variables $\mathit{Free}(\phi)$ in the $\mathbf{H}^*$-formula ϕ is inductively defined by the clauses that define the set of free variables $\mathit{Free}(\psi)$, for first-order ψ, plus the clause

$$\mathit{Free}\left(\mathsf{H}^k_n x_{11} \dots x_{nk} y_1 \dots y_n\ \phi\right) \quad = \quad \mathit{Free}(\phi) - \{x_{11}, \dots, x_{nk}, y_1, \dots, y_n\}.$$

An $\mathbf{H}^*$-formula without free variables is called a *sentence*. The satisfaction relation for formulae with free variables is defined in the standard way using assignments.

As an illustration of the expressive power of **H**, consider the following sentence:

$$\zeta \quad = \quad \begin{pmatrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{pmatrix} \exists z_1 \exists z_2 \exists z_3\ (\phi_1 \wedge \phi_2 \wedge \phi_3),$$

where

$$\begin{aligned} \phi_1 &= (x_1 = x_2) \rightarrow (y_1 = y_2) \\ \phi_2 &= R(x_1, x_2) \rightarrow (y_1 \neq y_2) \\ \phi_3 &= \bigwedge_{i \in \{1,2\}} \bigvee_{j \in \{1,2,3\}} (y_i = z_j). \end{aligned}$$

Let $\mathfrak{G}$ be a graph whose edges interpret R. Then by definition, $\mathfrak{G} \models \zeta$ iff for two unary functions f_1 and f_2 on the universe of $\mathfrak{G}$, (1) f_1 and f_2 are the same; (2) if x_1 and x_2 are joined by an edge, then $f_1(x_1) \neq f_2(x_2)$; and (3) the range of f_1 and f_2 is restricted to z_1, z_2, z_3. All in all, we see that f_1 (or f_2 for that matter) is a witness of $\mathfrak{G} \models \zeta$ iff it is a *3-coloring* of $\mathfrak{G}$.

The semantics for Henkin quantifiers overtly mentions functions, that reflect the (in)dependence relation between the universal and existential quantifiers carried by the Henkin quantifier. Bearing this in mind, it is quite straightforward to show that the truth condition of any sentence from **H** can be expressed in second-order, existential logic, symbolically Σ^1_1. It is a milestone result in the theory of partially ordered quantification that the converse holds as well: When it comes to expressive power, **H** and Σ^1_1 are equivalent, cf. [10, 44]. It was shown [10] that every $\mathbf{H}^*$-sentence is equivalent to a sentence in Σ^1_2 and a sentence in Π^1_2. This finding renders $\mathbf{H}^*$ translatable into Δ^1_2. Mostowski [29] showed that the converse does not hold: there is a sentence in Δ^1_2 that has no equivalent in $\mathbf{H}^*$. These results invariably apply to structures of arbitrary cardinality. In case one restricts oneself to finite ordered structures, a very nice computational characterization of $\mathbf{H}^*$ can be given, see [17] and also Section 4 of the current publication.

3 Games

There is a respectable tradition in logic to give game-theoretic accounts of concepts in logic. An early case in point are Lorenzen-style *dialogue games*. They are typically two-player games between Proponent and Opponent. Dialogue games aim to give a game-theoretic underpinning of the concept of proof. That is, a formula ϕ is provable in a logical system if, and only if, the Proponent has a way of playing the dialogue game of ϕ for the logical system at hand that wins against every way of playing by Opponent. In the game-theorist's parlance, we say that Proponent has a *winning strategy*.

So-called *game-theoretic semantics* was introduced by Hintikka giving a game-theoretic account of truth. For instance, consider a toy fragment of first-order logic, containing only strings of the form

$$\mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n \; R(\mathbf{x}),$$

where $\mathsf{Q}_i \in \{\exists, \forall\}$. Being a fragment of first-order logic, Tarski-semantics is properly defined for this toy language. But the Tarskian satisfaction relation can also be given a game-theoretic face, yielding games between Eloise and Abelard. To this end, let $\mathfrak{A}$ be a structure that interprets the predicate R and let the *semantic* *game* for a formula ϕ from the toy language on $\mathfrak{A}$ start from the position $\langle \phi, \mathfrak{A} \rangle$. The proceedings of this game are determined by the following game rules:

- If the position is $\langle \exists x_i\ \phi, \mathfrak{A} \rangle$, Eloise picks an object a_i from the universe of $\mathfrak{A}$, and the game continues as $\langle \phi, \mathfrak{A} \rangle$.

- If the position is $\langle \forall x_i\ \phi, \mathfrak{A} \rangle$, Abelard picks an object a_i from the universe of $\mathfrak{A}$, and the game continues as $\langle \phi, \mathfrak{A} \rangle$.

- If the position is $\langle R(x_1, \ldots, x_n), \mathfrak{A} \rangle$, the game ends. Eloise wins if the tuple $\langle a_1, \ldots, a_n \rangle$ that was built up during the game stands in the R-relation in $\mathfrak{A}$; otherwise Abelard wins.

The adequacy of the semantic games for the toy language, is typically cast as follows. For every formula ϕ and suitable structure $\mathfrak{A}$, $\mathfrak{A} \models \phi$ iff Eloise has a *winning strategy* in the semantic game of ϕ on $\mathfrak{A}$. As the reader acknowledges if we extend the toy language with connectives, negations, one also has to extend the set of game rules (and possibly tweak the current one) to maintain adequacy of the game-theoretic semantics with respect to the new logical system.

By this token it becomes clear that classes of semantic games should not be conceived of as objects floating in limbo. Just as one can compare the properties of two logical systems by means of model-theoretic means, one can compare the semantic games they give rise to. Again, Lorenzen's dialogue games are a case in point. A lively debate was held about the viability of the dialogue games for first-order logic in contradistinction to the dialogue games for Brouwer's intuitionistic logic. Some held the conviction that dialogue games for intuitionistic logic are 'more natural' than the ones for first-order logic, and took this as an argument in favor of Brouwer's system, cf. [41].

From the same viewpoint, the move from first-order logic to Independence Friendly logic can be appreciated. From a purely game-theoretic angle, Hintikka and Sandu [21, 23, 33] generalized the semantic games for first-order logic so as to incorporate imperfect information. In our view, this very argument may count as a motivation for Independence Friendly logic in itself. What exactly is the influx of the imperfect information in semantic games for Independence Friendly logic is a hard question, and definitely a topic for future research. As we pointed out, the idea of partial dependence relation over quantifiers in Independence Friendly logic has its precursor in Henkin's work. So from this angle alone it is worthwhile to develop at least some understanding of the game-theoretic face of Henkin quantifiers, involving imperfect information.

For the sake of simplicity let us restrict ourselves to **H**-formulae in which the first-order part is atomic. On this assumption, the game rules for the semantic game of the **H**-sentence

$$\mathsf{H}^k_n x_{11} \ldots x_{nk} y_1 \ldots y_n\ R(\mathbf{x}, \mathbf{y})$$

are simply the ones for the semantic game for

$$\forall x_{11} \ldots \forall x_{1k} \exists y_1 \ldots \forall x_{n1} \ldots \forall x_{nk} \exists y_n\ R(\mathbf{x}, \mathbf{y}).$$

But as the latter sentence is a sentence from our toy language, the game seems to have become a game with perfect information as before.

How can this be?

On second thought, it turns out that we have been a bit careless when introducing the semantic games for the toy language. Surely we gave the players eloquent names, but omitted to specify the players are such that the semantic games in which they participate would actually be modeled as games with *perfect information*. It would have made little sense, for instance, to declare that we think of Eloise as a cauliflower. It's not that cauliflowers cannot be regarded as game-theoretic agents, witness the literature on evolutionary game theory. Rather, had we done so modeling the semantic game of a formula from the toy language as a game with perfect information would be counterintuitive, to say the least.

To be on the safe side, we'd better postulate that Eloise has an infallible faculty of memory.

The imperfect information in semantic games for **H**-sentences $\mathsf{H}^k_n\ R(\mathbf{x})$ can be seen to be brought into being by assuming that each agent has exactly k memory cells. Henceforth, we shall assume that agents govern these memory cells in a *first in first out* manner. In unison, these assumption imply that when the agent is deciding on an object for y_i, it knows only the objects picked up over the k previous rounds, that is, the objects assigned to $x_{i1}, \ldots, x_{ik}$. Furthermore, I postulate that this agent is not *absentminded*, that is, it knows in which round of the game it is. This postulate implies that when choosing an object to assign to y_i the agent knows that the object selected will be assigned to y_i and not to, say, y_{i+1} or y_{i-1}.

In this manner, every **H**-sentence $\phi = \mathsf{H}^k_n\ R(\mathbf{x})$ and structure $\mathfrak{A}$ give rise to a semantic game that would be modeled as an *extensive game with imperfect information*, call it *Sem-game*$^{\mathbf{H}}(\phi, \mathfrak{A})$. An extensive game with imperfect information is a rigorous mathematical object $\langle N, H, P, \langle \mathfrak{I}_i \rangle_{i \in N}, W \rangle$, well-known from game theory [30]. N is the set of *players*. H is the set of *histories*—all permissible sequences of actions in the game. P is the *player function* deciding which player $P(h) \in N$ is to move at history h. $\mathfrak{I}_i$ is a partition of the histories in which player i is to move, modeling the imperfect information. W is the *win function*, that decides who has won when the game has come to an end.

In the context of ϕ and $\mathfrak{A}$, the set H equals

$$\bigcup_{0 \le i \le ((n \cdot k)+n)} A^i,$$

where A is the universe of $\mathfrak{A}$. With every history $h \in H$ of length $((n{\cdot}k) + n)$—i.e., *terminal history*—we straightforwardly associate an assignment function $\mathbf{a}_h$ to the variables $x_{11}, \ldots, x_{nk}, y_1, \ldots, y_n$.

Sem-game$^{\mathbf{H}}(\phi, \mathfrak{A})$ can be regarded as a tree structure—a *game tree*—defined by the prefix relation on H. The game tree is decorated by P.

The set $\mathfrak{I}_i$ contains all sets of histories that are indistinguishable for our k-cell, non-absentminded agent (first in first out, remember). The particulars of the agent at hand uniquely determine $\mathfrak{I}_i$. That is, $h, h' \in I \in \mathfrak{I}_i$ if, and only if, h and h' are equally long (non-absentmindedness) and the last k elements of h and h' coincide (k-cell and first in first out). Clearly, $W(h) =$ Eloise iff $\langle \mathbf{a}_h(x_{11}), \ldots, \mathbf{a}_h(x_{nk}), \mathbf{a}_h(y_1), \ldots, \mathbf{a}_h(y_n)\rangle$ is an R-tuple in $\mathfrak{A}$.

Any function $S : \mathfrak{I}_i \to A$ is a strategy for player i in *Sem-game*$^{\mathbf{H}}(\phi, \mathfrak{A})$. Say that a strategy for player i is *winning*, if i following the strategy at each of i's moves only results in terminal histories h such that $W(h) = i$,

Let $\phi = \mathsf{H}^k_n\, R(\mathbf{x})$ and let $\mathfrak{A}$ be a structure interpreting R. Let *Sem-game*$^{\mathbf{H}}(\phi, \mathfrak{A})$ be the extensive game with imperfect information modeling the semantic game of ϕ on $\mathfrak{A}$ played by a k-memory cell agent.

Proposition 1. *For every* **H**-*sentence* $\phi = \mathsf{H}^k_n\, R(\mathbf{x})$ *and structure* $\mathfrak{A}$ *interpreting* R, *a non-absentminded agent with k memory cells has a winning strategy in the semantic game Sem-game*$^{\mathbf{H}}(\phi, \mathfrak{A})$ *iff* $\mathfrak{A} \models \phi$.

Proof. The proof is straightforward once one notices that a series of Skolem functions $f_1, \ldots, f_n$ witnessing $\mathfrak{A} \models \phi$ encodes a winning strategy in *Sem-game*$^{\mathbf{H}}(\phi, \mathfrak{A})$, and vice versa. □

It was observed in [27, pg. 223] that many **H**-sentences appearing in the literature express the existence of *one* single function on the universe. The sentence ζ that expresses 3-colorability of graphs we discussed earlier is a case in point. In the same vein many other interesting **H**-sentences sit in a certain fragment of **H**, that was studied in [28]. This particular fragment is defined by the *function quantifier* F^k_n, that binds the variables $x_{11}, \ldots, x_{nk}, y_1, \ldots, y_n$, just like the Henkin quantifier with dimensions n and k. (We will adhere to the same notational conventions as with Henkin quantifiers.) The logic **F** is defined to be the language containing all strings (sentences) of the form

$$\mathsf{F}^k_n\, \mathbf{x}_1 \ldots \mathbf{x}_n y_1 \ldots y_n\, R(\mathbf{x}_1, \ldots, \mathbf{x}_n, y_1, \ldots, y_n), \tag{2}$$

where $\mathbf{x}_i = x_{i1}, \ldots, x_{ik}$ as before and R is an atom. As regards its semantics, any formula (2) is true on a structure $\mathfrak{A}$ interpreting R iff there exists *one single k*-ary function f on the universe of $\mathfrak{A}$ such that

$$\mathfrak{A} \models \forall \mathbf{x}\, R(\mathbf{x}_1, \ldots, \mathbf{x}_n, f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)).$$

Henkin quantifiers differ from function quantifiers in that the former allow for multiple functions $f_1, \ldots, f_n$, whereas function quantifiers allow for only one. For a model-theoretic comparison of logics with Henkin quantifiers and function quantifiers see [18, 28].

From a game-theoretic point of view, we show that the move from Henkin quantifiers to function quantifiers resembles to imposing absentmindedness on our k-cell agent playing according to the game rules of the semantic game of $\forall \mathbf{x}_1 \exists y_1 \ldots \forall \mathbf{x}_n \exists y_n \, R(\mathbf{x}, \mathbf{y})$ on $\mathfrak{A}$. So in particular the game rules for the sentence

$$\psi \quad = \quad \begin{pmatrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{pmatrix} R(\mathbf{x}, \mathbf{y}),$$

where

$$R(\mathbf{x}, \mathbf{y}) \quad = \quad (x_1 = x_2 \rightarrow y_1 = y_2) \wedge (y_1 = x_2 \rightarrow y_2 = x_1) \wedge (x_1 \neq y_1).$$

on the structure $\mathfrak{B}$ would be equal to the ones for $\forall x_1 \exists y_1 \forall x_2 \exists y_2 \, R(\mathbf{x}, \mathbf{y})$. (The sentence ψ characterizes the finite structures whose universes have even cardinality, see [35].) Considering an absentminded 1-cell agent, we see that during neither of his rounds it knows whether the object it choses will be assigned to y_1 or y_2; it is aware of the last action though. So in particular if a, b, c are three different objects from the universe of $\mathfrak{B}$, it cannot tell apart the histories $\langle a, b, c \rangle$ and $\langle c \rangle$. On the other hand it can distinguish $\langle c \rangle$ from $\langle a \rangle$ and $\langle c, b, a \rangle$.

Just as we had with $\mathbf{H}$, if ϕ is an $\mathbf{F}$-sentence let *Sem-game*$^{\mathbf{F}}(\phi, \mathfrak{A})$ be the extensive game with imperfect information that models an absentminded agent with k memory cells in the latter game. In particular in *Sem-game*$^{\mathbf{F}}(\psi, \mathfrak{B})$ there is an information partition containing both $\langle a, b, c \rangle$ and $\langle c \rangle$, but not $\langle a \rangle$ and $\langle c, b, a \rangle$. Generally speaking, in these extensive games with imperfect information for $\mathbf{F}$, two histories h and h' sit in the same information partition, if the last k elements in h and h' coincide. However as we saw before, h and h' need not be of equal length.

Proposition 2. *For every $\mathbf{F}$-sentence $\phi = \mathsf{F}^k_n \, R(\mathbf{x})$ and structure $\mathfrak{A}$ interpreting R, an absentminded agent with k memory cells has a winning strategy in the semantic game Sem-game$^{\mathbf{F}}(\phi, \mathfrak{A})$ iff $\mathfrak{A} \models \phi$.*

The reader may wonder, what's next? Well, in the same vein one may restrict the agent's powers to an even greater extent and supply it with a fixed array of actions. Recall that in the semantic games for $\mathbf{H}$ and $\mathbf{F}$ the agents pick up their actions from the universe of the structure at hand, that has unbounded cardinality. If we consider the agent to be non-absentminded and in possession of a fixed and finite number of actions, it is capable of 'playing Henkin quantifiers with restricted quantifiers', see [5, 34, 35]. To the best of my knowledge the logic that is played by absentminded agents with a limited number of memory cells and a fixed number of actions has not been studied.

In semantic games for **H**, the k-cell agent is supposed to recall only the last k variables. This undoubtedly is an assumption without theoretical backing. Hintikka and Sandu [21, 23, 33] overcome this needless restriction by introducing the / item in first-order logic, to indicate knowledge of a variable or absence thereof. The resulting system is the Independence Friendly logic we spoke of earlier. In this logic, the sentence

$$\forall x_1(\exists y_1/\{x_1\})\forall x_2(\exists y_2/\{x_2\})\ R(\mathbf{x},\mathbf{y})$$

gives rise to games in which Eloise does not know x_1 when deciding for y_1; but she recollects it when she is to decide for y_2. Given the syntactic formation rules of Independence Friendly logic, one infers rather straightforwardly that every *pattern of ignorance* concerning objects previously played can be accounted for. That is, if we have a first-order formula ϕ in whose semantic games the occurrence of $\exists x$ triggers a move for Eloise informed about $x_1, \ldots, x_n$, then the / item allows one to limit the knowledge of Eloise to any subset of $\{x_1, \ldots, x_n\}$. From this game-theoretic perspective Independence Friendly logic truly is the imperfect information generalization of first-order logic. But note that some sentences from Independence Friendly logic give rise to games that are hard to actually play, as they violate *perfect recall*, cf. [8, 40, 41]. A perfect information approach to Independence Friendly logic was pursued in [38].

Even more delicate flows of information were studied in the *Partial Information logic* by Parikh and Väänänen [32] whose formulae give rise to imperfect information games in which Eloise may be partially informed about the previous actions. In semantic games for the first-order formula $\forall x\exists y\ R(x,y)$, for instance, Eloise knows the object assigned to x. In Partial Information logic, the formula $\forall x(\exists y /\!/ f(x))\ R(x,y)$ typically gives rise to a semantic game in which Eloise is not aware of x, but she is cognizant of $f(x)$. So in case the function f maps every object on x itself Eloise is aware of x after all. But f may just as well return 1 if x is even and 0 otherwise. In this manner, if P is a predicate, the formula $\chi = \forall x(\exists y /\!/ P(x))\ (x \neq y)$ gives rise to games in which Eloise does not know x, but she knows whether Abelard chose a P-object. The formula χ can thus be seen true on any structure in which there is a P-object and a non-P-object. Under specific conditions on the nature of the functions appearing at the right-hand side of the // device, Partial Information logic is a decidable fragment of first-order logic.

It has been pointed out by various authors [22, 24, 41] that we are not really interested in the actual game playing of semantic games. To the ends we employ them it is very much indifferent what strategy is used, for instance, and whether the game is actually played in a platonic universe. Instead we are interested in the statements we can truthfully make *about* these games, in particular in the existence of winning strategies. There is one viewpoint from which this difference becomes clear, that we will highlight. There is a discrepancy between the complexity of the players and the complexity of the statements we make about them, or—more precisely—the expressive power of the logic required to express the winning conditions of Eloise. We saw that Eloise enjoys an infallible faculty of memory in the semantic games for first-order logic, or the toy fragment thereof. Yet, ipse facto, it takes the first-order sentence ϕ to express whether Eloise has a winning strategy in the semantic game of ϕ on any structure. On the other hand, we hired an agent with a limited number of memory cells to play the semantic games for **H**. As was pointed out in [10, 44], here we have to resort to the expressive power of full Σ^1_1!

Note that such a discrepancy does not always occur. For instance, limit attention to 0-cell agents, i.e., agents that don't see any of their opponent's actions. Then, Henkin quantifiers that are playable by such an agent look like

$$\begin{pmatrix} \exists y_1 \\ \vdots \\ \exists y_n \end{pmatrix},$$

and are clearly defined by the first-order prefix $\exists y_1 \dots \exists y_n$.

There is no a priori reason to stick to expressive power as the single measure of complexity. Van Benthem [39] takes up the axiomatization of game models with imperfect information, and needs *extra* axioms to enforce perfect information. Yet the axiom system seems to get more simple when k-cell agents are considered.

4 Computation

Fagin [11] gave birth to the area of descriptive complexity, revealing an intimate connection between model theory and the theory of computation. Descriptive complexity concerns itself with connecting up logical languages and complexity classes. This enterprise departs from the insight that with every logical sentence there is a computational cost associated to verifying its semantic value on an arbitrary finite structure; and the other way around, that the particulars of a computing device can be described in logic. The hope is that hard questions from complexity theory (think of P versus NP and NP versus coNP) can be solved by separating the logics they are associated with, see also Section 5.

In this section we will take up the descriptive complexity analysis of **H***. This will give us an algorithmic view on Henkin quantifiers. Furthermore it gives some insight in the way partially ordered quantifiers manifest themselves in the theory of computation. A more general variant of Theorem 7 from this section appeared in an excellent paper by Gottlob [17]. The references we use do not build on any of Gottlob's results nor on his main references.[1] An independent proof, that is. The descriptive complexity of $\mathbf{H}^*$ was raised as an open problem in [5].

First we give a recap of the basics of finite model theory and descriptive complexity.

Let σ be a finite set of relation symbols—a *vocabulary*—each of which comes with an integer, its *arity*. Every vocabulary contains the binary relation symbol =.

Let a σ-structure $\mathfrak{A}$ be an object of the form $\langle A, \langle R^{\mathfrak{A}} \rangle_{R \in \sigma} \rangle$, where A is the universe of $\mathfrak{A}$ and $R^{\mathfrak{A}} \subseteq A^a$, for a the arity of R. The symbol = is rigidly evaluated as the identity relation. If $< \in \sigma$, then $<^{\mathfrak{A}}$ shall be a linear order on A, and $\mathfrak{A}$ is called a *linear ordered structure*. If A is finite, $\mathfrak{A}$ is called a *finite structure*. Here and henceforth, all discussion will be restricted to finite structures unless indicated otherwise.

Sometimes when we write $\mathfrak{A}$ we actually mean *the binary encoding of* $\mathfrak{A}$. We refer the reader to Immerman's textbook [26], in which a detailed account is given of how one can encode structures in binary. For our ends, it suffices to take notice of the fact that the length of the binary encoding of a σ-structure $\mathfrak{A}$, symbolically $||\mathfrak{A}||$, is of size A^c, for some constant c depending on σ.

Let $\mathcal{K}$ be a class of σ-structures. A *property* Π over $\mathcal{K}$ is a function assigning a truth value $\Pi(\mathfrak{A}) \in \{\mathit{false}, \mathit{true}\}$ to every structure $\mathfrak{A}$ from $\mathcal{K}$. Let $\mathbf{L}$ be a logic, i.e., a set of sentences, for which the satisfaction relation $\models$ is defined. Every $\mathbf{L}$-sentence ϕ defines a property Π_ϕ on $\mathcal{K}$, where

$$\Pi_\phi(\mathfrak{A}) = \mathit{true} \qquad \text{iff} \qquad \mathfrak{A} \models \phi,$$

for every $\mathfrak{A} \in \mathcal{K}$. We say that ϕ and $\mathbf{L}$ *express* Π_ϕ. So the sentence ζ expresses the graph-property of 3-colorability.

Let $\mathbf{L}$ and $\mathbf{L}'$ be two languages over the same vocabulary. Then, write $\mathbf{L} \leq_{\mathcal{K}} \mathbf{L}'$ to indicate that every property over $\mathcal{K}$ expressible in $\mathbf{L}$ is expressible in $\mathbf{L}'$. Define $=_{\mathcal{K}}$ and $<_{\mathcal{K}}$ in the standard way.

Let C be a complexity class [14, 31]. We say that $\mathbf{L}$ *captures at least* C over $\mathcal{K}$, if each C-decidable property over $\mathcal{K}$ can be expressed by a sentence from $\mathbf{L}$ in the vocabulary of σ. We say that the *query complexity* of $\mathbf{L}$ over $\mathcal{K}$ is in C, if for every sentence ϕ in $\mathbf{L}$ in the vocabulary σ, the property Π_ϕ over $\mathcal{K}$ is decidable in C. Here it should be borne in mind, that the size of ϕ is constant. The complexity of Π_ϕ is measured solely by the size of the structures. Finally, say that $\mathbf{L}$ *captures* C over $\mathcal{K}$, if $\mathbf{L}$ captures at least C over $\mathcal{K}$ and the query complexity of $\mathbf{L}$ over $\mathcal{K}$ is in C.

[1] Following the reviewer's suggestion we tag presented proofs of already published results, that differ from the ones given in the literature, with our name.

Descriptive complexity began with *Fagin's Theorem*, in which the complexity class NP is captured.

Theorem 3 ([11]). *Over graphs, Σ^1_1 captures* NP.

The result can be extended so as to hold for arbitrary structures, cf. [26]. Blass and Gurevich [5] drew upon the connection with Henkin quantifiers and obtained that **H** captures NP. This result is readily obtained in virtue of the fact that $\mathbf{H} = \Sigma^1_1$, due to [10, 44]. The remainder of this section is dedicated to the descriptive complexity of $\mathbf{H}^*$.

For future reference, we lay down an easy Prenex normal form result.

Proposition 4. *Every* $\mathbf{H}^*$*-sentence* ϕ *is equivalent to an* $\mathbf{H}^*$*-sentence of the following form:*

$$\pm_1 \mathsf{H}_{(1)}\mathbf{x}_1 \ldots \pm_n \mathsf{H}_{(n)}\mathbf{x}_n\ \psi,$$

where $\pm_i \in \{\neg, \neg\neg\}$ *and* ψ *is first-order.*

Proof. A standard inductive proof suffices, the only non-trivial case being the conjunction. But $\mathsf{H}_{(1)}\mathbf{x}\ \phi_1(\mathbf{x}) \wedge \mathsf{H}_{(2)}\mathbf{y}\ \phi_2(\mathbf{y})$ is easily seen to be equivalent to $\mathsf{H}_{(1)}\mathbf{x}\mathsf{H}_{(2)}\mathbf{z}\ (\phi_1(\mathbf{x}) \wedge \phi_2(\mathbf{z}))$, where $\mathbf{z}$ is a string of variables none of which appear in $\mathbf{x}$. □

Our main observation concerns the computational complexity of $\mathbf{H}^*$, that is associated with the complexity class $\mathrm{P}^{\mathrm{NP}}_{\parallel}$.[2] This denotes the class of properties decidable in deterministic polynomial time with the help of an NP-oracle that can be asked a polynomial number of queries in parallel only once. The action of querying the oracle takes only one time step. Further, P^{NP} contains those problems decidable in deterministic polynomial time with an NP-oracle. Some grasp a complexity class best by its complete problems, that is, its problems to which every problem in the complexity class can be reduced (by means of a polynomial time, many one reduction). Wagner [42] showed that the graph-property of having an odd chromatic number is $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-complete. Denote the class of graphs with an odd chromatic number by ODD-COLOR.

Theorem 5 ([17]). *The query complexity of* $\mathbf{H}^*$ *is in* $\mathrm{P}^{\mathrm{NP}}_{\parallel}$.

Proof (M. Sevenster). It suffices to show that for an $\mathbf{H}^*$-sentence ϕ in the vocabulary σ, deciding whether ϕ is true on a finite σ-structure $\mathfrak{A}$ can be done in $\mathrm{P}^{\mathrm{NP}}_{\parallel}$.

[2]Gottlob's [17] theorem is cast in terms of $\mathrm{LOGSPACE}^{\mathrm{NP}}$, that is, the class of problems decidable in logarithmic space with an NP-oracle. Recall that $\mathrm{LOGSPACE}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{NP}}_{\parallel}$, due to [43].

First we describe an algorithm that computes whether ϕ is true on $\mathfrak{A}$. Thereafter we observe that this algorithm can be implemented on a Turing machine that works in $\mathrm{P}^{\mathrm{NP}}_{\parallel}$.

As for the algorithm, due to Proposition 4 we may assume without loss of generality that ϕ has the form:

$$\pm_1 \mathsf{H}_{(1)}\mathbf{x}_1 \ldots \pm_n \mathsf{H}_{(n)}\mathbf{x}_n \, \psi(\mathbf{x}),$$

where ψ is a first-order formula over the variables $\mathbf{x} = \mathbf{x}_1, \ldots, \mathbf{x}_n$. Let the algorithm start off by writing down all variable assignments in $A^{\mathbf{x}}$, and label every such assignment $\mathbf{a}$ with *true* if $\langle \mathfrak{A}, \mathbf{a}\rangle \models \psi(\mathbf{x})$, and *false* otherwise. Note that consequently ψ's truth conditions on $\mathfrak{A}$ are completely spelled out. Since ψ is first-order this can be done in LOGSPACE.

Put $i = n$ and $\chi_{i+1} = \phi$. For every i from n through 1, proceed as follows for $\pm_i \mathsf{H}_{(i)}\mathbf{x}_i$ in ϕ:

- Write down all assignments in $A^{\mathbf{x}_1,\ldots,\mathbf{x}_{i-1}}$.
- For every assignment $\mathbf{a} \in A^{\mathbf{x}_1,\ldots,\mathbf{x}_{i-1}}$ ask the oracle whether $\langle \mathfrak{A}, \mathbf{a}\rangle \models \mathsf{H}_{(i)}\mathbf{x}_i \, \chi_{i+1}$.
- Label $\mathbf{a}$ with *true* if the answer of the oracle was positive and $\pm_i = \neg\neg$ or the answer was negative and $\pm_i = \neg$; otherwise label it *false*.
- Erase all labeled assignments from $A^{\mathbf{x}_1,\ldots,\mathbf{x}_i}$ and let the current list of assignments fully specify the truth conditions of $\chi_i(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1})$; that is, let χ_i be the formula that holds of an assignment $\mathbf{a}$ on $\mathfrak{A}$ if and only if $\mathbf{a}$ is labeled *true*.

Finally, upon arriving at $n = 0$, if the empty assignment is labeled *true* the algorithm accepts the input; otherwise, it rejects it.

By means of an elementary inductive argument this algorithm can be shown correct.

Apart from consulting the oracle, this algorithm runs in polynomial deterministic time: enumerating all assignments over n iterations takes at most $n \cdot |A^{\mathbf{x}}|$ steps, which is clearly polynomial in the size of the input, $\|\mathfrak{A}\|$, because the number of variables in $\mathbf{x}$ is constant. Since $\mathbf{H}$ captures NP it is sufficient (and necessary) to employ an NP-oracle. This renders the algorithm in P^{NP}, since the number of queries are bounded by the polynomially many different assignments. Yet, this result can be improved, since per iteration the oracle can harmlessly be consulted in parallel. So the algorithm needs a constant number of n parallel queries to the NP-oracle. (Recall that the size of ϕ is constant.) In [6] it was shown that a constant number of rounds of polynomially many queries to an NP-oracle is equivalent to one round of parallel queries. Therefore, the algorithm sits in $\mathrm{P}^{\mathrm{NP}}_{\parallel}$. $\square$

Let $\mathbf{H}^+$ be the *first-order closure* of $\mathbf{H}$. That is, the closure of $\mathbf{H}$ under boolean operations and existential quantification (but not under application of Henkin quantifiers). More formally, $\mathbf{H}^+$ is generated by the following grammar:

$$\phi \quad ::= \quad \psi \mid \neg\phi \mid \phi \vee \phi \mid \exists x\, \phi,$$

where ψ ranges over the $\mathbf{H}$-formulae. The first-order closure of (fragments of) Σ^1_1 was taken up in [2]. In this publication, the authors observe that the first-order closure of Σ^1_1 captures $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$, on linear ordered structures. Since $\mathbf{H} = \Sigma^1_1$, the following result follows directly.

Proposition 6. *Over linear ordered structures,* $\mathbf{H}^+$ *captures* $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$.

It is readily observed from the languages' grammars that every sentence in $\mathbf{H}^+$ is a sentence in $\mathbf{H}^*$ as well. Therefore, for every class of structures $\mathcal{K}$, $\mathbf{H}^+ \leq_{\mathcal{K}} \mathbf{H}^*$. This is actually the last step we have to make to establish the main result.

Theorem 7 ([17]). *Over linear ordered structures,* $\mathbf{H}^*$ *captures* $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$.

Proof (M. Sevenster). Let $\mathcal{L}$ denote the class of linear ordered structures. By Theorem 5 we have that $\mathbf{H}^*$'s query complexity is in $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$, also over $\mathcal{L}$. It remains to be proved therefore that $\mathbf{H}^*$ captures at least $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$. To this end, let Π be an arbitrary $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$-decidable property over $\mathcal{L}$. In virtue of Proposition 6, we obtain that there is a sentence ϕ from $\mathbf{H}^+$ that expresses Π over $\mathcal{L}$. As we concluded right before this theorem, for every class of structures $\mathcal{K}$, $\mathbf{H}^+ \leq_{\mathcal{K}} \mathbf{H}^*$. So in particular it is the case that $\mathbf{H}^+ \leq_{\mathcal{L}} \mathbf{H}^*$. Whence, Π is expressible in $\mathbf{H}^*$ as well, and the claim follows. □

We wish to warn the reader who is about to jump to conclusions about parallel computation and partially ordered quantification. Admittedly, the complexity class $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$ is based on parallel Turing machines and it is captured by $\mathbf{H}^*$, on linear ordered structures. However, this does not mean that verifying a single $\mathbf{H}$-formula $\mathsf{H}\mathbf{x}\, \phi$ can be done by parallel means, as this requires 'simply' an NP-machine. The parallel way of computing comes in effect only when we compute the semantic value of several $\mathbf{H}$-formulae at the same moment in time. For instance, if $\mathsf{H}\mathbf{x}\, \phi(y)$ is an $\mathbf{H}$-formula with one free variable y, then verifying all of

$$\langle \mathfrak{A}, a_1 \rangle \models \mathsf{H}\mathbf{x}\, \phi(y) \quad \ldots \quad \langle \mathfrak{A}, a_m \rangle \models \mathsf{H}\mathbf{x}\, \phi(y)$$

for objects $a_1, \ldots, a_m \in A$, can be done in one round of m parallel queries to an NP-oracle. It is this principle that underlies the fact that $\mathbf{H}^*$'s query complexity is in $\mathrm{P}^{\mathrm{NP}}_{\shortparallel}$.

On the other hand, it is noteworthy that the very fact that a polynomial number of parallel queries suffice is due to the fact that $\mathbf{H}^*$-formulae do only contain first-order variables. This, namely, makes it sufficient to spell out all variable assignments, simply being tuples of objects, and to compute the formula's semantic value with respect to this list. By contrast, if one wishes to verify a second-order formula like $\exists X \forall Y \exists Z\ \phi$ on a structure, spelling out variable assignments amounts to checking triples of *subsets* of tuples of objects. Interestingly, full second-order logic captures the *Polynomial Hierarchy*, whereas $\mathbf{H}^*$ 'gets stuck' at $\mathrm{P}_{\parallel}^{\mathrm{NP}}$. In this sense Theorem 5 provides the computational upper-bound of partially ordered, yet first-order, quantification.

One way to appreciate the fact that the logics $\mathbf{H}^+$ and $\mathbf{H}^*$ coincide on linear ordered structures is by means of the *Henkin depth* of $\mathbf{H}^*$-formulae:

$$\begin{array}{rcl} hd(\phi) & = & 0, \quad \text{for first-order } \phi \\ hd(\neg\phi) & = & hd(\phi) \\ hd(\phi \vee \psi) & = & \max\{hd(\phi), hd(\psi)\} \\ hd(\exists x\ \phi) & = & hd(\phi) \\ hd(\mathsf{H}_n^k \mathbf{x}\ \phi) & = & hd(\phi) + 1, \end{array}$$

reading $\mathsf{H}_n^0 x_1 \dots x_n$ as $\exists x_1 \dots \exists x_n$.

Clearly every $\mathbf{H}^+$-sentence has a Henkin depth of at most one. Therefore, by Theorem 7 we get that for every $\mathbf{H}^*$-sentence ϕ there exists an $\mathbf{H}^+$-sentence ψ, such that $hd(\psi) \leq 1$ and on the class of linear ordered structures ϕ and ψ define the same property. Put differently, on linearly ordered structures granting Henkin quantifiers to nest does not yield greater expressive power. Gottlob proves an even stronger normal form for $\mathbf{H}^*$ on linear ordered structures. In Gottlob's terminology, an $\mathbf{H}^*$-sentence ϕ is in *Stewart normal form*, if it is of the form

$$\exists \mathbf{x}\ \left(\mathsf{H}_{(1)}\mathbf{y}\ \phi_1(\mathbf{x}, \mathbf{y})\ \wedge\ \neg\mathsf{H}_{(2)}\mathbf{z}\ \phi_2(\mathbf{x}, \mathbf{z})\right),$$

where ϕ_1 and ϕ_2 are first-order. This normal form is inspired by the work of Stewart [36, 37], hence the name. Clearly the Henkin depth of every formula in Stewart normal form is at most one. Gottlob proves that on the class of linear ordered structures for every $\mathbf{H}^*$-sentence ϕ there exists an $\mathbf{H}^*$-sentence ψ in Stewart normal form, that expresses the same property.

This result cries out for an effective translation procedure from $\mathbf{H}^*$ into $\mathbf{H}^+$ of course, but unfortunately we cannot provide it. The translation hinges on finding a way of reducing the number of Henkin prefixes in a quantifier block. It gives some insight in the problem to show that

$$\begin{pmatrix} \forall u_1 & \exists v_1 \\ \forall u_2 & \exists v_2 \end{pmatrix} \begin{pmatrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{pmatrix} \phi \tag{3}$$

is equivalent to

$$\left(\begin{array}{cccc} & & \forall u_1 & \exists v_1 \\ & & \forall u_2 & \exists v_2 \\ \forall u_1 & \forall u_2 & \forall x_1 & \exists y_1 \\ \forall u_1 & \forall u_2 & \forall x_2 & \exists y_2 \end{array}\right)\phi, \tag{4}$$

see also [5]. But the real challenge is to find a way to handle negations appearing in between Henkin prefixes, making use of the finiteness of the structure and its linear order.

Dawar, Gottlob, and Hella [7] raise the question whether $\mathbf{H}^*$ captures $P_{||}^{NP}$ over *unordered* structures. Surprisingly, it turns out that $\mathbf{H}^*$ does not capture $P_{||}^{NP}$ in the absence of a linear order, unless the *Exponential Boolean Hierarchy* collapses, amongst other hierarchies. In complexity theory the collapse of this hierarchy is considered to be highly unlikely.

Further still, a study by Hyttinen and Sandu [25] implies that essentially one has to make use of the finiteness of the structures. Consider the logical languages

$$\begin{array}{rcl} \mathbf{H}^+_1 & = & \mathbf{H} \\ \mathbf{H}'_k & = & \text{first-order closure of } \mathbf{H}^+_k \\ \mathbf{H}^+_{k+1} & = & \{\mathsf{H}\mathbf{x}\,\phi \mid \phi \in \mathbf{H}'_k\}. \end{array}$$

Clearly the Henkin depth of any sentence from $\mathbf{H}^+_k$ is k, and $\bigcup_k \mathbf{H}^+_k = \mathbf{H}^*$. The authors prove that on the standard model of arithmetic the language $\mathbf{H}^+_{k+1}$ has strictly stronger expressive power than $\mathbf{H}^+_k$, for every $k \geq 1$.

For the sake of concreteness, consider the property Odd-color over graphs. By Theorem 7, the similar property over linear ordered graphs is expressible in $\mathbf{H}^*$ (and $\mathbf{H}^+$). A linear ordered graph $\mathfrak{G}$ is a structure $\langle G, R^{\mathfrak{G}}, <^{\mathfrak{G}}\rangle$ such that $\langle G, R^{\mathfrak{G}}\rangle$ is a graph and $<^{\mathfrak{G}}$ is a linear order on G. We claim that ξ expresses Odd-color on linear ordered graphs, where ξ is

$$\exists x_1 \exists x_2\ (EVEN(x_2) \wedge SUC(x_1, x_2) \wedge COLOR(x_2) \wedge \neg COLOR(x_1)).$$

In ξ, *EVEN* is the predicate that holds for exactly those objects that are even with respect to $<$, and *SUC* holds for every pair of objects x_1, x_2 such that x_2 is the immediate $<$-successor of x_1. *EVEN* and *SUC* are clearly expressible in Σ^1_1 and consequently in $\mathbf{H}$. Intuitively, *COLOR* holds for all objects x such that the graph at hand is n-colorable, where n is the number of objects $<$-preceding x. Formally, we define $COLOR(x)$ as follows:

$$\left(\begin{array}{cc} \forall y_1 & \exists z_1 \\ \forall y_2 & \exists z_2 \end{array}\right)(y_1 = y_2) \to (z_1 = z_2)\ \wedge\ R(y_1, y_2) \to (z_1 \neq z_2)\ \wedge\ (z_1 < x)\ \wedge\ (z_2 < x),$$

in spirit akin to ζ. We leave it for the reader to check that ξ indeed expresses Odd-color. It is readily observed that ξ can be cast as a $\mathbf{H}^{+}$-sentence, that is not in Stewart normal form. Yet by the Prenex normal form result, Proposition 4, we can extract the Henkin quantifiers from $EVEN(x_2)$, $SUC(x_1, x_2)$ and $COLOR(x_2)$, and obtain an equivalent formula of the form $\mathsf{H}_{(1)}\mathbf{x}\mathsf{H}_{(2)}\mathbf{y}\mathsf{H}_{(3)}\mathbf{z}\ldots$. By merging these, as we got from (3) to (4), we get an equivalent formula with one Henkin quantifier $\mathsf{H}_{(4)}\mathbf{u}\ldots$. The formula that results after replacing $\mathsf{H}_{(4)}\mathbf{u}\ldots$ in ξ is in Stewart normal form.

5 Concluding remarks

As we hoped to have shown, Henkin's idea has exciting manifestations in game theory, model theory, and computational complexity. Each of these manifestations shows a different face of the Henkin quantifier: interaction in the absence of full information, expressive power on formal structures, and algorithmic verification. Our results provide another instance when the disciplines at stake are strongly intertwined. Our Propositions 1 and 2 are cases in point. But admittedly, our approach was not highly systematic. We meandered from non-absentmindedness to absentmindedness, and from partially ordered quantification to Partial Information logic. Improving our understanding of the sparkling interface of logic and game theory is definitely worthwhile.

For instance what kind of game-theoretic underpinning can we give for $\mathbf{H}^{*}$? What does its game-theoretic semantics look like? And can it maybe inspire us to define an *interactive protocol* [16] kind of computing device that computes $\mathrm{P}_{\shortparallel}^{\mathrm{NP}}$? After all, interactive protocols are games with imperfect information.

An intriguing question was raised in [15] related to the finite model theory of Carnap's first-order modal logic $\mathbf{C}$. It is shown that even over finite structures, $\mathbf{C} < \mathbf{H}^{*}$, but what complexity class is actually captured by $\mathbf{C}$ is left as an open question. To this problem we may add the issue of developing a game-theoretic foundation for $\mathbf{C}$.

Finally we mention a game-theoretic gap that needs to be filled in the interest of logic and descriptive complexity. We used the computational result saying that every constant series of parallel queries can be reduced to one session of parallel queries [6]. The logical face of this theorem is the *flatness result*, holding that over linear ordered structures a $\mathbf{H}^{*}$-sentence of arbitrary Henkin depth has an equivalent $\mathbf{H}^{*}$-sentence of Henkin depth at most one. The question arises what would be the game-theoretic face of the aforementioned flatness result, in particular in the realm of *model comparison games* à la Ehrenfeucht and Fraïssé [9, 13]. Model comparison games are typically used to prove that some property is not expressible in a logic. As such they are tools par excellence to separate NP from coNP, for instance. Although considerable progress has been made along these lines [2, 12] the big questions from complexity theory are still unanswered. A fertile approach to prove non-expressibility results is to simplify model comparison games, in order to develop a library of intuitive tools for separating logics, cf. [1, 3]. Along these lines the flatness result concerning Henkin quantifiers may give rise to less complicated, but powerful, games.

References

[1] M. Ajtai and R. Fagin. Reachability is harder for directed than for undirected graphs. *Journal of Symbolic Logic*, 55(1):113–150, 1990.

[2] M. Ajtai, R. Fagin, and L. Stockmeyer. The closure of monadic NP. *Journal of Computer and System Sciences*, 60(3):660–716, 2000.

[3] S. Arora and R. Fagin. On winning strategies in Ehrenfeucht–Fraïssé games. *Theoretical Computer Science*, 174(1–2):97–121, 1997.

[4] J. Barwise. On branching quantifiers in English. *Journal of Philosophical Logic*, 8:47–80, 1979.

[5] A. Blass and Y. Gurevich. Henkin quantifiers and complete problems. *Annals of Pure and Applied Logic*, 32(1):1–16, 1986.

[6] S. R. Buss and L. Hay. On truth-table reducibility to SAT. *Information and Computation*, 91(1):86–102, 1991.

[7] A. Dawar, G. Gottlob, and L. Hella. Capturing relativized complexity classes without order. *Mathematical Logic Quarterly*, 44(1):109–122, 1998.

[8] F. Dechesne. *Game, Set, Maths: Formal investigations into logic with imperfect information*. PhD thesis, SOBU, Tilburg university and Technische Universiteit Eindhoven, 2005.

[9] A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae*, 49:129–141, 1961.

[10] H. B. Enderton. Finite partially ordered quantifiers. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 16:393–397, 1970.

[11] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *SIAM-AMS Proceedings, Complexity of Computation*, volume 7, pages 43–73, 1974.

[12] R. Fagin. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.

[13] R. Fraïssé. Sur quelques classifications des systèmes de relations. Publications Scientifiques, Série A, 35–182 1, Université d'Alger, 1954.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[15] A. Gheerbrant and M. Mostowski. Recursive complexity of the Carnap first order modal logic C. *Mathematical Logic Quarterly*, 52(1):87–94, 2006.

[16] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[17] G. Gottlob. Relativized logspace and generalized quantifiers over finite ordered structures. *Journal of Symbolic Logic*, 62(2):545–574, 1997.

[18] L. Hella. Definability hierarchies of generalized quantifiers. *Annals of Pure and Applied Logic*, 43(3):235–271, 1989.

[19] L. Henkin. Some remarks on infinitely long formulas. In P. Bernays, editor, *Infinitistic Methods. Proceedings of the Symposium on Foundations of Mathematics*, pages 167–183, Oxford and Warsaw, 1961. Pergamon Press and PWN.

[20] J. Hintikka. Quantifiers vs. quantification theory. *Linguistic Inquiry*, 5:153–177, 1974.

[21] J. Hintikka. *Principles of mathematics revisited*. Cambridge University Press, 1996.

[22] J. Hintikka. Hyperclassical logic (a.k.a. IF logic) and its implications for logical theory. *Bulletin of Symbolic Logic*, 8(3):404–423, 2002.

[23] J. Hintikka and G. Sandu. Game-theoretical semantics. In J. F. A. K. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 361–481. North Holland, Amsterdam, 1997.

[24] W. Hodges. Formal aspects of compositionality. *Journal of Logic, Language and Information*, 10(1):7–28, 2001.

[25] T. Hyttinen and G. Sandu. Henkin quantifiers and the definability of truth. *Journal of Philosophical Logic*, 29(5):507–527, 2000.

[26] N. Immerman. *Descriptive Complexity*. Graduate texts in computer science. Springer, New York, 1999.

[27] M. Krynicki and M. Mostowski. Henkin quantifiers. In M. Krynicki, M. Mostowski, and L.W. Szczerba, editors, *Quantifiers: Logics, Models and Computation*, volume I of *Synthese library: studies in epistemology, logic, methodology, and philosophy of science*, pages 193–262. Kluwer Academic Publishers, The Netherlands, 1995.

[28] M. Krynicki and J. Väänänen. Henkin and function quantifiers. *Annals of Pure and Applied logic*, 43(3):273–292, 1989.

[29] M. Mostowski. Arithmetic with the Henkin quantifier and its generalizations. In F. Gaillard and D. Richard, editors, *Seminaire de Laboratoire Logique, Algorithmique et Informatique Clermontoise*, volume II, pages 1–25. 1991.

[30] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[31] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[32] R. Parikh and J. Väänänen. Finite information logic. *Annals of Pure and Applied Logic*, 134(1):83–93, 2005.

[33] G. Sandu. On the logic of informational independence and its applications. *Journal of Philosophical Logic*, 22(1):29–60, 1993.

[34] G. Sandu and J. Väänänen. Partially ordered connectives. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 38(4):361–372, 1992.

[35] M. Sevenster and T. Tulenheimo. Partially ordered connectives and Σ^1_1 on finite models. In A. Beckmann, U. Berger, B. Löwe, and J. V. Tucker, editors, *Proceedings of the 2nd Computability in Europe Conference (CiE 2006), Logical Approaches to Computational Barriers*, volume 3988 of *LNCS*, pages 515–524. Springer, 2006.

[36] I. A. Stewart. Logical characterization of bounded query class I: Logspace oracle machines. *Fundamenta Informaticae*, 18:65–92, 1993.

[37] I. A. Stewart. Logical characterization of bounded query class II: Polynomial-time oracle machines. *Fundamenta Informaticae*, 18:93–105, 1993.

[38] J. Väänänen. On the semantics of informational independence. *Logic Journal of the IGPL*, 10(3):337–350, 2002.

[39] J. F. A. K. van Benthem. Games in dynamic-epistemic logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[40] J. F. A. K. van Benthem. The epistemic logic behind IF games. In R. Auxier, editor, *Jaakko Hintikka*, Library of Living Philosophers. Carus Publishers, 2004.

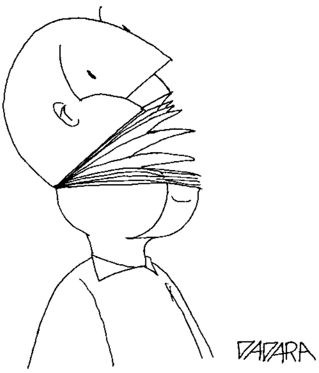
[41] J. F. A. K. van Benthem. Logic and games, lecture notes. Draft version, unpublished.

[42] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51(1–2):53–80, 1987.

[43] K. W. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[44] W. Walkoe. Finite partially-ordered quantification. *Journal of Symbolic Logic*, 35(4):535–555, 1970.

Technical Contributions



Relations over words and logic: a chronology

Christian Choffrut, L.I.A.F.A., Université Paris 7,
2 Pl. Jussieu – 75 251 Paris Cedex – France
Christian.Choffrut@liafa.jussieu.fr

The purpose of this short note is to give credit to the right people who produced original work on the connection between rational relations and logic. Indeed, my experience is that some authors seem to partially ignore the literature or at least neglect to cite it correctly. It is probably due to the fact that language theory and logic, though having largely filled the original gap which separated them, still have different backgrounds. I hope that recalling the chronology might be of some help. If I had some doubt about the necessity of this reminder, a recent experience proved it was justified. Indeed, I posted an early version of the present work on my web page. Kamal Lodaya from the University of Chennai happened to come across it, got interested and posed a few questions. Doing some bibliographical search he found that the relations which after Läuchli and Savioz I had called "special", had in fact been introduced three years earlier by D. Angluin and D. N. Hoover as "regular prefix relations".

Now we come to the point. Given n finite, nonempty alphabets Σ_i, $i = 1, \ldots, n$, I'm interested in the class of subsets, also called *relations*, of the direct product $\Sigma_1^* \times \cdots \times \Sigma_n^*$ which are *rational* (known as *regular* in the anglo-saxon literature). A simple example: the relation which is the graph of the operation of concatenation of two words and which consists of all triples of the form (u, v, uv) where $u, v \in \Sigma^*$, is defined by the rational expression $\Delta_1^*\Delta_2^*$ where $\Delta_1 = \bigcup_{a\in\Sigma}(a, 1, a)$ and $\Delta_2 = \bigcup_{a\in\Sigma}(1, a, a)$. These relations are also defined via an extension of the finite automata operating on tuples of words rather than on words, introduced by Rabin and Scott in the late fifties, [8]. They were studied by Elgot and Mezei who proved most of their general properties, [6]. The main decision issues were settled by Fischer and Rosenberg, [7]. It just happens that this class does not form a Boolean algebra unless $n = 1$ or all alphabets Σ_i's are reduced to a single symbol. Until the mid eighties, only two subclasses of the rational relations were known to be closed under the Boolean operations, to wit the recognizable and the synchronous relations which are therefore natural candidates for logical definability. A new class was discovered by D. Angluin and D. N. Hoover, [1], then rediscovered by Läuchli and C. Savioz.

The reader curious of learning more on rational relations is referred to the standard textbooks, such as [3, 4]. I only fix the notations by saying that the free monoid generated by an alphabet Σ is denoted by Σ^*. An n-ary relation is a subset of $(\Sigma^*)^n$. It is assumed that the reader has a minimum knowledge on Rabin's monadic second order logic of k successors. Finally, the comprehension of the remainder is facilitated if the reader bears in mind the following inclusions of classes of relations whose precise definitions are given in due time, representing respectively the class of recognizable, special, synchronous and rational relations.

$$\text{Rec} \subseteq \text{Spec} \subseteq \text{Sync} \subseteq \text{Rat}$$

The three inclusions are strict exactly under the same conditions that the class of rational relations is not a Boolean algebra.

1969: synchronous relations

Though discovered 36 years ago, the logical characterization of this class of relations due to Eilenberg, Elgot and Shepherdson is practically never cited. Intuitively, it can be described as follows. Given an n-tuple of words $(w_1, \ldots, w_n) \in \Sigma^* \times \cdots \times \Sigma^*$, pad all components by as few occurrences of a new symbol, say # to fix ideas, as necessary in order to achieve equal length, i.e., transform $(abaa, bb, bba)$ into $(abaa, bb\#\#, bba\#)$. Such an n-tuple of words may be viewed as a word over the composite alphabet $\Delta = (\Sigma \cup \{\#\})^n - \{\#\}^n$, e.g., $(abaa, bb\#\#, bba\#)$ can be viewed as the length 4 word $[abb][bbb][a\#a][a\#\#]$

3 tapes

a	b	a	a
b	b		
b	b	a	

converted into 1 tape

a b b	b b b	a # a	a # #

Given a relation $R \subseteq (\Sigma^*)^n$ transform all its n-tuples by padding them as just explained and denote by $R^\# \subseteq \Delta^*$ the resulting subset. Then R is synchronous if there exists a finite automaton on the alphabet Δ which recognizes $R^\#$. E., g., the relation $\{(a^{n+1}, a^n) \mid n \geq 0\} \cup \{(a^{2n}, a^{2n+1}) \mid n \geq 0\}$ is synchronous but the relation $\{(a^{2n}, a^n) \mid n \geq 0\}$ is not. The authors introduce the first order theory

$$\text{Th} = (\Sigma^*, \text{Eq}, \leq_{\text{pref}}, (L_a)_{a \in \Sigma}) \tag{1}$$

where the binary predicate $\text{Eq}(u, v)$ is true if and only if the two words u and v have the same length, the binary predicate $u \leq_{\text{pref}} v$ is true if and only if u is a prefix of v and for each $a \in \Sigma$, the unary predicate $L_a(u)$ is true if and only if the word u ends with the letter a. Observe in passing that this signature is denoted by $\mathbf{S}_{\text{len}}$ in [2] and that this logic, in the context of infinite words, is called chain logic $+ E$ in [10]. The logical characterization is as follows, [5].

Theorem 1. *A subset $R \subseteq (\Sigma^*)^n$ is synchronous if and only if it is definable in the theory (1).*

The sufficiency of the condition is a simple consequence of the closure properties of the synchronous relations under the Boolean operations, composition of relations and projections. The authors prove the converse when the relation is given through a rational expression. Mimicking the automaton yields a much more intuitive proof and can be reconstructed by a good PhD student.

1984: special relations

I discovered this family in a paper of Wolfgang Thomas, [10] where he cited a publication of Läuchli and Savioz. However the correct reference as far as I know is [1]. The merit of this family is that it is a new Boolean class of relations with a neat logical characterization. The starting point is a restriction of the theory of the k successors SkS on the structure $(\Sigma^*, (s_a)_{a\in\Sigma})$ where for each $u \in \Sigma^*$, s_a is interpreted as the function defined by $s_a(u) = ua$ for each $u \in \Sigma^*$. Indeed, a formula $\varphi(x_1, \ldots, x_n)$ with n free individual variables and no free set variable defines an n-ary relation $R \subseteq \Sigma^*$ by setting $R \models \varphi(x_1, \ldots, x_n)$. Furthermore, Läuchli and Savioz introduce the first order theory

$$\mathrm{Th} = (\Sigma^*, 1, \mathrm{LCP}, (P_L)_{L\in \mathrm{Rat}\Sigma^*}) \tag{2}$$

where 1 is the empty word, LCP is the function that assigns the largest common prefix of two words and for all rational subsets $L \in \mathrm{Rat}\Sigma^*$, the predicate $P_L(x, y)$ is true whenever $y \in xL$. This structure is studied under the terminology $\mathbf{S}^+_{\mathrm{reg}}$ in [2]. The theory yields a new subclass of rational relations which is strictly intermediate between the recognizable and the synchronous relations. Indeed, consider a denumerable set of symbols $X = \{x_i\}_{i\in\mathbb{N}}$ and define the least collection C of sequences of strings on X which contains the sequences reduced to one symbol x_i and which satisfies the two conditions

(i) if $(u_1, \ldots, u_p) \in (X^*)^p$ belongs to C and if σ is a permutation on the set $\{1, \ldots, p\}$, then $(u_{\sigma_1}, \ldots, u_{\sigma_p}) \in (X^*)^p$ belongs to C.

(ii) if $(u_1, \ldots, u_p) \in (X^*)^p$ belongs to C and if x_i and x_j are two distinct symbols occurring in none of the u_i's, then $(u_1, \ldots, u_{p-1}, u_p x_i, u_p x_j) \in (X^*)^{p+1}$ belongs to C.

For example, $(x_1x_3, x_1x_2x_4, x_1x_2x_5)$ is of this form, (x_1x_2, x_2x_1) and (x_1x_2, x_3x_4) are not. Finally, a relation R is *special* if there exist a sequence $(u_1, \ldots, u_p)$ in C and a rational subset L_i for each x_i such that R is the set of p-tuples obtained by substituting an arbitrary word of L_i for each occurrence of x_i. As particular cases of special relations, we have the generalized diagonal $\{\overbrace{(u, \ldots, u)}^{n\text{times}} \mid u \in \Sigma^*\}$, all recognizable relations (see below), the relation $\{(uv, uw) \in \Sigma^* \times \Sigma^* \mid u \in \Sigma^*, |v| \equiv 0$

mod 2, $|w| \equiv 1 \mod 3\}$, etc .... I chose the formulation given by Läuchli and Savioz, rather than that of Angluin and Hoover. Indeed, the latter authors do not mention the theory (2). Also instead of the notion of special relations, they give a more complicated, though substantially equivalent, notion of prefix automata.

Theorem 2. *Given a subset $R \subseteq (\Sigma^*)^n$, the following conditions are equivalent*

(i) *R is definable in SkS by a formula having n free variables and no free subset variables*

(ii) *R is definable in the first order theory (2)*

(iii) *R is a finite union of special subsets*

1990: recognizable relations

The reason for this last characterization to remain hidden it that it was published in the proceedings of an ASMICS meeting. Furthermore, it was stated more generally in terms of trace monoids: these are quotients of free monoids by a congruence generated by a reflexive and symmetric relation called *relation of partial commutations $I \subseteq \Sigma\times\Sigma$*. Direct products of free monoids $\Sigma_1^*\times\cdots\times\Sigma_n^*$ are a special case where $I = \bigcup_{i\neq j} \Sigma_i \times \Sigma_j$.

We recall that a subset R of $\Sigma_1^* \times \ldots \times \Sigma_n^*$ is *recognizable* if there exists a finite monoid M and a morphism $f : \Sigma_1^* \times \ldots \times \Sigma_n^* \to M$ such that $R = f^{-1}f(R)$ holds. It is a classical exercise to prove that the class $\mathrm{Rec}(\Sigma_1^* \times \ldots \times \Sigma_n^*)$ of recognizable relations is a Boolean algebra. It can be shown that R is recognizable if and only if it is a finite union of direct products of the form $X_1 \times \ldots \times X_n$ where X_i is a recognizable subset of Σ_i^*, a result which is attributed to Elgot and Mezei by Eilenberg. From a logical viewpoint, the idea is to consider an n-tuple of strings $(w_1, \ldots, w_n)$ as a disjoint union of n linear orders together with two constants for each component representing the first and the last positions and a predicate asserting that a certain position is labeled with a given letter. More precisely, the model theoretic structure is of the form

$$\Big(\bigcup_{1\leq i\leq n} I_i, <, ((Q_a)_{a\in\Sigma_i})_{1\leq i\leq n}\Big) \tag{3}$$

where $I_i = \{(0,i), \ldots, (|w_i|-1, i)\}$ for $i = 1, \ldots, n$. The predicate $(j_1, i_1) < (j_2, i_2)$ is true if and only if $i_1 = i_2$ and $j_1 < j_2$ holds and the predicate $Q_a((j,i))$ is true if and only if the j-th occurrence of w_i is equal to a. For example if $n = 2$, $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{a, c\}$ and $(w_1, w_2) = (babb, aca)$, then the corresponding structure is

$$\begin{tikzcd}
(0,1) \arrow[r, dash] & (1,1) \arrow[r, dash] & (2,1) \arrow[r, dash] & (3,1) \\
(0,2) \arrow[r, dash] & (1,2) \arrow[r, dash] & (2,2) &
\end{tikzcd}$$

We have, e.g., $Q_a(1,1) = \texttt{true}$, $Q_c(2,2) = \texttt{false}$, $(0,1) < (3,1) = \texttt{true}$ and $(2,0) < (1,2) = \texttt{false}$.

We insist that in the previous two cases, the monoid was the model, while here it is each element of the monoid which is a model in itself. Then the following holds, [9].

Theorem 3. *A subset R of is recognizable if and only if it is the set of models of some formula in the theory defined in (3).*

One direction of the proof is standard. The crux for showing that a recognizable relation is expressible through the above logic is the fact that each of the n components of the relation $R \subseteq \Sigma_1^* \times \cdots \times \Sigma_n^*$ can be independently controlled by a finite automaton.

References

[1] D. Angluin and D. N. Hoover. Regular prefix relations. *Mathematical Systems Theory*, 17:167–191, 1984.

[2] M. Benedikt, L. Libkin, T. Schwentick, and L. Ségoufin. Definable relations and first-order query languages over strings. *Journal of the Association of Computing Machinery*, 50:694–751, 2003.

[3] J. Berstel. *Transductions and context-free languages*. B. G. Teubner, 1979.

[4] S. Eilenberg. *Automata, Languages and Machines,* volume A. Academic Press, 1974.

[5] S. Eilenberg, C.C. Elgot, and J.C. Shepherdson. Sets recognized by n-tape automata. *Journal of Algebra*, 13:447–464, 1969.

[6] C. C. Elgot and J. E. Mezei. On Relations Defined by Finite Automata. *IBM Journal,* 10:47–68, 1965.

[7] P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences,* 2:88–101, 1968.

[8] M. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop,* pages 125–144, 1959.

[9] W. Thomas. Logical definability of trace languages. Proceedings of the ASMICS-Workshop TUM-I9002, TU München, 1990.

[10] W. Thomas. Infinite trees and automaton-definable relations over the ω-words. *Theoretical Computer Science*, 103:143–159, 1992.

BINARY WORDS WITH FEW SQUARES

Tero Harju [*] Dirk Nowotka [†]

Abstract

A short proof is given for a result of Fraenkel and Simpson [Electronic J. Combinatorics 2 (1995) #R2] stating that there exists an infinite binary word which has only three different squares u^2.

1 Introduction

Consider the set $\Sigma_k = \{0, 1, \ldots, k-1\}$ as an alphabet of symbols, called letters. The set of all *words* over Σ_k, denoted by Σ_k^*, consists of the finite sequences of elements from Σ_k. We denote by $|w|$ the length of the word w, i.e., the number of occurrences of the letters in w. An *infinite word* over Σ_k is a mapping from $w\colon \mathbb{N} \to \Sigma_k$ which is usually presented as the ordered sequence $w(1)w(2)\cdots$ of the images.

A word u is a *factor* of a word $w \in \Sigma_k^*$, if $w = w_1uw_2$ for some words w_1 and w_2. A nonempty factor u^2 ($= uu$) is called a *square* in w. The word w is *square-free* if it has no squares.

It is easy to see that every binary word $w \in \Sigma_2^*$ with $|w| \geq 4$ has a square. Indeed, the only square-free binary words are $0, 1, 01, 10, 010, 101$. On the other hand, Entringer, Jackson, and Schatz [1] showed in 1974 that there exists an infinite word with 5 different squares. Later Fraenkel and Simpson [2] showed that there exists an infinite binary word over the alphabet $\{0, 1\}$ that has only three squares 00, 11, and 0101. A somewhat simplified proof of this result was given by Rampersad, Shallit, and Wang [4]. We shall give here a still shorter proof of this result based on square-free words over a three letter alphabet.

[*]Turku Centre for Computer Science (TUCS), Department of Mathematics, University of Turku, FIN-20014 Turku, Finland; email: `harju@utu.fi`

[†]Institute of Formal Methods in Computer Science, University of Stuttgart, D-70569 Stuttgart, Germany; email: `nowotka@fmi.uni-stuttgart.de`

Example 1. We can show that the word $w = 101100111000111100111011$ of length 24 has the squares u^2 only for $u \in \{0, 1, 11\}$. Also, it can be checked that the word w has the maximum length 24 among the words having only the squares 0^2, 1^2, and $(1^2)^2$. Indeed, it can be checked that all words of length 25 or more do have three squares 0^2, 1^2, and u^2 for some word $u \notin \{00, 11\}$.

The following result is due to Axel Thue; see, e.g. Lothaire [3] for background and a proof of this theorem.

Theorem 1. *There exists an infinite square-free word over the three letter alphabet* Σ_3.

2 Three squares

In the rest of this paper, we prove

Theorem 2. *There exists an infinite binary word W that has only the squares u^2 for $u \in \{0, 1, 01\}$.*

Proof. Let w be an infinite square-free word over the alphabet Σ_3 provided by Theorem 1. Also, let W be the word where each 0 (1, 2, resp.) in w is substituted by A (B, C, resp.) where

$$\begin{aligned} A &= 1^3 0^3 1^2 0^2 1 0 1^2 0^3 1^3 0^2 1 0\,, \\ B &= 1^3 0^3 1 0 1^2 0^3 1^3 0^2 1 0 1^2 0^3 1 0\,, \\ C &= 1^3 0^3 1^2 0^2 1 0 1^2 0^3 1 0 1^3 0^2 1 0 1^2 0^2\,. \end{aligned}$$

It is easy to check that these words have only the squares 0^2, 1^2 and $(01)^2$. Denote $\Delta = \{A, B, C\}$. We have the following marking property of $1^3 0^3$:

$$\text{the factor } 1^3 0^3 \text{ occurs only as a prefix of each } A,\ B,\ C \tag{a}$$

Also, we notice that the longest common prefix (suffix, resp.) of two words from Δ is $1^3 0^3 1^2 0^2 1 0 1^2 0^3 1$ of length 18 ($0^2 1 0$ of length 4, resp.). Since the words A, B and C are longer than 22, we obtain

$$\begin{aligned} &\text{no word } Z \in \Delta \text{ can be factored as } Z = yx \text{ where } x \text{ is a suffix} \\ &\text{of a word } X \in \Delta \setminus \{Z\} \text{ and } y \text{ is a prefix of a word } Y \in \Delta \setminus \{Z\}. \end{aligned} \tag{b}$$

Suppose that W has a square U^2, where $U \notin \{0, 1, 01\}$. It is straightforward using (a) to verify that the short words XY for different $X, Y \in \Delta$, do not contain other squares than 0^2, 1^2, and $(01)^2$. In particular, we may assume that U^2 has a factor Z from Δ.

Suppose first that U does not have a factor from Δ, and thus that U has at most one occurrence of the marker $1^3 0^3$. Now $U^2 = xZy$ for some $Z \in \Delta$, where x is a suffix of a word $X \in \Delta$ and y is a prefix of a word $Y \in \Delta$ for $X, Y \neq Z$, since w is square-free. We divide our considerations into two cases.

(1) Assume that U has exactly one occurrence of $1^3 0^3$. In this case, $Z = yx$, since y begins with the unique occurrence of $1^3 0^3$ inside the second U. This, however, contradicts the property (b).

(2) Assume that U has no occurrences of $1^3 0^3$. In this case, $U = xu = vy$ so that $Z = uv$, where u and y are proper prefixes of $1^3 0^3$, and as they are both suffixes of U, one of them is a suffix of the other. Necessarily $u = y$, and hence also $x = v$. But now $Z = yx$ contradicts the property (b).

Suppose then that U has a factor from Δ. If $U = uXv$ with $X \in \Delta$, then necessarily $U^2 = uXvuXv$ is a factor in W such that $vu \in \Delta^*$, because of the marking property (a). Therefore, $U = xX_1X_2 \cdots X_n y$ such that $X_0X_1 \cdots X_n Z X_1 \cdots X_n X_{n+1}$ is a factor of W where $X_i \in \Delta$ for each i, x is a suffix of X_0, $Z = yx \in \Delta$, and y is a prefix of X_{n+1}. Now $Z \neq X_0$, since otherwise $(X_0X_1 \cdots X_n)^2$ would be a factor of W and this would contradict the square-freeness of w. Similarly, $Z \neq X_{n+1}$. Again $Z = yx$ contradicts (b). □

We observe that the word

$$w = 10110011100011001000111000100$$

of length 29 has only the squares u^2 for $u \in \{0, 1, 100\}$. It can be shown, with the aid of a computer, that each word w of length $|w| \geq 30$ with exactly three different squares, has the squares 0^2, 1^2 and $(01)^2$ or 0^2, 1^2 and $(10)^2$.

References

[1] R. C. Entringer, D. E. Jackson, and J. A. Schatz. On nonrepetitive sequences. *J. Combin. Theory, Ser. A*, 16(2):159–164, 1974.

[2] A. S. Fraenkel and J. Simpson. How many squares must a binary sequence contain? *Electronic J. Combinatorics*, 2, 1995. Research Paper 2, approx. 9 pp. (electronic).

[3] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics*. Addison-Wesley, Reading, MA, 1983. Reprinted in the Cambridge Mathematical Library, Cambridge Univ. Press, 1997.

[4] N. Rampersad, J. Shallit, and M.-w. Wang. Avoiding large squares in infinite binary words. *Theoret. Comput. Sci.*, 339(1):19–34, 2005.

Fine and Wilf's theorem for abelian periods

Sorin Constantinescu Lucian Ilie[*][†]
Department of Computer Science, University of Western Ontario
N6A 5B7, London, Ontario, CANADA

Abstract

We prove a variant of the well-known Fine and Wilf's periodicity theorem in the case of two relatively prime abelian periods. A result for the case of non-relatively prime abelian periods is conjectured.

Fine and Wilf's theorem. Fine and Wilf's periodicity theorem is one of the most widely used results on words. It was initially proved by Fine and Wilf [10] in connection with real functions but then adopted as a natural result for words, see [6, 13, 14]. We say that a word has a certain integer as period if the word repeats itself after that period; e.g., the word abaabaaba has periods 3, 6, and 8. It is not difficult to see that, given a set of integers, any long enough word which has those periods will have also their greatest common divisor as period. The essential question is how long the word should be. Fine and Wilf's theorem states the optimal length for two periods: it is the sum of the two periods minus their greatest common divisor.

The result has been extended for three periods in [4] and to many periods in [11], the optimal bound for the general case been given in [7] and [19]. Further variants of Fine and Wilf's theorem are given in [1, 3, 5, 15, 2].

Abelian periods. The notion of a period is closely related with that of repetition, already studied by Thue [17, 18]. A repetition is simply a periodic word that appears as a subword in another one. The notion of abelian repetition is well known from the so-called Erdös problem [9] which asked whether abelian squares can be avoided over four letters (affirmatively solved by [12]). An abelian square is an adjacent permuted occurrence of the same string; e.g., abb.bab. Abelian repetitions are obtained accordingly. Our definition below is slightly different from the one of [8] which gives algorithms for computing similar repetitions (called *weak repetitions*).

[*]Research partially supported by NSERC.
[†]Corresponding author; e-mail: `ilie@csd.uwo.ca`

We begin by introducing a few notations. Let $A = \{a_1, a_2, \ldots, a_k\}$ be an alphabet. For a word $w \in A^*$ and a letter $a \in A$, we denote the number of occurrences of a in w by $|w|_a$; the length of w is $|w| = \sum_{i=1}^{k} |w|_{a_i}$. The empty word is denoted ε.

The *characteristic vector* of w is $\|w\| = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_k})$. For two vectors with the same number of integer components, we use the component-wise addition, subtraction, scalar multiplication, and ordering. Notice that, for two words u and v, $\|u\| = \|v\|$ means that u is a permutation of v and $\|u\| \le \|v\|$ means that u can be obtained from v by permuting and, possibly, deleting some of its letters.

A word w has *abelian period* p if we can write $w = u_0u_1 \ldots u_ru_{r+1}$, where $u_i \in A^*$, $r \ge 0$ such that

(i) $|u_1| = |u_2| = \cdots = |u_r| = p$ and
(ii) $\|u_0\| \le \|u_1\| = \|u_2\| = \cdots = \|u_r\| \ge \|u_{r+1}\|$.

For example, the word aabbbabbbaba has abelian period 3; it can be factorized as a.abb.bab.bba.ba .

Notice that one might be tempted to give the above definition with $u_0 = \varepsilon$. Such a definition would have the problem of not being preserved for factors. (The definition of weak repetitions in [8] has $u_0 = u_{k+1} = \varepsilon$.)

Remark further that any multiple of an abelian period is also an abelian period. Also, abelian period is a generalization of ordinary period. Precisely, w has (ordinary) period p if and only if w has abelian period p that can be arbitrarily "shifted," that is, for any possible length of u_0 between 0 and p, the properties (i)-(ii) still hold.

Fine and Wilf's theorem for abelian periods. We prove in this section our variant of Fine and Wilf's theorem for abelian periods. As we show in the last section, the result holds only for relatively prime abelian periods. We shall conjecture a weaker result for the general case.

Theorem 1. *If a word w has abelian periods p and q which are relatively prime and $|w| \ge 2pq - 1$, then w has period one.*

Proof. Assume $p < q$ and let $w = u_1u_2 \ldots u_{m+1}$ and $w = v_1v_2 \ldots v_{n+1}$ be the factorizations of w with respect to the periods p and q, respectively. From the overlaps in w between the us and the vs we can write each v_i, $1 \le i \le n$ as $v_i = x_iu_{b_i+1}u_{b_i+2} \ldots u_{b_{i+1}-1}y_i$, for some suffix x_i of u_{b_i} and prefix y_i of $u_{b_{i+1}}$; see Fig. 1. To be precise, we set b_i as the smallest index j for which $|u_0u_1 \ldots u_j| \ge |v_0v_1 \ldots v_{i-1}|$.

We have then $|x_i|+|y_i| \equiv q \pmod p$ and $|x_{i+1}|+|y_i| = p$ and hence $|x_{i+1}| \equiv |x_i|-q \pmod p$, for any $i \ge 1$. Inductively, we get $|x_{i+r-1}| \equiv |x_i| - (r-1)q \pmod p$, for $r \ge 1$, $i \ge 1$. For $i = 1$, we have $|x_r| \equiv |x_1| - (r-1)q \pmod p$. As p and q are relatively prime, there is r, $1 \le r \le p$, such that $|x_r| \equiv 0 \pmod p$ (take

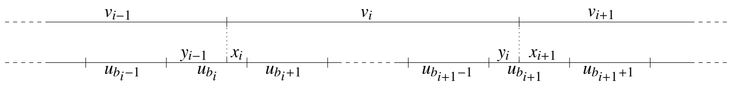


Figure 1: The two abelian periods

$r-1 = (|x_1|(q^{-1} \mod p)) \mod p)$ and, since $|x_r| < p$ by the definition of b_r, it must be that $x_r = \varepsilon$.

Therefore $v_0v_1 \dots v_{r-1} = u_0u_1 \dots u_{b_r}$ and $|v_0v_1 \dots v_{r-1}| = |v_0| + (r-1)q \le pq$. However, if $|v_0| = q$, then we can start the congruences one step earlier, that is $|x_r| \equiv |x_0| - (r-1)q \pmod p$, for $r \ge 1$, and obtain a smaller r.

Therefore, we need only $pq-1$ symbols at the beginning of w to have a perfect match between us and vs. Since p and q are co-primes, the next match will appear after exactly pq symbols. But $|w| \ge 2pq-1$, so the second match is guaranteed. More precisely, $v_rv_{r+1} \dots v_{r+p-1} = u_{b_r+1}u_{b_r+2} \dots u_{b_r+q}$. Now, the vectors $\alpha = ||v_2||$ and $\beta = ||u_2||$ must have the same non-zero components. Since the sum of the non-zero components of α is q, if there are more than two components greater than 0 in α then one of them, say α_i, is less than q. But $p\alpha_i = q\beta_i$, which implies that p/q is reducible, a contradiction. Consequently, α and β must have only one non-zero component which means that w contains only one letter. □

The general case. In general, Fine and Wilf's theorem cannot be extended to abelian periods which are not relatively prime. That is, if $\gcd(p,q) = d \ge 2$, then the two abelian periods p and q cannot impose the abelian period d no matter how long the word is. Here is an example. The infinite word $w = (\mathsf{bbaaababbbaa})^\omega$ has abelian periods 4 and 6 but not 2. (The notation x^ω means $xxx\dots$.)

A number of problems appear naturally. The first problem is to find out what is imposed by two non-relatively prime periods. In particular, is it true that gcd $= d$ implies the word has at most d letters? Also, is the bound in Theorem 1 optimal? Can the definition of the abelian period be modified so that Fine and Wilf's theorem holds in all cases? The case of more periods could be also of interest. Finally, one can also attempt similar generalizations for approximate periods, but the correct formulation of the problem becomes less clear.

References

[1] J. Berstel and L. Boasson, Partial words and a theorem of Fine and Wilf, *WORDS 1997 (Rouen), Theoret. Comput. Sci.* **218**(1) (1999) 135 – 141.

[2] F. Blanchet-Sadri, Periodicity on partial words, *Comput. Math. Appl.* **47**(1) (2004) 71 – 82.

[3] F. Blanchet-Sadri and R. Hegstrom, Partial words and a theorem of Fine and Wilf revisited, *Theoret. Comput. Sci.* **270**(1-2) (2002) 401 – 419.

[4] G. Castelli, F. Mignosi, and A. Restivo, Fine and Wilf's theorem for three periods and a generalization of Sturmian words, *Theoret. Comput. Sci.* **218** (1999) 83 – 94.

[5] S. Cautis, F. Mignosi, J. Shallit, M.-w. Wang, and S. Yazdani, Periodicity, morphisms, and matrices, *Theoret. Comput. Sci.* **295** (2003) 107 – 121.

[6] C. Choffrut and J. Karhumäki, Combinatorics on Words, in: G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages, Vol. I*, Springer-Verlag, Berlin, Heidelberg, 1997, 329 – 438.

[7] S. Constantinescu and L. Ilie, Fine and Wilf's theorem for any number of periods, in: T. Harju, J. Karhumaki, eds., *Proc. of the 4th WORDS, (Turku, 2003)*, TUCS General Publication **27** (2003) 65 – 74.

[8] L.J. Cummings and W.F. Smyth, Weak repetitions in strings, *J. Combin. Math. Combin. Comput.* **24** (1997) 33 – 48.

[9] P. Erdös, Some unsolved problems, *Hungarian Academy of Sciences Mat. Kutato Intezet Kozl.* **6** (1961) 221 – 254.

[10] N.J Fine and H.S. Wilf, Uniqueness theorem for periodic functions, *Proc. Amer. Math. Soc.* **16** (1965) 109 – 114.

[11] J. Justin, On a paper of Castelli, Mignosi, Restivo, *Theoret. Inform. Appl.* **34** (2000) 373 – 377.

[12] V. Keränen, Abelian squares are avoidable on 4 letters, *Automata, languages and programming (Vienna, 1992)*, Lecture Notes in Comput. Sci. **623**, Springer, Berlin, 1992, 41 – 52.

[13] M. Lothaire, *Combinatorics on Words*, Addison-Wesley, Reading, Mass., 1983 (reprinted, Cambridge Univ. Press, 1997).

[14] M. Lothaire, *Algebraic Combinatorics on Words*, Cambridge Univ. Press, Cambridge, 2002.

[15] F. Mignosi, A. Restivo, and P.V. Silva, On Fine and Wilf's theorem for bidimensional words, Selected papers in honor of Jean Berstel, *Theoret. Comput. Sci.* **292**(1) (2003) 245 – 262.

[16] J.-S. Sim, C. Iliopoulos,K. Park, and W.F. Smyth, Approximate periods of strings, *Theoret. Comput. Sci.* **262**(1-2) (2001) 557 – 568.

[17] A. Thue, Über unendliche Zeichenreihen, *Norske vid. Selsk. Skr. Mat. Nat. Kl.* **7** (1906), 1 – 22.

[18] A. Thue, Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen, *Norske vid. Selsk. Skr. Mat. Nat. Kl.* **1** (1912) 1 – 67.

[19] R. Tijdeman and L. Zamboni, Fine and Wilf words for any periods, *Indag. Math. (N.S.)* **14**(1) (2003) 135 – 147.

Recursion Theorems and Self-Replication Via Text Register Machine Programs

Lawrence S. Moss[*]

Abstract

Register machine programs provide explicit proofs of results such as the s^m_n-Theorem, Kleene's Second Recursion Theorem, and Smullyan's Double Recursion Theorem. Thus these programs provide a pedagogically useful approach. We develop without appeal coding, universal programs, or quotation. None of the results are new from the point of view of computability theory apart from the particular formulations themselves. We introduce the notion of a *text register machine*; this is a register machine whose registers contain words from some alphabet and whose instructions are again words from the same alphabet. We work with a particular instruction set whose language of programs we call `1#`. Tools for writing and evaluating `1#` programs have been made freely available: see `www.indiana.edu/~iulg/trm`.

It is generally recognized that the greatest advances in modern computers came through the notion that programs could be kept in the same memory with 'data,' and that programs could operate on other programs, or on themselves, as though they were data."

Marvin Minsky [5]

1 Introduction

What is the simplest setting in which one can formalize the notion that "programs could operate on other programs, or on themselves"? This paper contains a proposal for such a formalization in the form of a programming language `1#`.[1] Using

[*]Mathematics Department, Indiana University, Bloomington, IN 47405 USA. Email: `lsm@cs.indiana.edu`

[1]This paper leaves open the pronunciation of the name of the language. The symbol # has many names, including *check*, *octothorpe*, *pound sign*, *hash*, and *crosshatch*. It is not quite the sharp sign ♯. The names of languages like *A*# and *C*# use "sharp" anyways.

programs of 1# we obtain explicit programs corresponding to the fixed point theorems from the theory of computation. In addition to re-proving classic results, our goal is to suggest a useful way of teaching them.

To get the simplest formulation of any complicated idea is not always easy, and to do so for programs operating on programs one must make some choices. The goal of this paper is to present a setting in which one could explain the concept to a person who can read mathematical notation but who doesn't necessarily know about programming or computability. Thus the model of computation in this discussion should be as intuitively simple as possible. For this, I have chosen a certain flavor of register machines. Second, the notion of a program should also be as simple as possible: both the syntax and semantics should be explainable in less than fifteen minutes. This rules out high-level programming languages. As they are standardly studied, register machines work on numbers, but their programs are not numbers. So we work with a variant notion, *word register machines*. Such machines process strings over the tiny alphabet $\mathcal{A} = \{1, \#\}$. The key additional feature of our machine is that its instructions and programs are words over this same set $\mathcal{A}$. We call such machines *text register machines*.

Word register machines are a Turing-complete computational formalism. Text register machines also illustrate versions of the main foundational theorems of recursion theory explicitly. So when a result says "there is a program such that ..." then it is often possible to easily exhibit such a program. The results I have in mind are the s^m_n-Theorem and the Second Recursion Theorem. Usually the s^m_n-Theorem is treated by appealing to the Church-Turing thesis (that is, saying that the construction is "obvious but tedious") or else done via the coding of sequences by numbers. Our development is more direct. It also leads to explicit *self-reproducing programs*.

We turn to the main development itself in Section 3. Before that, we have some discussion of how our proposal fits into the history of the subject.

2 Historical and conceptual points

The register machine formalism was introduced in Shepherdson and Sturgis' 1963 paper [6]. Their goal was to provide a formalism for which one could verify that all partial recursive functions are computable by some "finite, discrete, deterministic device supplied with unlimited storage." Their notion of a register machine comes in several variants. Broadly speaking, these variants include machines whose registers contain natural numbers, and also with machines whose registers contain words $w \in \mathcal{A}^*$ over some set $\mathcal{A} = \{a_1, \ldots, a_s\}$ of alphabet symbols. We are concerned in this paper with the latter variant. It has for the most part been forgotten in the literature. For example, Fitting [2] writes, "Register machines come from [Shepherdson and Sturgis 1963]. As originally presented, they manipulated numbers not words; the version presented here is a straightforward modification." And to be sure, I had not known of word register machines before I wrote up this paper. Getting back to the idea itself, it might be worthwhile to recall (in a somewhat updated terminology and notation) the formulation in [6].

In Sections 5 and 6 of [6] we find a formulation of *partial recursive functions on* $\mathcal{A}^*$ and then a definition of *unlimited register machines* over $\mathcal{A}$. These are register machines whose instructions are as follows:

$P^{(i)}(n)$: place a_i on (the right-hand) end of $\langle n \rangle$

$D(n)$: delete the (left-most) letter of $\langle n \rangle$, provided that $\langle n \rangle \neq \epsilon$.

$J^{(i)}(n)[E1]$: jump to line $E1$ if $\langle n \rangle$ begins with a_i, otherwise go to the next instruction.

In these, $\langle n \rangle$ denotes the content of register n, and ϵ the empty word. A machine whose instructions are of the above types is called a URM($\mathcal{A}$).

The main result of Appendix B is that all partial recursive functions over $\mathcal{A}^*$ may be computed on some URM($\mathcal{A}$). A variant of the instruction set above comes in Appendix C. It replaces the second and third types of instructions by one "scan and delete" scheme defined as follows:

$Scd(n)[E_1 \ldots, E_s]$: scan the first letter of $\langle n \rangle$; if $\langle n \rangle = \epsilon$, go to the next instruction; if the first letter of $\langle n \rangle$ is a_i, then delete this and go to line E_i.

Then the main result of Appendix C is that one can simulate the delete and jump instructions using the scan and delete operation.

The paper contains numerous other results. However, it does not formulate direct results like the s^m_n-Theorem for URM($\mathcal{A}$) computations. We would like to do this directly. For this, it is essential that $\mathcal{A}$ include whatever symbols are needed to formulate the syntax of the overall language. (In the setting of [6], this is problematic for an interesting reason. If the symbols $P^{(i)}$ are taken to be atomic symbols $P^{(i)}$ – and this seems to be the prima facie interpretation – then the alphabet $\mathcal{A}$ would have to include those symbols. But this is absurd if $\mathcal{A}$ is to be finite. So we must reformulate the language to get around this.)

Register machines are used in many textbooks due to their intuitive appeal. Needless to say, in writing this paper I looked at many sources to make sure that the development was new. Occasionally register machines are called *counter machines*. Minsky's textbook [5] on automata theory and computability presents a machine model that amounts to arithmetic register machines. He calls them *program machines*; it is not clear from the book why he did not mention [6] in the text even though it appears in the references. He notes that the program is " 'built into' the programming unit of the machine.” Hence [program machines] “are *not*, then, ‘stored-program’ computers.” But he then makes the quote reproduced above the introduction of the present paper. Perhaps Minsky did not even present the word register machines because register machines are not a direct model of a stored-program computer. But our point is that register machines operating on words do allow one to come closer to the notions that Minsky cites than machines operating on numbers. His book also does discuss Turing machines whose symbol set goes beyond one or two symbols to be an arbitrary finite set, including the set of symbols used in a “representation” of a Turing machine itself. Problem 7.4–3 of [5] asks a reader to construct a self-reproducing Turing machine, and the solution of course would have to use an expanded symbol set.

There are two papers that present material that seems close to ours. One is Corrado Böhm’s paper [1]. He proposes a language P'' with a small instruction set, and proves that it is Turing-complete. P'' is intended to be run on Turing machines. Here are some differences with our work: we feel that our instruction sets would be easier for a complete novice to use than P''. (For example, moving register contents is a simple loop here.) Certainly the programs for the self-replicating program that we end up with is considerably smaller than what one finds for descendants of P'' such as BF. Finally, our languages are regular sets of expressions, whereas languages like P'' are context-free but not regular. (This is a very minor point.) We also know that Neil Jones in [4] has proposed simple languages and studied the s^m_n- and Recursion Theorems. The difference here is that Jones’ languages are not based on as simple a machine model as register machines. There are good reasons why Jones works with the languages that he formulated, of course.

There might be classroom situations where one might want a presentation like the one we outline here. I think back to my own first exposure to Computability Theory, an taking an inspiring course given by Herbert Enderton at UCLA around 1977. The course used register machines as its primary model, and in the first few weeks we had to run programs written on punched cards. Later on, the course turned to μ-recursion, and via the usual coding machinery it presented the s^m_n-Theorem (but not the Recursion Theorem). The development here would allow one to efficiently present all the main results of interest without any of the coding; for some courses this would be a good idea. Instead of punched cards one now has graphical interfaces, and for this the technical overhead in learning the language would be small indeed. The material would work well in any course that wants to discuss self-replicating computer programs, a topic that comes up in various settings where reproduction is studied, including compilers, artificial life, and security. So someone teaching those topics could introduce them using 1# and the freely available tools for it.

3 The language 1#

A register is a time-varying word indexed by a positive integer. We work with a machine whose registers are R1, R2, R3, ... Each program uses a fixed finite number of registers. We may either take the ideal machine which we now describe to have infinitely many registers, or else to have a number which includes all registers in whatever program we are running on it.

Syntax The basic alphabet of symbols is $\mathcal{A} = \{1,\#\}$.

There are five types of instructions, and the full syntax is listed in Figure 1.

The set of programs is just the set of all nonempty concatenations of sequences of instructions. The set of instructions is a regular set, and hence so is the set of programs. The programs are uniquely (and efficiently) parsed into sequences of instructions.

We sometimes employ abbreviations in writing programs, and in particular we use | for the concatenation operation on programs. We also add explanations in English. None of this is part of the official language.

We experimented with variations on the syntax in order to get an instruction set which minimized the lengths of the programs of interest. Nothing beat Figure 1. Having extra characters allows for binary numbers, but this does not quite compensate for the extra branches in the case statements.

Semantics The easiest way to present the semantics is to run an evaluator in connection with our tutorial on this topic. For those reading this on its own, here are the details.

The registers store words $w \in \mathcal{A}^*$. Running a program, say p, means executing a sequence of steps, and at each step one or another of the instructions which comprise p is *active*. It is convenient to number those instructions. We begin with instruction 1 of p. When p starts, some registers might contain words; these are the *inputs*. Actually, we do not distinguish between a register being empty and its containing the empty string. The various instructions in our set involve writing to the end of a register, popping an element from the front of a register and then branching according to what was found, and outright transfer (=goto) statements. Here is more detail on all of these.

All writing is to be done at the end (the right side) of words. So if R6 contains the string 11## and we execute the instruction 111111#, then R6 will then have 11##1. After executing a write instruction, the next instruction of p (if there is one) is the active one.

Executing a case statement $1^n\#^5$ removes the leftmost symbol of the word in register n, so that item is no longer there. After this, there are three *continua-*

instr	meaning	instr	meaning
1^n#	add 1 at the end of Rn	1^n####	go backward n steps
1^n##	add # at the end of Rn	1^n#####	cases on Rn
1^n###	go forward n steps		

Figure 1: The instruction set of 1#. Here $n \geq 1$, and 1^n is n consecutive 1s.

tion branches. In order, those branches are first for the case when register n is empty, then when the popped element is 1, and finally for #. Suppose R3 contains the string 1##1 and instruction 17 of our program p contains the instruction 111#####. In executing this, we drop the initial 1, the tail ##1 then remains in R3, and finally we proceed to instruction $17 + 2 = 19$ of p.

The transfer instructions are all *relative*; i.e., they specify a forward or backward transfer of some positive number of instructions.

Consider the execution of a program p. If at some point, instruction k is active and it asks to transfer forward l steps, and if p has $k + l - 1$ instructions, then after executing instruction k, we say that *p halts*. There are similar ways for p to halt following the add instructions and even case statements. Informally, and a bit incorrectly, we say that p halts if the active line is "one below the last instruction of p."

1#-computable partial functions Let $n \geq 0$, and let $p \in \mathcal{A}^*$. We define the partial function $\varphi_p^{(n)} : (\mathcal{A}^*)^n \rightharpoonup \mathcal{A}^*$ by

$$\varphi_p^{(n)}(x_1, \ldots, x_n) \quad = \quad y$$

if p is a program, and if p is run with x_1 in R1, x_2 in R2, …, x_n in Rn, and all other registers empty, then eventually the machine comes to a halt with y in R1 and all other registers empty. These partial functions $\varphi_p^{(n)}$ are called *1#-computable*. (We allow $n = 0$, and in this case we would write $\varphi_p(\ )$. And in all cases we usually drop the superscript from the notation when it is clear.)

This notion of 1#-computability is not the only one worth studying. For many purposes, one would want *1#-computability using the first k registers*. Equally well, one often wants definitions that keep the original input intact; the notion studied here loses the input.

Notation like $\varphi_p(x_1, \ldots, x_n) = \varphi_q(y_1, \ldots, y_m)$ has the usual meaning: either both sides are undefined, or both are defined and equal.

The empty string ϵ is not a program (by definition), and so $\varphi_\epsilon^{(n)}$ is the empty function for all n. Incidentally, our stipulation that ϵ not be a program might be reconsidered. One might then reconsider the status of $\varphi_\epsilon^{(n)}$. This would result in minor changes to our results.

It is clear that restricting our notion from sequences from $\mathcal{A} = \{1, \#\}$ to sequences of 1's alone, the 1#–partial computable functions are exactly the partial computable functions in the classical sense. There is also a formulation of this result which uses $\mathcal{A}$ to represent numbers in binary.

4 Programs

4.1 Programs to move register contents

Here is a program $\mathsf{move}_{2,1}$ which writes the contents of R2 onto the end of R1, emptying R2 in the process:

11#####	cases on R2	1111####	go backward 4
111111###	go forward 6	1#	1: add 1 to R1
111###	1 branch: go forward 3	111111####	go backward 6
1##	# branch: add # to R1		

Note that in our displayed examples we often write the instructions going down the two columns. Similarly, we can build $\mathsf{move}_{m,n}$ for all distinct numbers m and n. The official program $\mathsf{move}_{m,n}$ is

$$1^m\#\#\#\#\#111111\#\#\#111\#\#\#1^n\#\#1111\#\#\#\#1^n\#111111\#\#\#\#.$$

4.2 Comparison and reversal

It is a good exercise to write some programs dealing with string manipulations. One would be to write a program compare with the following property. When run with x in R1 and y in R2, compare halts with 1 in R1 (and nothing in any other register) if $x = y$, and with R1 (and all other registers) empty if $x \neq y$.

A better exercise is to write a program that reverses words.

4.3 A program to write a program to write a word

Figure 2 contains a program write with the following property: when write is started with a word x in R1 and R2 empty, the output is a word $y = \varphi_{\mathsf{write}}(x)$ in R1 and all other registers empty. Moreover, y is a concatenation of instructions 1# and 1##. And running y writes the original x after whatever happens to be in R1, provided $x \neq \epsilon$. For such x we indeed have

$$\varphi_{\varphi_{\mathsf{write}}(x)}(\,) \quad = \quad x.$$

The official program is shown at the bottom of Figure 2; the parse is on top. The horizontal line indicates that $\mathsf{move}_{2,1}$ is concatenated at that point.

1#####	cases on R1	111111####	go backward 6
111111111###	empty branch	11#	1: add 1 to R2
11111###	to 1 branch	11##	add # to R2
11#	# branch	111111111####	go backward 9
11##	add # to R2	$\mathsf{move}_{2,1}$	
11##	add # to R2		

```
1#####111111111###11111###11#11##11##111111####11#11##
111111111####11#####111111###111###1##1111####1#111111####
```

Figure 2: The program write.

4.4 s^1_1

We construct a program s^1_1 which, when when started with a program p in R1 and a word q in R2, and R3 and R4 empty, yields a program $\varphi_{\mathsf{s}^1_1}(p,q)$ which, when started with r in R1, yields the same word as would be obtained when p is started with q in R1 and r in R2. In particular,

$$\varphi_{\varphi_{\mathsf{s}^1_1}(p,q)}(r) \quad = \quad \varphi_p(q,r). \tag{1}$$

We take s^1_1 to be

$$\mathsf{move}_{1,3} \,|\, \mathsf{move}_{2,1} \,|\, \mathsf{write} \,|\, \mathsf{move}_{1,2} \,|\, \varphi_{\mathsf{write}}(\mathsf{move}_{1,2}) \,|\, \mathsf{move}_{2,1} \,|\, \mathsf{move}_{3,1}$$

We use | as a symbol for concatenation of programs. Note also that the third of the seven segments is write, the program that we saw in Section 4.3 just above; the fifth is the result of applying that program to $\mathsf{move}_{1,2}$. Then the following equations show the desired result (1):

$$\begin{array}{lcl} \varphi_{\mathsf{s}^1_1}(p,q) & = & \mathsf{move}_{1,2} \,|\, \varphi_{\mathsf{write}}(q) \,|\, p \\ \varphi_{\mathsf{move}_{1,2} \,|\, \varphi_{\mathsf{write}}(q) \,|\, p}(r) & = & \varphi_p(q,r) \end{array}$$

There are also versions of s^m_n for all m and n.

4.5 The programs diag and self

This section illustrates self-replicating programs in 1#. We begin with the following program which we'll call diag for the rest of this paper. When diag is run with x in R1, the result is $\varphi_{\mathsf{write}}(x) \,|\, x$ in R1. Hence for $x \neq \epsilon$,

$$\varphi_{\varphi_{\mathsf{diag}}(x)}(\,) \quad = \quad \varphi_{\varphi_{\mathsf{write}}(x) \,|\, x}(\,) \quad = \quad \varphi_x(x). \tag{2}$$

Here is the informal description of diag:

Move x from R1 into R2, and at the same time put $\varphi_{\mathsf{write}}(x)$ in R3. Move R3 back to the now-empty R1. Finally, move R2 onto the end of R1.

The way to formalize the "at the same time" is to write one combined loop:

`1#####`	cases on R1	`1111111####`	go back 7
`11111111111###`	empty: go 11	`11#`	add 1 to R2
`111111###`	1 branch: go 6	`111#`	add 1 to R3
`11##`	#: add # to R2	`111##`	add # to R3
`111#`	add 1 to R3	`11111111111####`	go back 11
`111##`	add # to R3	$\mathsf{move}_{3,1}$	
`111##`	add # to R3	$\mathsf{move}_{2,1}$	

One way to get a self-writing program self is to apply this program diag to diag itself. When we run diag on itself, we get $\varphi_{\mathsf{write(diag)}}|\mathsf{diag}$ in R1. So when we run self on nothing, we first write diag into R1; and second we run diag. This gives us self, as desired. For a more formal proof, we use (2) with $x = \mathsf{diag}$:

$$\varphi_{\mathsf{self}}(\,) \quad = \quad \varphi_{\varphi_{\mathsf{diag}}(\mathsf{diag})}(\,) \quad = \quad \varphi_{\mathsf{diag}}(\mathsf{diag}) \quad = \quad \mathsf{self}.$$

One can find the full programs on `www.indiana.edu/~iulg/trm`.

Programs like self are often called *quines* following Douglas Hofstadter in [3]; there are several web pages devoted to collections of them, for example. A standard example is to take λ-terms as the programs, β-conversion as the notion of execution, and then to consider $(\lambda x.xx)(\lambda x.xx)$.

It might be useful to expose the device behind our self-replicating program self by rendering it into English. We are interested in "programs" (sequences of instructions) of a simple form, including instructions to print various characters, and instructions which accept one or more sequences of words as arguments, and so on. We'll allow quotation, and we won't attempt to formulate a minimal language, or a formal semantics. We're aware of the semantic problems that are being pushed under the rug, but the point is only to hint at a rendering of diag and self. Perhaps the most direct example of a self-replicating program would be `write me`, but this not immediately translated to `1#`. Instead, we want a version of diag, and we take

```
write the instructions to write what you see before it
```
(3)

Here "what you see" and "it" refer to the input. So applying this informal rendering of diag to `write me` would give

```
print "w" print "r" print "i" print "t"
print "e" print " " print "m" print "e" write me
```

Applying this version of diag to itself gives a long program which would look like

```
print "w" print "r" print "i" print "t" print "e"      ···
print "r" print "e" print " " print "i" print "t"        (4)
write the instructions to write what you see before it
```

Executing (4) prints instructions to print (3) followed by (3) itself. This is (4).

5 Exercises

I have used most of the exercises below as classroom exercises before turning to Kleene's Recursion Theorem. As one might expect, some of the problems can be solved by appealing to the Recursion Theorem. However, the direct solutions are usually shorter.

1. Write a program which when started on all empty registers writes itself to R1 and # to R2.

2. Write an infinite sequence of pairwise different programs

   $$p_1, p_2, \ldots, p_n, \ldots,$$

   such that for all n, running p_n with all registers empty gives p_{n+1} in R1.

3. Write a self-replicating program that begins with the program to transfer ahead one instruction, 1###.

4. Which programs p have the property that there is a self-replicating program which begins with p?

5. Write a program s which when run with R2, R3, . . ., initially containing the empty string, writes s itself into R1 after whatever happens to be there to begin with.

6. Write a program p with the property that when run on a string q in R1, p runs and halts with 1 in R1 if $q = p$, and runs and halts with R1 empty if $q \neq p$. (So p "knows itself".)

7. Write a program p with the property that when run on a program q in R1, the result is the same as would be the case when q is run with p in R1. (So *program* and *data* have changed roles!) [You will want a circular interpreter for this.]

8. Write two "twin" programs s_1 and s_2 with the properties that (a) $s_1 \neq s_2$; (b) running s_1 with all registers empty gives s_2 in R1; (c) running s_2 with all registers empty gives s_1 in R1.

9. Work out a version of the Busy Beaver problem in this setting.

6 Kleene's Second Recursion Theorem

Here is our formulation of this fundamental result: Let $p \neq \epsilon$, and consider $\varphi_p^{(2)}$. Then there is a program q^* so that for all r,

$$\varphi_{q^*}(r) \quad = \quad \varphi_p(q^*, r).$$

For the proof, let $\hat{q}$ be as follows (later we set $q^* = \varphi_{\hat{q}}(\hat{q})$):

$$\mathsf{diag} \,|\, \mathsf{move}_{1,2} \,|\, \varphi_{\mathsf{write}}(\mathsf{move}_{1,4}) \,|\, \mathsf{move}_{2,1} \,|\, \varphi_{\mathsf{write}}(\mathsf{move}_{4,2} \,|\, p).$$

This program $\hat{q}$ uses only the first three registers. We have that for all x,

$$\varphi_{\hat{q}}(x) \quad = \quad \mathsf{move}_{1,4} \,|\, \varphi_{\mathsf{diag}}(x) \,|\, \mathsf{move}_{4,2} \,|\, p.$$

In particular, $\varphi_{\hat{q}}(\hat{q}) = \mathsf{move}_{1,4} \,|\, \varphi_{\mathsf{diag}}(\hat{q}) \,|\, \mathsf{move}_{4,2} \,|\, p$. Now for all r

$$\varphi_{\varphi_{\hat{q}}(\hat{q})}(r) \quad = \quad \varphi_{\mathsf{move}_{1,4} \,|\, \varphi_{\mathsf{diag}}(\hat{q}) \,|\, \mathsf{move}_{4,2} \,|\, p}(r) \quad = \quad \varphi_p(\varphi_{\hat{q}}(\hat{q}), r).$$

(This last point is worth checking in detail; it uses the fact that $\hat{q}$ only uses the first three registers.) Let $q^* = \varphi_{\hat{q}}(\hat{q})$. So we have $\varphi_{q^*}(r) = \varphi_p(q^*, r)$, as desired.

There are also versions of this result for $\varphi_p^{(k)}$ with $k \geq 3$, and the details are similar.

7 Smullyan's Double Recursion Theorem

Our final result is Smullyan's Double Recursion Theorem, a result used by Smullyan in work on recursion theory connected to the Gödel Incompleteness Theorems. Let p and q be programs. Consider the functions of three arguments $\varphi_p^{(3)}(a, b, x)$ and $\varphi_q^{(3)}(a, b, x)$. There are programs a^* and b^* so that for all x, $\varphi_{a^*}(x) = \varphi_p(a^*, b^*, x)$, and $\varphi_{b^*}(x) = \varphi_q(a^*, b^*, x)$.

We lack the space to exhibit a^* and b^* explicitly in terms of p and q (but Exercise 8 in Section 5 above could be a good first step). One example application of the Double Recursion Theorem would be to find p and q so that $\varphi_p(p) = q$, $\varphi_p(q) = p$, $\varphi_q(p) = q$, and $\varphi_q(q) = p|q$.

8 Using `1#`

Several students in my class enthusiastically implemented `1#`. Some of their work has been polished, and it is available at `www.indiana.edu/~iulg/trm`along with other material. The most widely usable is a Java program that comes with a pleasant interface that allows one to watch registers change as programs run. Other interpreters come with tools which make it easier to write programs; so in effect one can write in a slightly-higher-level language that is translated back to a bona fide `1#` program. There also is a universal program (a circular interpreter) for `1#`, that is a `1#` program u such that for all p and q, $\varphi_u(p, q) = \varphi_p(q)$.

Acknowledgements

My thanks to Will Byrd and Jiho Kim for numerous comments, corrections, references, and suggestions.

References

[1] Böhm, Corrado."On a family of Turing machines and the related programming language", ICC Bull. 3, 187-194, July 1964.

[2] Fitting, Melvin. *Computability Theory, Semantics, and Logic Programming*. Oxford Logic Guides, 13. The Clarendon Press, Oxford University Press, New York, 1987.

[3] Hofstadter, Douglas R. *Gödel, Escher, and Bach: an Eternal Golden Braid*. Basic Books, Inc. New York, New York.

[4] Jones, Neil. Computer implementation and applications of Kleene's *s-m-n* and recursion theorems. in Y. N. Moschovakis (ed) *Logic From Computer Science*, 243–263, Math. Sci. Res. Inst. Publ., 21, 1992.

[5] Minsky, Marvin L. *Computation: Finite and Infinite Machines*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967.

[6] Shepherdson, J. C. and Sturgis, H. E. Computability of recursive functions. *J. Assoc. Comput. Mach.* 10 217–255, 1963.

The Puzzle Corner

BY

Laurent Rosaz

LRI, Orsay CNRS-Université de Paris Sud
Bât 490, 91405 Orsay France
Laurent.Rosaz@lri.fr

Readers are invited to send comments, and to send exercises, even if they don't know the answer. Write to Laurent.Rosaz@lri.fr.

76 The 4 buoys

A geophysicist tells me he has dropped 4 buoys, floating on the sea, which are equidistant of each other. Should I call him a liar ?

77 Going far ?

Your are on the road. At km 0 is the unique gas station. You have N cars (and drivers) and you can fill the N tanks at the gas station. It is forbidden to store gas elsewhere than in the tanks. It is allowed at any time to decant gas from one car tank to another. A full tank allows a car to drive for a constant distance K. You have car number one (and that makes you the chief) and you are ready to abandon other cars on the way. How far can you go ?

Solutions to Previous Puzzles

74 Another fake coin problem: independent tests

There are N coins of which one is counterfeit. In each test, you measure a set of coins and the answer reveals if the set contains the counterfeit coin or not (so, the classical group testing setting). Obviously, we can find the counterfeit coin in $\lceil log_2 n \rceil$ tests by a straightforward binary division algorithm. It is also obvious that this number of test is optimal. However, in this procedure each test depends directly on the answers to the previous tests, and we want all tests to be independent of each other. In other words, we want to be able to make all tests in parallel. Is it possible to identify the counterfeit coin in $\lceil log_2 n \rceil$ tests by such an algorithm ? (if yes, give the algorithm; if not, give the lower bound) [Thanks to G. Kucherov for the problem]

Solution Number the coins from 1 to N. In the i^{th} test, put coin k iff the i^{th} bit in the writing of k in base 2 is a 1. It is easy to deduce the faulty coin from the $\lceil log_2 n \rceil$ results.

75 Number of loops on a grid

You are on an infinite grid. You are interested in paths from point $(0,0)$ to $(0,0)$ made of elementary North, South, East or West steps. For example,

$$NEEWESWNNWWSSSWNEE$$

is such a path of length 18. Let X_n be the number of such paths of length $2n$.

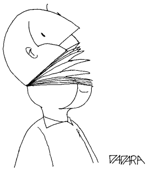
Compute X_n. After a little calculus, you should be able to get the answer in a very simple form, namely $(C_X^Y)^\delta$ (You should need $\sum_0^n (C_n^k)^2 = C_{2n}^n$). Could you explain the final result by a more direct proof ?

Solution First solution: choose the number k of west moves, then choose the k west moves, then the k east Moves, then the $n-k$ north moves. This leads to the formula

$$\begin{aligned}\sum_0^n C_{2n}^k * C_{2n-k}^k * C_{2n-2k}^{n-k} &= \sum_0^n \frac{(2n)!}{k!(2n-k)!} * \frac{(2n-k)!}{k!(2n-2k)!} * \frac{(2n-2k)!}{(n-k)!^2} \\ &= \sum_0^n \frac{(2n)!}{k!^2(n-k)!^2} \\ &= \frac{(2n)!}{n!^2} \sum_0^n \frac{n!^2}{k!^2(n-k)!^2} \\ &= C_{2n}^n \sum_0^n (C_n^k)^2 = (C_{2n}^n)^2\end{aligned}$$

Second solution: To every loop l of length n associate $NE(l)$, the set of steps which are north or east, and $NW(l)$, the set of steps which are north or west. The function $l \to (NE(l), NO(l))$ is a one-to-one function from the loops to X^2 where X is the set of n steps among $2n$. Hence the number of paths is $card^2(X) = (C_{2n}^n)^2$.

Reports from Conferences



Report on BCTCS 2006

The 22nd British Colloquium for Theoretical Computer Science

4–7 April 2006, Swansea, Wales

Faron Moller

The British Colloquium for Theoretical Computer Science (BCTCS) is an annual forum for researchers in theoretical computer science to meet, present research findings, and discuss developments in the field. It also provides an environment for PhD students to gain experience in presenting their work in a wider context, and benefit from contact with established researchers.

BCTCS 2006 was held at Swansea University during 4–7 April 2006. The event attracted 122 participants, and featured an interesting and wide-ranging programme of 6 invited talks and 62 contributed talks; roughly half of the participants and speakers were PhD students. Abstracts for all of the talks are provided below; further details, including on-line slides from the talks, are available from the BCTCS website at `http://www.bctcs.ac.uk/`. The financial support of the Engineering and Physical Sciences Research Council (EPSRC), the London Mathematical Society (LMS), the British Computer Society (BCS), and the Welsh Development Agency (WDA) is gratefully acknowledged.

A highlight of the meeting this year was a lengthy discussion, led by Samson Abramsky and chaired by Faron Moller, on the formation of a Learned Society for Computer Science in the UK. Such a Society was first proposed for Theoretical Computer Science 18 months earlier in a widely-circulated letter signed by Samson Abramsky, Faron Moller and David Pym which received overwhelming support; this led directly to wider interest in the UK in creating a Learned Society which would envelop the whole of Computer Science. This mammoth endeavour is being developed by a Working Party involving representatives from all relevant national organisations, and the meeting demonstrated near-unanimous support for its efforts (with only a small handful disappointed that the idea of a UK Society just for TCS would now be realised only as a sub-group of a wider organisation).

Another novel feature of BCTCS 2006 was the form of its opening Invited Lecture, which came in the guise of Peter Mosses' Public Inaugural Lecture marking his recent appointment to a professorship at Swansea University.

BCTCS 2007 will be hosted jointly by Oxford and Oxford Brookes Universities in Oxford University's St. Anne's College from 2–5 April 2007. Researchers and PhD students wishing to contribute talks concerning any aspect of theoretical computer science are warmly welcomed to do so. Further details are available from the BCTCS website at `http://www.bctcs.ac.uk`.

Invited Talks at BCTCS 2006

Hajo Broersma, Durham University
Toughness in graphs: structural and algorithmic aspects
(LMS Keynote Lecture in Discrete Mathematics)

The ***toughness*** of a graph G is a vulnerability or reliability measure related to its connectivity, but it involves the number of components $\omega(G-S)$ that result after deleting a subset S of the vertex set of G. Formally, a non-complete graph $G = (V, E)$ is called ***t-tough*** if $t{\cdot}\omega(G-S) \leq |S|$ for every set $S \subseteq V$ with $\omega(G-S) > 1$. The concept of toughness was introduced by Chvátal in the seventies. Since then a lot of research has been done, mainly relating toughness conditions to the existence of cycle structures. Historically, most of the research was based on a number of conjectures by Chvátal. More recently, research has also focused on computational complexity issues. We will survey progress and open problems in both directions.

Stephen Cook, University of Toronto
A tutorial on proof complexity

NP = co-NP iff there is a propositional proof system in which every tautology has a polynomial size proof. Proving NP $\neq$ co-NP would show NP $\neq$ P, and seems out of reach at present. But it motivates trying to prove lower bounds on proof length for specific proof systems. We summarize some results here, and also explain the interesting connection between proof systems and complexity classes.

Tony Hoare, Microsoft Cambridge
Unifying theories of concurrency

The goal of unifying theories is one that inspires much good basic research in all mature branches of scientific endeavour. To show that two or more theories are just special cases of some yet more general theory is a strong support of the credibility of all the theories involved. Unification is a scientific ideal that can justifiably be pursued by theorists for its own sake.

In Computer Science, a good theory of programming can also deliver important practical benefits. It provides a sound conceptual basis for the construction of programming tools, which make the results of the theory available to the practising software engineer. Many such tools, incorporating specialised theories of concurrency like Esterel, are now finding application in the design of hardware and of software systems embedded in aeroplanes and cars.

In order to extend the utility of these tools, it eventually becomes necessary to use them in combination with other specialised tools. To ensure the quality and soundness of such a combined tool, it is essential that it should be based on a unification of their theoretical foundations. So without in any way detracting from the value of proliferation of theories, I would like to suggest that unifying theories is a suitable Grand Challenge to inspire collaborative research in Theoretical Computer Science. I will give a brief example of my own current attempt to unify two familiar theories of concurrency, CCS and CSP. It finds an interesting application of the concept of a Scott retraction.

Mark Jerrum, University of Edinburgh

A tutorial on efficient sampling

Sampling – it might be of points from some spatial distribution or of specified combinatorial structures – is an interesting algorithmic pursuit. Moreover, many sampling problems are externally motivated, for example, by models in statistical physics. There have been a number of notable successes in the area, but many open questions remain. It seems an opportune moment to conduct a SWOT analysis.

Peter Mosses, Swansea University

The meaning of it all: Programming language semantics, from Scott and Strachey to semantics/online

(BCS-FACS Keynote Lecture in Formal Methods)

Since the middle of the last century, hundreds of programming languages have been designed and implemented – and new ones are continually emerging. The texts that make up the programs of a language – the syntax of the programming language – can usually be described quite precisely and efficiently using formal grammars first developed in linguistics. However, the formal description of what the programs do – their semantics – is much more challenging. Like syntax in the 1950s, precise semantics is commonly regarded as impractical and too costly. Research in semantics allows us to reason about software and has provided valuable insight into how (not) to design programming languages, but few semantic descriptions of full languages have been published, and hardly any of these are available online.

One of the major approaches to formal semantics is denotational semantics, developed by Scott and Strachey in the late 1960s. Recent research has shown how to combine some aspects of denotational semantics with other approaches. A radical change to the way semantic descriptions are organised has also dramatically improved their practicality, and should allow efficient online access to a repository of semantic descriptions, as well as contributing to the solution of a long-standing open problem: programming the automatic generation of compilers and interpreters from language descriptions.

Moshe Vardi, Rice University, Houston

Alternation as an algorithmic construct

Alternation was introduced by Chandra, Kozen, and Stockmeyer (JACM, 1981) as a generalization of nondeterminism. The typical use of alternation is in complexity-theoretic settings. The focus of this talk is on presenting alternation as a powerful algorithmic construct. The driving examples are various automated-verification tasks, where alternation yields elegant and optimal algorithms.

Contributed Talks at BCTCS 2006

Thorsten Altenkirch, University of Nottingham

Stop thinking about bottoms when writing programs!

Reasoning about functional programs is often obscured by dealing with the possibility of non-terminating programs, i.e. programs denoting bottom. I argue that most programs can be perfectly well understood in a total setting and this provides a more appropriate framework for reasoning about programs formally and informally. There are some programs where partiality cannot be avoided, eg interpreters, however I will show that this impurity can be dealt with by a monadic interface, the partiality monad. The latter is based on joint work with with Venanzio Capretta and Tarmo Uustalu.

Ioannis Baltopoulos, University of Cambridge

Model checking business processes

In this talk we present ongoing work in the area of model checking of business processes expressed in the pi-calculus. The Business Process Execution Language (BPEL) is an OASIS standard aimed at providing a language for modelling the behaviour of executable business processes and business protocols (abstract processes) collectively known as Business Processes.

Using as a starting point previous work on the semantics of the BPEL language done in Petri nets we initially present a pi-calculus semantics for the language, by means of a set of transformation rules from XML descriptions to pi-calculus ones. The application of the semantics to business processes results in formal process descriptions that lend themselves to property checking through the modal mu-calculus.

Future extensions of this work will involve capturing temporal information in the process descriptions that would enable the calculation of performance metrics and the verification of temporal properties.

Joachim Baran, University of Manchester

Linear temporal logics and grammars

Linear temporal logics are widely associated with automata, i.e. a formula can be (more or less) easily translated into an automaton that accepts the formula's models as its language and vice versa. The most prominent example is probably the linear-time mu-calculus, whose expressiveness coincides with ω-Buchi automata. However, when considering logics that go beyond ω-regular expressiveness, the relationship to automata is either not clear or not straightforward anymore.

The linear-time fixed-point logic with chop (LFLC) is an extension of the linear-time mu-calculus. Its expressiveness is beyond context-free properties and it has been shown that the logic's models coincide with the languages of alternating context-free grammars over finite and infinite words. We give here a short introduction to the field of temporal logics and grammars and our latest research results concerning the representation of the empty word and about the alternation hierarchy in LFLC.

Nick Cameron, Imperial College London

A state abstraction for Java like languages

An objects' state, intended as some abstraction over the value of fields, is always in the mind of (imperative object-oriented languages) programmers. When concurrency comes in, the need for a state abstraction becomes even more prevalent: as the state of an object changes so, usually, does the synchronization behaviour.

We introduce a language feature for expressing the notion of state in Java-like languages. The proposed feature takes the form of a new kind of class, that we call a state class. We first introduce and motivate the new construct through examples written in a dialect of Java called StateJ. Then, we provide a formal account of our proposal by presenting syntax, typing, reduction, and type soundness for the FSJ calculus, a minimal core calculus (in the spirit of Featherweight Java) for modelling the state construct.

Luis Cereceda, London School of Economics

Recolouring graph colourings

Suppose we are given a graph G together with two proper vertex k-colourings of G, α and β. How easily can we decide whether it is possible to transform α into β by recolouring vertices of G one at a time, making sure that at each stage we have a proper k-colouring of G? We consider some algorithmic aspects of this and related questions. Special attention will be given to the case k=3.

This is joint work with Jan van den Heuvel and Matthew Johnson.

David Cunningham, Imperial College London

Implementing atomicity with locks

In a multi-threaded shared-memory system, many single-threaded algorithms fail because of atomicity violations that break the programmer's assumptions. In practice, this is prevented by explicitly coding synchronisation into the program using locking primitives. The programmer attempts to enforce the atomicity that is required for the algorithm to be correct. However this is error-prone, causing hard-to-find bugs in the application. Trying to simplify the solution can result in coarse synchronisation and the loss of the benefits of parallelism.

The atomic section is a much higher level primitive, which programmers can use to directly specify the atomicity requirements of their algorithms. Other threads will never interfere with code in an atomic section. This means there is no possibility of bugs such as deadlocks and race conditions, that can occur when the programmer is forced to implement such a guarantee. The implementation of atomic sections uses transactions to rollback the state if interference occurs, but this is slow without hardware support, cannot support IO in an atomic section, and may have undesirable performance characteristics when threads contend heavily for a resource.

We propose an implementation of atomic sections that uses locks to ensure the atomicity of object oriented code. We define a static analysis on a simple object calculus without dynamic binding. This infers the objects touched by an atomic section, in terms of paths through the initial heap. At run-time these paths are evaluated and the objects "locked" for the duration of the atomic section.

Sharon Curtis, Oxford Brookes University

Multirelational folds

Multirelations are isomorphic to predicate transformers, but provide a different perspective when it comes to the algebraic treatment of angelic and demonic non-determinism. Folds in multirelations take a form very familiar to functional programmers.

Neil Datta, Imperial College London

Computational idioms, symmetry, reversibility and K-theory

Operator algebras have been introduced over the last decade as candidate operational or denotational semantical domains for quantitative extensions of programming languages. Unlike other operator algebras, C*-algebras are topologically stable in infinite dimensions and so it appears they are particularly suitable for the characterisation of recursive behaviour as a limit. Moreover, associated with this topological stability is the property that every C*-algebra is dual to a lattice of projections. There is a general aim to interpret this lattice as an axiomatic semantics which is dual to the operational or denotational perspective encoded in the elements of the C*-algebra.

We are particularly interested in approximately finite-dimensional C*-algebras (AF-algebras). AF-algebras are defined as the limit of an infinite sequence of increasing finite dimensional C*-algebras; each algebra is embedded into its successor by means of an involutive (*-) homomorphism. There is a famous classification theorem for AF-algebras which states that each *-isomorphism equivalence class of AF-algebras is uniquely associated with a "dimension group" (the algebra's "K-theory") arising from its projective structure. We would like to apply this theorem to the long-standing question of comparing the expressiveness of programming languages in a more refined way than merely stating whether or not a language is Turing complete. We suppose that the dimension group somehow characterises the idiom and relative richness of the computational properties or situations that are expressible in a language.

The topological stability of C*-algebras is a consequence of a symmetry constraint arising from the "*" operation. The computational nature of this constraint is not yet clear but is related to reversibility. The behaviour of programs is in general irreversible; it is interesting that naïve AF-encodings of a range of reasonably expressive programming languages produce the same AF-algebra. We will discuss the application of the inherent symmetry of C*-algebras in distinguishing different computational idioms.

Aleksandar Dimovski, University of Warwick

Game semantics supported component verification

Game semantics provides algorithms for software model checking. Open programs are modeled by looking at the way in which they can observably interact with their environment. Computation is seen as a game between two players, the environment and the program, and the model of a program is given as a strategy for the second player.

One of the most important features of game semantics models is compositionality. The model of a large program is constructed from the models of its constituting components (sub-programs). This facilitates using compositional (i.e. divide-and-conquer) verification techniques that decompose the verification of a large program into manageable subparts.

We present a technique for performing compositional verification in an iterative and fully automated fashion; the approach uses learning and model checking.

Mike Dodds, University of York

Graph transformation in constant time

Graph transformation systems are applicable to a wide range of problems including pointer safety, pattern recognition, and forming the basis of graph programming languages. They have the disadvantage, however, that the application of a rule generally requires the construction of a subgraph isomorphism, a problem known to be NP-complete (or polynomial if rules are considered as fixed). In this talk I will discuss so-called "rooted" graph transformation rules, a restricted form of rules where applicability can be checked in constant time (assuming fixed rules). Rooted rules define a set of root vertex labels which can appear at most once in the graph to be transformed, and require that all matched nodes are reachable from some root. I will show that, despite these restrictions, rooted rules are surprisingly general. I will discuss using rooted rules to simulate unrooted rules, and show that rooted rules are general enough for Turing machine simulation. I will also discuss possible applications of rooted rules, in particular in modelling pointer manipulations.

This is joint work with Detlef Plump.

Alastair Donaldson, University of Glasgow

General techniques for symmetry reduction in model checking

Model checking is a potentially useful technique for verifying designs of concurrent systems, but is limited by the state-explosion problem: as the number of components in a concurrent system grows, the state-space of a model of the system suffers combinatorial explosion. Replication in the system being modeled may induce symmetries on the global state-space of a model, and if this replication can be identified in advance, model checking can be performed over a, generally smaller, quotient state-space, consisting of one state per equivalence class with respect to the symmetry. The success of symmetry reduction depends on efficient techniques to compute orbit representatives, and the problem of representative computation is known to be NP-hard.

In this talk, we present exact, efficient solutions to this problem for certain classes of symmetry group, and approximate solutions for arbitrary symmetry groups. We describe an approach to classifying an arbitrary symmetry group based on its structure as a disjoint or wreath product of subgroups, so that an appropriate symmetry reduction strategy can be chosen for the group. We briefly describe TopSPIN, a new symmetry reduction package for the SPIN model checker which implements our techniques.

This is joint work with Alice Miller.

Martin Dyer, University of Leeds

The complexity of counting homomorphisms to directed acyclic graphs

For a fixed (di)graph H chosen from some class, we consider the complexity of the problem of counting the number of homomorphisms to it from some graph G taken from some, possibly different, class. The usual aim is to prove a "dichotomy theorem" showing that for some H the problem is in P and for every other H the problem is #P-complete. We review previous work on this problem, then describe a new result for the problem of counting homomorphisms to directed acyclic graphs, providing a graph-theoretic classification of the "easy" graphs. An interesting feature of the classification, which is absent from previous dichotomy results, is that there is a rich supply of tractable graphs with complex structure.

Edith Elkind, University of Warwick

Nash equilibria in graphical games on trees, revisited

Graphical games have been proposed as a game-theoretic model of large-scale distributed networks of non-cooperative agents. When the number of players is large, and the underlying graph has low degree, they provide a concise way to represent the players' payoffs. It has recently been shown that the problem of finding Nash equilibria on a general degree-3 graphical game is complete for the complexity class PPAD, indicating that it is unlikely that there is any polynomial-time algorithm for this problem. We show here that in contrast, degree-2 graphical games are tractable.

Our algorithm uses a dynamic programming approach, which was introduced by Kearns, Littman and Singh in the context of graphical games on trees. The algorithm of Kearns et al. is a generic algorithm which can be used to compute all Nash equilibria. The running time is exponential, though approximate equilibria can be computed efficiently. Kearns et al. proposed a modification to the generic algorithm in order to find a Nash equilibrium in polynomial time, provided that the underlying graph is a bounded-degree tree. We show that this modified algorithm is incorrect: the output is not always a Nash equilibrium.

In this talk, we focus on graphical games in which the underlying graph is a path or "path-like". First, we show that Nash equilibria can be computed in quadratic time if the underlying graph is a path, and therefore in polynomial time if the underlying graph has maximum degree 2. Our algorithm, which is based on the approach of Kearns et al., can be used to compute Nash equilibria of graphical games on arbitrary trees, but the running time can be exponential, even when the tree has bounded degree. We show that this is inevitable: any two-pass algorithm of this type will take exponential time, even on bounded-degree trees with pathwidth 2.

It is an open question whether our algorithm runs in polynomial time on graphs with pathwidth 1 but we show that finding a Nash equilibrium for a graphical game in which the underlying graph has maximum degree 3 and constant pathwidth is PPAD-complete (and hence is unlikely to be tractable).

This is joint work with Leslie Ann Goldberg and Paul Goldberg.

Simon Foster, University of Sheffield

A formal model for web-service composition

Orchestration describes the rules which define the patterns underlying the way in which a composite web-service interacts with other services, thereby enabling composite functionality. In this talk we look at Cashew-S, a language based on OWL-S, which we have created for specifying orchestration with formal semantics provided by a timed process calculus, and examine how this fits into the general theory of service composition via processes. We believe that this research will eventually enable us to give a formal model for choreography, the method by which services engage in conversations, and will further aid in automated service composition.

Stephen Gorton, University of Leicester

Task-oriented business requirements elicitation for web services

The expression of semantically-rich business requirements for web services is restricted by current composition and management solutions available, e.g. BPEL. These solutions often address orchestration concerns, rather than actual requirements. Methods such as BPMN, although able to express requirements in terms of business activities and flows, are unable to express non-core requirements such as resource and performance constraints. In this talk, we will present a language to accurately express business requirements specifications through the use of graphical notation and policies. We will present a business goal G, which defines a set of tasks T, a task map m and a set of operators O. Subsequently, we will explain how this method can map to current composition and management solutions such as graphical notations and BPEL.

Alexey Gotsman, University of Cambridge

On proving liveness properties of programs

We describe a new counterexample-guided refinement-based algorithm for proving liveness properties of programs built upon an extension of a recently discovered technique for proving program termination. Our implementation of the algorithm provides an automatic, interprocedural, path sensitive and context-sensitive liveness prover for the C programming language. It supports such language features as arbitrarily nested loops, arbitrarily nested recursive functions, pointers and side-effects, and function-pointers.

This is joint work with Byron Cook, Andreas Podelski, and Andrey Rybalchenko.

Jonathan Grattage, University of Nottingham

QML: A functional quantum programming language

QML is a high-level functional quantum programming language that allows the contraction of quantum variables, in apparent contradiction of the no-cloning theorem of quantum mechanics. Of course, no cloning actually takes place. The contraction, or non-linear use of variables, is modeled as sharing, as with most programming languages. This is achieved by the use of two if-then-else constructs: a classical-if, which measures quantum data; and a quantum-if, which allows quantum parallelism and entanglement. We show how these constructs allow contraction-as-sharing, and briefly discuss the issues of orthogonality with regard to judgments needed for the quantum-if.

Alexander Green, University of Nottingham

Reversible quantum circuits from irreversible functions

Quantum circuits are, by their nature, reversible. It is possible to create reversible circuits that perform irreversible functions. These transpositions often require additional input qubits ("heap"), and produce extra output ("garbage"), that allows the function to be reversible. In this talk we give three laws governing circuits containing heap and garbage, show how they can be optimised, and show how they can be used to prove the measurement postulate. The three laws also hold for classical reversible circuits, but other laws are also satisfied such as that the measurement of a bit has no effect on other bits in the circuit.

Abubakar Hassan, King's College London

Compiling interaction nets

Interaction nets have been put forward as both a graphical programming paradigm and as an intermediate language into which we can compile other languages. Although we can already program in interaction nets, they still lack what modern programming languages should offer. In this talk, we address two issues. Firstly, we consider how to expand this programming paradigm to become a useful and usable programming language, providing features such as a module system, data types, input/output etc. Secondly, we consider how to compile such a language by giving a compilation scheme of interaction nets to bcode (our target/intermediate language) which can be executed by our abstract machine, or further be translated into Java bytecodes for execution by a Java virtual machine. We conclude the talk by giving a formal correctness proof of the compiler.

Michaela Heyer, University College Cork

An intelligent example-generator for graph theory

As with most areas of mathematics, graph theory relies heavily on the use of examples and counterexamples to prove, disprove or support new conjectures. The intelligent example-generator has a way of providing these example graphs in a very efficient manner by combining different aspects of computer science and mathematics. In this talk I give a broad overview of the areas involved and describe the main steps in the process of generating the example graphs: the generation of all graphs of a given size; the use of automated logical inference together with expert knowledge of graph theory to greatly reduce the amount of work to be done; and the use of a 100 node Beowulf cluster to speed up the process through parallel testing of graph properties.

Catherine Hope, University of Nottingham
Fusion in less space

There are many advantages to writing functional programs in a compositional style, such as clarity and modularity. However, the resulting programs can be inefficient in terms of space due to the use of intermediate data structures. These structures may be removed using deforestation techniques, but whether the space performance is actually improved depends on the structures being consumed in the same order that they are produced. In this talk I explore this problem and present a solution.

Wan Huang, London School of Economics
Enumerating Nash equilibria for game trees

We develop an algorithm for finding all Nash equilibria in a game tree with imperfect information. The algorithm is based on the “sequence form” which was introduced by von Stengel, and employs linear programming duality and polyhedral theory. The sequence form represents “mixed strategies” in game trees compactly, with exponentially fewer variables than the strategic form. The Nash equilibria are the solutions to an optimization problem (called a linear complementarity problem) derived from the sequence form. We show how to remove all the redundant variables and some of the redundant constraints of this optimization problem by considering terminal sequences of moves. We also give a proof that all Nash equilibria correspond to some convex combination of certain vertices in the polyhedra.

Andrew Hughes, University of Sheffield
Combining timing, localities and migration in a process calculus

Process calculi provide an abstract representation of concurrency, and have been used in both theoretical and practical contexts. Since the early days of CCS, CSP and ACP, process algebras have diverged into two groups: those that add the concept of time, and those that bring in mobility. The latter has been represented in two different ways: as scope mobility (most notably in the pi-calculus) and as migration.

My research attempts to combine these two separate ideas within a single process calculus. This continues my earlier work with the CaSE and Cashew Nuts calculi developed at Sheffield. These both have a notion of time, represented by clock hierarchies, but do not include mobility. I will discuss these briefly, before demonstrating the use of other calculi which allow the representation of distribution, via locality, and process migration. I will also consider how these ideas might be combined.

Markus Jalsenius, University of Warwick
Improved mixing bounds for the anti-ferromagnetic Potts model on Z^2

We consider the anti-ferromagnetic Potts model on the integer lattice Z^2. The model is used in statistical physics and corresponds to graph colourings with two parameters, q and λ: parameter q is the number of colours and $\lambda \in [0, 1]$ is used to weight colourings. We are interested in sampling from the set of q-colourings on Z^2 where each colouring is assigned a probability proportional to its weight. In the case $\lambda = 0$ we are sampling from the uniform distribution on proper q-colourings. If $\lambda = 1$ we are sampling from the uniform distribution on all q-colourings. We use Markov chains, Glauber dynamics, to sample colourings. Each state corresponds to a colouring and the stationary distribution is identical to the distribution we want to sample from. It is known that Glauber dynamics is rapidly mixing if $q > 7$, $\lambda \in [0, 1]$, or if $q = 7$, $\lambda = 0$, $\lambda > 1/8$, or if $q = 6$, $\lambda = 0$, $\lambda > 1/4$. We show that Glauber Dynamics is rapidly mixing for $q \geq 6$ and any $\lambda \in [0, 1]$. We also show rapid mixing for a larger range of λ than was previously known for $q = 3, 4$ and 5.

This is joint work with Leslie Ann Goldberg, Russell Martin and Mike Paterson.

Mauro Jaskelioff, University of Nottingham
Towards operations on operational semantics

Standard structural operational semantics has poor modularity. We build upon the work of Turi's functorial operational semantics, an abstract category-theoretical model of operational semantics which guarantees that the defined semantics are well behaved. Working in this abstract setting we reason about the combination of operational semantics and we define operations for achieving it. We show how to use these operations to combine four toy languages featuring different effects.

Kenneth Johnson, Swansea University
The theory of spatial data types and constructive volume geometry

Spatial data types model data in space. There are a vast number of examples in computing, ranging from medical images to computer memories. Spatial data types are modeled using algebras of total functions from a topological space of points to a topological algebra of attributes.

In this talk, we motivate a general theory of spatial data types by introducing Constructive Volume Geometry (CVG), an algebraic framework for the high-level programming of spatial objects in graphics. We discuss the problem of the expressive power of CVG in the context of the 4-colour channel model: Do the CVG terms define all the spatial objects one wants?

We sketch some general theory and show how to solve the expressiveness problem in general. Using variants of the Stone-Weierstrass Theorem we prove a density result which shows that some subsets of spatial objects under certain primitive operations can approximate all other spatial objects.

Oliver Kullmann, Swansea University
Using (hyper)graph decomposition for satisfiability decision

Given a hard satisfiability problem F (boolean or not), an interesting option is to use some (hyper)graph representation $G(F)$ with the property that, if $G(F)$ splits into connected components, then we can work on the components independently. In this talk, I present a unifying approach, looking at the whole picture from "local" hypergraph cuts to "global" tree decompositions (typically associated with fixed parameter tractability). In this way the different decomposition strategies can be much better linked to the "logical" properties of the problem F (as opposed to the "graphical" properties represented by $G(F)$); we identify the missing link here as the cause for the (hitherto) failure of graph decomposition algorithms for practical applications (while being attractive in theory).

Ranko Lazic, University of Warwick

LTL *with the freeze quantifier and register automata*

Temporal logics, first-order logics, and automata over data words have recently attracted considerable attention. A data word is a word over a finite alphabet, together with a piece of data (an element of an infinite domain) at each position. Examples include timed words and XML documents. To refer to the data, temporal logics are extended with the freeze quantifier, first-order logics with predicates over the data domain, and automata with registers or pebbles.

We investigate relative expressiveness and complexity of standard decision problems for LTL with the freeze quantifier, 2-variable first-order logic (FO^2) over data words, and register automata. The only predicate available on data is equality. Previously undiscovered connections among these formalisms, and to counter automata with incrementation errors, enable us to answer several questions left open in recent literature.

We show that the future-time fragment of LTL with freeze which, corresponds to FO^2 over finite data words, can be extended considerably while preserving decidability, but at the expense of non-primitive recursive complexity, and that most further extensions are undecidable. We also prove that, surprisingly, over infinite data words, LTL with freeze and without the "until" operator, as well as nonuniversality of one-way nondeterministic register automata, are undecidable even when there is only 1 register.

This is joint work with Stephane Demri (CNRS & ENS Cachan & INRIA Futurs).

Cindy Li, University of Liverpool

Efficient probe selection in microarray design

DNA microarray technology, originally developed to measure the level of gene expression, is becoming one of the most widely used tools in genomic study. Microarrays have been proved to benefit areas including gene discovery, disease diagnosis, and multi-virus discovery. The crux of microarray design lies in how to select a unique probe that distinguishes a given genomic sequence from other sequences. However, in cases that the existence of a unique probe is unlikely, e.g. in the context of a large family of closely homologous genes, the use of a limited number of non-unique probes is still desirable.

Due to its significance, probe selection attracts a lot of attention. Various probe selection algorithms have been developed in recent years. Good probe selection algorithms should produce as small a number of candidate probes as possible. Efficiency is also crucial because the data involved is usually huge. Most existing algorithms usually select probes by filtering, which is usually not selective enough and quite a large number of probes are returned. We propose a new direction to tackle the problem and give an efficient algorithm to select (randomly) a small set of probes and demonstrate that such a small set of probes is sufficient to distinguish each sequence from all the other sequences. Based on the algorithm, we have developed a probe selection software RandPS, which runs efficiently and effectively in practice. A number of experiments have been carried out and the results will be discussed.

This is joint work with Leszek Gasieniec, Paul Sant and Prudence WH Wong.

Olga Lightfoot, Queen Mary, University of London

Real arithmetic test suite for a theorem prover

Theorem proving has been used with great success in system verification, in particular, in reasoning about problems expressed in terms of continuous mathematics over the real numbers. Such mathematical expressions lie in the domain of higher order logic, so we are interested in the performance of theorem provers in handling operations on functions in the domain of real numbers. Despite the good level of automation of linear arithmetic operations, non-linear operations over both algebraic and transcendental functions still present a considerable challenge. We discuss the construction of a test suite for evaluating the real arithmetic capabilities of an interactive theorem prover and, using PVS, show what can be deduced about a theorem prover from such a test.

David Manlove, University of Glasgow

Vertex and edge covers with clustering properties: complexity and algorithms

We consider the concepts of a t-total vertex cover and a t-total edge cover ($t \geq 1$), which generalize the notions of a vertex cover and an edge cover, respectively. A t-total vertex (respectively edge) cover of a connected graph G is a vertex (edge) cover S of G such that each connected component of the subgraph of G induced by S has at least t vertices (edges). These definitions are motivated by combining the concepts of clustering and covering in graphs. Moreover they yield a spectrum of parameters that essentially range from a vertex cover to a connected vertex cover (in the vertex case) and from an edge cover to a spanning tree (in the edge case). For various values of t, we present NP-completeness and approximability results (both upper and lower bounds) and FPT algorithms for problems concerned with finding the minimum size of a t-total vertex cover, t-total edge cover and connected vertex cover, in particular improving on a previous FPT algorithm for the latter problem.

This is joint work with Henning Fernau (University of Trier).

Erik Arne Mathiesen, Queen Mary, University of London

Abstract Hoare logic and dynamical systems

Setting out to define a Hoare logic for dynamical systems, we define an abstraction of Hoare logic in the setting of traced monoidal categories. More particularly, we define a class of subcategories of the category of posets and monotone mappings on which we define a sound and complete system of inference rules. This provides us with a way of generating Hoare-logic-like rules for general systems through embeddings into the before-mentioned class of subcategories. A particular instance is the embedding of the traced monoidal category of while programs which yields Hoare's original logic. Of special interest to us are embeddings of certain dynamical systems. We conclude by discussing further aspects of the framework such as partial vs total correctness.

Neil Mitchell, University of York

CATCH - Case And termination check for Haskell

There are two main ways in which a Haskell program may fail at runtime. Firstly, the program may not terminate. Secondly, if the program has any incomplete (non-exhaustive) patterns in definitions or case alternatives then it may encounter a pattern match error.

This work presents an automated analysis which checks a Haskell program, and attempts to generate a proof that the Haskell program encodes a total function. It includes a constraint language that can be used to reason about the data in a Haskell program, along with mechanisms to propagate these constraints between program components.

Matthias Mnich, London School of Economics

Computation of correlated equilibria in succinctly-representable games

A correlated equilibrium is a more general notion of a Nash equilibrium modelling the simplest form of co-operation in a game via "shared randomness". The question of whether a polynomial-time algorithm exists for computing correlated equilibria in succinctly representable games has recently been solved by Papadimitriou (STOC 2005). This approach applies certain properties of the ellipsoid algorithm for linear programming that allows it to run on a problem with polynomially-many unknowns and exponentially many constraints. We give an overview of that idea and also include other results encompassing other classes of games. Finally, possibilities to use this algorithm for game trees with imperfect information (called extensive games) are discussed, where the number of pure strategies may be exponential in the size of the extensive game.

Peter Morris, University of Nottingham

Containing families

We examine the idea of indexed-containers from a programming perspective. We show how to use this notion to characterise inductive families in the dependently-typed functional language Epigram. This gives us a flexible and compositional semantic notion of Strict Positivity and enables us to write generic programs not in a type system we model but for the full Epigram type system.

Dimitrios Mostrous, Imperial College London

Session types for object-oriented languages

A session takes place between two parties; after establishing a connection, each party interleaves local computations with communications (sending or receiving) with the other party. Session types characterise such behaviour in terms of the types of values communicated and the shape of protocols, and have been developed for the pi-calculus, CORBA interfaces, and functional languages. We study the incorporation of session types into object-oriented languages through the language Moose, a multi-threaded language with session types, thread spawning, iterative and higher-order sessions. Our design aims to consistently integrate the object-oriented programming style and sessions, and to be able to treat various case studies from the literature.

We describe the design of Moose, its syntax, operational semantics and type system, and develop a type inference system. After proving subject reduction, we establish the progress property: once a communication has been established, well-typed programs will never starve at communication points.

This is joint work with Mariangiola Dezani (University of Turin) and Nobuko Yoshida and Sophia Drossopoulou (Imperial College).

Jonty Needham, University of Bath

A fully abstract game semantics for answer set programming

We present a fully abstract interaction model of answer set programming based on logic games semantics. The field of games semantics has proved to be a powerful mathematical framework in which to view a wide variety of programming languages and logics. We present an outline of games semantics and then prove the correctness result for the denotation. We also present a result about providing an alternative algorithm for grounding which reduces computation time, as well as the correctness of a debugging algorithm.

Gregg O'Malley, University of Glasgow

Stable matching problems with constant length preference lists

The Stable Marriage problem (SM) and many of its variants have been studied extensively in the literature. In recent years much research has focused on the model SMTI, where a given participant's preference list may involve ties and be incomplete. It is currently known that, in this setting, stable matchings may have different sizes, and the problem of finding a maximum stable matching is NP-hard. In this talk we consider various restrictions of SMTI where some of the preference lists are constrained to be of constant length. Such restrictions arise naturally in practical applications. We prove that in some cases, finding a maximum stable matching can be achieved in polynomial time, and for others we show that the problem remains NP-hard.

Nick Palmer, University of Warwick

PAC-learnability of probabilistic deterministic finite state automata in terms of variation distance

We consider the problem of PAC-learning distributions over strings, represented by probabilistic deterministic finite automata (PDFAs). PDFAs are a probabilistic model for the generation of strings of symbols, that have been used in the context of speech and hand-writing recognition and bioinformatics. Recent work on learning PDFAs from random examples has used KL-divergence as the error measure; here we use variation distance. We build on recent work by Clark and Thollard, and show that the use of variation distance allows simplifications to be made to the algorithms, and also a strengthening of the results; in particular that using variation distance, we obtain polynomial sample size bounds that are independent of the expected length of strings.

Nick Papanikolaou, University of Warwick
A framework for automated verification of quantum cryptographic protocols

This talk will consider the problem of proving correctness and security properties for communication protocols, and cryptographic protocols in particular, with a view to showing how formal methods may be used in the analysis of schemes for quantum communication. Quantum cryptographic techniques rely on the laws of quantum mechanics to establish a secret key between two users; indeed, several protocols implementing these techniques have been shown to be perfectly secure in the information–theoretic sense.

We will give an account of the process algebra CQP (Communicating Quantum Processes) and how it may be used to build accurate models of quantum protocols; also, the verification of protocol models using a probabilistic model-checker will be discussed. Finally, we will report on our progress in developing an integrated framework for simulating and verifying properties of systems with finite-dimensional quantum state spaces.

This is joint work with Rajagopal Nagarajan (University of Warwick) and Simon Gay (University of Glasgow).

Mike Paterson, University of Warwick
Overhang

How big an overhang beyond the edge of the table can we reach by stacking n identical blocks of length 1? The classical solution achieves an overhang of $1/2H_n$, where $H_n \approx \ln n$ is the n^{th} harmonic number. This solution is widely believed to be optimal. We show that it is exponentially far from optimal.

This is joint with Uri Zwick (Tel Aviv University).

Daniel Paulusma, Durham University
Locally constrained graph homomorphisms and degree refinement matrices

We consider three types of locally constrained graph homomorphisms: bijective, injective and surjective. Degree refinement matrices have tight connections to locally constrained graph homomorphisms. If a graph G has a homomorphism of a given type to a graph H then we say that the degree refinement matrix drm(G) of G is smaller than drm(H). In this way we obtain three partial orders. Computing drm(G) is easy, so an algorithm deciding comparability of two matrices in one of these partial orders would be a heuristic for deciding if G has a homomorphism of a given type to H. For the locally bijective constraint this corresponds to a well-known heuristic for graph isomorphism. For local surjectivity and injectivity we show that the problem of matrix comparability belongs to the complexity class NP.

This is joint work with Jiří Fiala and Jan Arne Telle.

Kasper Pedersen, University of Warwick
A scan Markov chain for sampling colourings

Consider a graph G with maximum vertex degree Δ. A proper q-colouring of G is an assignment of colours to the vertices of G such that no edge is monochromatic. Calculating the exact number of proper q-colourings of a graph is #P-complete but this number can approximated by sampling from the uniform distribution of proper q-colourings, π, provided that q is sufficiently large. Sampling from π is done by simulating a Markov chain with state space Ω and stationary distribution π where Ω is the set of all q-colourings of G. The mixing time of a Markov chain is how long it takes to get close to its stationary distribution.

We are interested in discovering Markov chains that (1) are mixing in as few steps as possible and (2) succeed in mixing for as few colours as possible. It is known that whenever $q > (11/6)\Delta$, a random update Markov chain (in which, during each step, one vertex is chosen uniformly at random and updated) mixes in $O(n \log n)$ steps (Vigoda, 2000). We study the mixing time of “scan” Markov chains. A Markov Chain is a scan if the vertices of G are visited in an order specified by some permutation which must remain constant during each sweep. It has been shown that scan mixes in $O(\log n)$ steps when $q > 2\Delta$ and in a polynomial number of steps when $q = 2\Delta$ (Dyer, Goldberg and Jerrum, 2005).

In this talk we present a scan which mixes in $O(nlogn)$ steps provided $q \geq 2\Delta$.

Doron Peled, University of Warwick
Efficient model checking for LTL *with partial order snapshots*

Certain behavioural properties of distributed systems are difficult to express in interleaving semantics, whereas they are naturally expressed in terms of partial orders of events or, equivalently, in terms of Mazurkiewicz traces. Examples of such properties are serializability of a database or snapshots.

Recently, a modest extension of LTL by an operator that expresses snapshots has been proposed. It combines the ease of linear (interleaving) specification with this useful partial order concept. The new construct allows one to assert that a global snapshot (also called a slice or a cut) was passed, perhaps not in the observed (interleaved) execution sequence, but possibly in a (trace) equivalent one. A model checking algorithm was suggested for a subset of this logic, with PSPACE complexity in the size of the system and the checked formula. For the whole logic, a solution that is in EXPSPACE in the size of the system (PSPACE in the number of its global states) was given.

In this talk, we present a model checking algorithm which is PSPACE in the size of a system of communicating sequential processes when restricting snapshots to boolean combinations of local properties of each process. Concerning the size of the formula, it is PSPACE for the case of snapshot properties expressed in DNF, and EXPSPACE where a translation to DNF is necessary.

This is joint work with Peter Niebert (Marseille).

Alexis Petrounias, Imperial College London

The small chorded object-oriented language

The chord construct is a concurrency mechanism inspired by the join from the Join-Calculus. Chords were implemented in an extension of C# called Polyphonic C#, and also are available in COmega. They promise to raise the level of abstraction concurrent programs are written in, hence offering a more powerful means of reasoning about the behaviours of such programs.

Furthermore, the inclusion of chords in COmega means they will be used in conjunction with traditional, imperative concurrency constructs such as monitors and threads. In order to study the interactions between other language constructs and chords it is necessary that we fully understand the behaviour of chords. We therefore provided a formal model describing the fundamental semantics of chorded languages, namely the Small Chorded Object-Oriented Language (SCHOOL), to our knowledge the first formalisation of a chorded language.

In this talk I will give an introduction to chords, show what programming with chords looks like, and briefly describe the formal model of chorded programming languages known as Simplified SCHOOL.

Pattarawit Polpinit, University of Warwick

Comparing parallel and sequential selfish routing in the atomic players setting

We consider the problem of routing traffic flow to optimize the performance of a congested network. In particular, we specialize the traffic model with atomic players where each player controls an amount of splittable flow, and aims to minimize their own cost. We are given a network and a linear cost function for each edge. The objective is to compare the optimal social cost with the cost arising from selfish routing decisions by the players.

In this work, we compare a game with two different settings: a parallel setting and a sequential setting. The sequential setting represents a set of flows where all players make decisions sequentially while the parallel setting is a set of flows at Nash equilibrium, i.e. the optimal point for the player given the other players' flows. We prove that in a two-players two-link network model, the social cost of the sequential setting is at most 9/8 times the minimum possible for the parallel setting. We also consider the more general setting in which there are m players in the model.

Sam Sanjabi, University of Oxford

Full abstraction for additive aspects

Aspect-Oriented Programming (AOP) is an emerging programming paradigm that has, from the practitioner's point of view, been extensively studied in the last few years. It allows programmers to "weave" fragments of new code into existing code without modifying the base code. These fragments can potentially exchange data with the base code, or even elide its execution altogether. It has therefore become apparent that current aspect oriented languages such as AspectJ (an aspect-oriented extension of Java) – while providing a powerful programming tool – also break many desirable modularity principles, making it almost impossible to reason about code in the presence of aspects. It seems therefore necessary to subject aspect-orientation to some degree of formal analysis in order to learn to control program behaviour, while retaining as many of the benefits of the paradigm as possible. However, theoretical study – while increasingly active – has lagged far behind implementation in this field.

In this talk, I shall present (to my knowledge) the first known fully abstract game semantics for a non-trivial fragment of AOP: additive aspects. Additive aspects are those which do not prevent the base computation from executing, but may otherwise exchange information with it in any way. The full abstraction result is achieved (in the style of McCusker) by demonstrating the existence of a compositional, fully abstract translation between a simple language of additive aspects, and Idealised Algol with General References (an imperative language known to have an existing games model). After presenting the main theorem, I discuss some the implications, applications, and potential extensions of the result.

Paul Sant, University of Luton

Combinatorics of colouring 3-regular trees

An instance of the Colouring Pairs of Binary Trees Problem (CPBT) consists of two 3-regular trees, both with *n* leaves. The challenge of CPBT is to show that, for every instance of the problem, there exists 3-edge-colourings of the trees such that the sequence of colours associated with root-edges in both trees is the same. Interest in the problem stems from its equivalence to a number of other important combinatorial problems including, specifically, the historically-famous 4-colour problem of planar maps. We present a number of recent results, both combinatorial and algorithmic, related to CPBT and pose a number of challenges.

This is joint work with Alan Gibbons (King's College London).

Rahul Savani, London School of Economics

Hard-to-solve bimatrix games

A bimatrix game is a two-player game in strategic form, a basic model in game theory. A Nash equilibrium is a pair of (possibly randomized) strategies, one for each player, so that no player can do better by unilaterally changing his or her strategy. The problem of finding one Nash equilibrium of a bimatrix game is considered as "one of the most important concrete open questions on the boundary of P today" (Papadimitriou, 2001).

In this talk, we show that the commonly used Lemke-Howson algorithm for finding one equilibrium of a bimatrix game is *not* polynomial. This question had been open for some time. The algorithm is a pivoting method similar to the simplex algorithm for linear programming. We present a class of square bimatrix games for which the shortest Lemke-Howson path grows exponentially in the dimension *d* of the game. We construct the games using pairs of dual cyclic polytopes with 2*d* facets in *d*-space.

This is joint work with Bernhard von Stengel.

Anton Setzer, Swansea University

Inductive recursive definitions and generic programming

Inductive-recursive definitions were originally introduced by Peter Dybjer in order to capture the general principle for defining sets in Martin-Löf type theory. They generalise the notion of a strictly positively inductively defined set by allowing to define a set inductively while simultaneously defining recursively an element of another type (which could be a set, the type of sets, or a higher type like the type of functions mapping sets to sets). Together with Peter Dybjer, the author has developed a closed formalisation of inductive-recursive definition, which gives rise to a condensed definition of inductive-recursive definitions. Although inductive-recursive definitions were originally introduced in the context of dependent type theory, their relevance seems not to be restricted to dependent types.

In the specific case where the recursively defined object is a set we obtain a relative general notion of generic functions. The set of elements of the inductive-recursively defined set are codes for data types, and the recursively defined set determines for each code the data type it denotes. A generic function can take a code for a data type and an element of the data type it denotes, and compute from it a code for another data type and an element of the data type it denotes, where the target code could be given by an inductive-recursive definition different from the the one used by the arguments. Some examples of generic functions will make use of inductive-recursive definitions of higher types.

In this talk we introduce the notion of inductive-recursive definitions and give some examples of generic functions definable using inductive-recursive definitions.

Nikolaos Siafakas, King's College London

A fully labeled lambda calculus

Levy's labeled lambda calculus has been of major importance in the research of efficient implementations of functional programming languages. If we compute the normal form of a term then the resulting label will describe a path in the graph of the term. The Geometry Of Interaction (GOI) machine, which is an implementation of Girard's Geometry of Interaction semantics for Linear Logic, will follow exactly the path described by the label. The investigation of the structure of the labels has led to optimised versions of the GOI machine, such as the Jumping Abstract Machine (JAM), where the length of the path to be traversed is significantly reduced. However, the structure of the labels is different from the structure of the paths. The latter depend on the choice of translation of the lambda calculus into Linear Logic proof nets (call-by-value or call-by-name) and the optimisations rely on the transposition of the structural information obtained from the labels into paths. For each translation we present a fully labeled lambda calculus which unifies the information yielded from the paths with the structure provided by Levy's labels. Our main goal is to find new and efficient ways of computing the paths in the GOI machine.

Alexandros Skaliotis, King's College London

Logarithmic simulated annealing for protein folding

Protein folding is the process by which a sequence of amino-acids conforms to a three-dimensional shape that determines the biological function of the resulting protein. We consider the problem of predicting these conformations based on the hydrophilic-hydrophobic model introduced by Dill et al. in 1995. A problem instance consists of a chain of amino-acids, each labeled “H” (hydrophobic) or “P” (hydrophilic). This sequence has to be placed in a 2D or 3D grid in a non-overlapping way that preserves the original adjacencies of the amino acids. Following Anfinsen’s hypothesis which suggests that proteins fold to a minimum energy state, the goal is to minimise the overall energy. In the simplest variation, this corresponds to maximising the number of adjacent hydrophobic pairs. The protein folding problem in the HP model is NP-hard in both 2D and 3D. In 2004, Fu and Wang gave an $\exp(O(n^{1-1/d})ln(n))$ algorithm for d-dimensional protein folding simulation in the HP-model.

We investigate the application of logarithmic simulated annealing to the problem by employing a set of moves proposed by Lesh et al. in 2003 and Blazewicz et al. in 2005. Albrecht et al. in 2006 show that after $(n/a)^{O(G)}$ Markov chain transitions, the probability of being in a minimum energy conformation is at least $1-a$, where n is the length of the instance and G is the maximum value of the minimum escape height from local minima of the underlying energy landscape. For selected benchmark instances we performed an experimental estimation of values for G. These indicate that $G < n^{1-1/d}$ which is competitive to the bound by Fu and Wang.

This is joint work with Andreas Albrecht (University of Hertfordshire) and Kathleen Steinhofel (King’s College London).

Colin Sng, University of Glasgow

Popularity in the capacitated house allocation problem

We consider the problem of finding a popular matching in the Capacitated House Allocation problem (CHA), in which we have a set of agents and a set of houses. Each agent has a preference list ranking a subset of houses in some order of preference, and each house may be matched to a number of agents that must not exceed its capacity. A matching M is popular if there is no other matching M' such that the number of agents who prefer their houses in M' to M exceeds the number of agents who prefer their houses in M to M'. Here, we give a polynomial-time algorithm to determine if an instance of CHA admits a popular matching, and to find a largest such matching if one exists. Specifically, the complexity of the algorithm is $O(r^{3/2}s^{1/2})$ where r is the number of agents and s is the number of houses.

Mike Stannett, University of Sheffield

Future trends in hypercomputation

Hypercomputation (the theory of “super-Turing machines”) is a rapidly expanding area of Theoretical Computer Science, with links to physics, philosophy, biology and mathematics. We present a user-friendly guide to the current state-of-the-art, including quantum computation and cosmological models, and suggest a number of theoretical problems whose solution might prove important to the future development of the subject.

Wouter Swierstra, University of Nottingham

Isomorphisms for context-free types

We can show two types to be isomorphic by constructing explicit coercion functions. What should we do if we cannot find such an isomorphism? There's no way to be sure we're not looking hard enough. I briefly show how a variation of classical monadic parser combinators can be used to provide evidence that two inductive types are not isomorphic.

Traditionally, monadic parser combinators consume an input sequence of symbols. Instead of input strings, however, we consider parsing multisets. Interestingly, we can then interpret the results of our parsers as a powerseries. We can show that two types are isomorphic if and only if their associated powerseries are identical. Distinguishing non-isomorphic types now simply reduces to finding a disparity between two powerseries.

Rick Thomas, University of Leicester

FA-presentable structures

A structure consists of a set together with a collection of relations. For example, a group consists of a set together with a ternary relation (representing the composition of elements in the group), a unary relation (yielding the identity element) and a binary relation (representing the process of taking the inverse of an element). A natural general question is the following: given a structure, can we perform computations in it?

The natural approach would be to take some general model of computation such as a Turing machine. A structure would then be said to be computable if its domain can be represented by a set which is accepted by a Turing machine and if there are decision-making Turing machines for each of its relations. However, there have been various ideas put forward to restrict the model of computation used; whilst the range of structures decreases, the computation can become more efficient and certain properties of the structure may become decidable.

One interesting approach was introduced by Khoussainov and Nerode who considered structures whose domain and relations can be checked by finite automata as opposed to Turing machines; such a structure is said to be "FA-presentable". This was inspired, in part, by the theory of "automatic groups" introduced by Epstein et al; however, the definitions are somewhat different.

We will report on some recent results obtained in conjunction with Graham Oliver and Andre Nies. In particular, we will survey some of what is known about the possible structure of FA-presentable groups and rings.

Ashutosh Trivedi, University of Warwick

Average time games

An average time game is played on the infinite graph of configurations of a finite timed automaton. The two players, Min and Max, construct an infinite run of the automaton by taking turns to perform a timed transition. Player Min wants to minimize the average time per transition and player Max wants to maximize it.

In this work the strategy improvement algorithm for average payoff games on finite graphs is generalized to solve average time games. A direct consequence is an elementary proof of determinacy for average time games. This generalizes results of Asarin and Maler for optimal time reachability games and it partially solves a problem posed by Bouyer et al., to design an algorithm for solving average payoff games on priced timed automata.

Zheng Wang, University of Manchester
A component model for verified software

Component-Based Software Development (CBSD) is a promising approach for assembling pre-existing software components into an integrated software system; this potentially helps to achieve effective software reuse, easy software evolution a and low cost-to-performance ratio. Software verification is a long-standing grand challenge. The aim is to verify software correctness using theorem provers or model checkers. This task is practically impossible for large software systems, for either technical or cost reasons, and is currently not done except for small to medium-sized safety-critical applications. Our research focuses on cross-fertilisation between these two areas: how to make the verification task practically feasible by decomposing and devolving it to software components.

Cross-fertilising CBSD with software verification could offer a practical way to construct verified software from pre-verified components. The central issue is how to design a CBSD methodology that can reuse component proofs via composition operators and the cornerstone of such CBSD methodology is its underlying software component model that provides the semantic framework of components. In order to tackle this issue, we build a software component model that defines the semantics and syntax of software components and their composition. In our component model, components can be built, verified and stored in a repository. Larger components can then be composed from these components using composition operators that carry proof plans that reuse the existing proofs of the components. Thus verification of large systems can be decomposed and devolved to the components, i.e. it can be done by composing or reusing component proofs via proof plans associated with the composition operators.

In this talk I give an overview of our component model and show how it helps to construct verified software from pre-verified components, and demonstrate the feasibility of our component model with an industrial case study: an automatic train protection system.

Taoyang Wu, Queen Mary, University of London
Fixed point free property is* NP*-complete

Considering a permutation group G naturally acting on a finite set X, an element $g \in G$ is called ***fixed point free*** if it doesn't fix any point in X. A natural problem is whether there is any fixed point free element for given G and X. When G is input as a set of generators, we show that the problem is NP-complete via reduction from 3-SAT problem. To this end we also present another NP-complete problem: the solvability of a certain type of incongruences.

This is joint work with Peter Cameron.

Yonghong Xiang, Durham University

Fault-tolerant properties of k-ary n-cube

First, some introduction of fault-tolerant properties of interconnection networks will be introduced. Then, our consideration of finding a longest fault free path in a faulty k-ary n-cube with at most $2n-2$ faults for even integer k and $n \geq 3$ will be given.

Hong Qing Yu, University of Leicester

Semantic web services composition via planning as model checking

The ability to automatically compose services is one essential aspect of service oriented architecture. It can reduce time and cost in development and maintenance of complex services and software systems. We are developing a technique to realize this aim by combining the "planning as model checking" approach with semantic web service concepts. We have modified a planning as model checking algorithm by using a bounded on-the-fly depth-first search algorithm that its possible service execution plans are generated on the fly. One of the challenges is to model a web service as a state transition system. The approach will be suitable in the context of ontologies, but for now we are simply using dictionaries for mapping operations and parameters. The planning as model checking approach forms part of a larger framework to automatically compose services, which addresses several drawbacks of current composition approaches.

Michele Zito, University of Liverpool

Lower bounds for dominating sets in web graphs

In this work we study the size of generalised dominating sets in graph processes that model aspects of the World Wide Web. We show that graphs generated in this way have fairly large dominating sets (i.e., linear in the size of the graph). Various results will be described that enable us to prove increasingly good bounds. The proof techniques we present may be applicable to the study of other combinatorial properties of web graphs.

This is joint work with Colin Cooper (King's College London) and Ralf Klasing (Bordeaux).

Report on UC 2005

The Fourth International Conference on Unconventional Computation 3–7 October 2005, Seville, Spain

Ion Petre

The fourth meeting in the series of international conferences on Unconventional Computation, UC 2005, was held in beautiful Seville, Spain, October 3-7, 2005. The conference was organized under the auspices of EATCS by the Center for Discrete Mathematics and Theoretical Computer Science (Auckland, New Zealand) and the Department of Computer Science and Artificial Intelligence of the University of Seville, Spain. The organizing committee consisted of M. Cavaliere, C. Graciani Dìaz, M.A. Gutièrrez-Naranjo, A. Nepomuceno Fernàndez, Gh. Păun, M.J. Pèrez-Jimènez (chair), F.J. Romero-Campero, A. Riscos-Nùñez, A. Romero Jimènez, F. Sancho Caparrini, D. Sburlan, U. Speidel (registration, Auckland, New Zealand).

The series of conferences on Unconventional Computation `https://www.cs.auckland.ac.nz/CDMTCS/conferences/uc/` is devoted to all aspects of unconventional computation, theory as well as experiments and applications. Typical, but not exclusive, topics are: natural computing including quantum, cellular, molecular, neural and evolutionary computing; chaos and dynamical systems based computing; and various proposals for computations that go beyond the Turing model. The previous three UC conferences were held in Auckland, New Zealand (1998), Brussels, Belgium (2000) and Kobe, Japan (2002).

The scientific program of the conference consisted of 5 invited lectures, 3 tutorials, and 18 contributed papers, covering as varied topics as genetic algorithms, quantum computing, bio-inspired computing, nanotechnology, self-assembly, cellular automata, optical computing, neurocomputing, and others.

The first invited talk was that of L. Grover on “Quantum searching amidst uncertainty”, discussing classical and quantum algorithms for selecting a marked entry from a database with most entries marked. The second invited talk was given by T. Bäck on "Using genetic algorithms to evolve behavior in cellular automata", covering both theoretical issues, as well as impressive applications in industry.

The third invited talk was given by S. Istrail on “Logic functions of the genomic cis-regulatory code", discussing several cis-regulatory processing functions (with examples from sea urchin) viewed as logic operations. He discussed the topic in more details also in his tutorial, with a special stress on the intertwining of discrete and continuous mathematics in the modelling of cellular processes.

The fourth invited lecture was by C. Torras on “Natural inspiration for artificial adaptivity: Some neurocomputing experiences in robotics", discussing some very interesting experimental robotic systems and how their implementation was influenced by biological paradigms. The last invited talk was by N. Seeman on “Structural DNA nanotechnology: Molecular constructions and computations". This was an especially inspiring talk for the computer science dominated audience. Seeman presented several of the major nano-scale constructs achieved in his lab in the last two decades and implications of nanotechnology for computing.

Along with the tutorial by S. Istrail, UC 2005 was host to two other 2-hour tutorials. One was by I. Petre and G. Rozenberg on “Computing with living cells", where the two speakers presented a survey of their research on the mathematical modelling of gene assembly in ciliates, including invariants, assembly power, and gene patterns. The other tutorial was by G. Păun on “Elementary aspects of membrane computing". This tutorial reviewed the main types of P-systems investigated so far, and it also discussed some promising research directions of the P-community, including efforts on systems biology, more biology-driven features in P-systems, and comparisons to other modelling frameworks, in particular ambients and brane calculus.

The most popular topic in UC 2005 was membrane computing. Good presentations were given on issues ranging from simulators (C. Bonchis) and solving numerical decision problems (A. Risco-Nuñèz) to population P-systems (M. Gheorghe), model-checking (Dang et al), and computational efficiency (M.J. Pèrez Jimènez). Quantum computing was also well represented with three good presentations: A. Cabello on communication complexity, M. Nagy on the concept of a universal computer, and T. Tusarova on a new complexity class.

Reflecting the broad scope of the area of unconventional computing, several other research topics were discussed in papers accepted at UC 2005. Good presentations were given on an optical model of computation (D. Woods), counterfactual computing (P. Zuliani), autopoietic automata (J. Wiedermann), graph automata (Tosic and Agha), pseudo-biliard systems (I. Potapov), firing squad synchrinisation problems (K. Kobayashi and also Umeo et al), amorphous computing (M. Hagiya), cellular automata (Inokuchi et al), and self-assembly (N. Jonoska).

Our host at UC 2005, M.J. Pèrez-Jimènez, and his excellent team at the Department of Computer Science and Artificial Intelligence, University of Seville, organized a wonderful social program in which we could stroll through beautiful Seville, but also sample the delights of Spanish cuisine. Our warmest thanks to the whole organizing committee.

UC 2005 was a successful conference of high scientific level, very-well organized. The next UC meeting will take place on September 4-8, 2006, in York, United Kingdom.

Report on OPODIS 2005

The 9th International Conference on Principles of Distributed Systems 12–14 December 2005, Pisa, Italy

Eric Ruppert

This conference was held from December 12 to 14, 2005 in Pisa, Italy, drawing 52 attendees from across Europe, as well as Canada, Israel, Japan and the United States. The conference was hosted by the Università di Pisa, with Giuseppe Prencipe and Vincenzo Gervasi making the local arrangements. The conference covered a wide range of topics in the theory, design and implementation of distributed systems. 30 papers were presented at the conference. These were chosen from among 109 submissions from 30 countries by a 29-member programme committee, co-chaired by James Anderson and Roger Wattenhofer. The progamme also included two invited talks, which reflected two special focusses of the conference this year: wireless networks and real-time systems. Proceedings will be published in Springer's LNCS series.

The conference opened with an invited talk by David Peleg on autonomous mobile robots. He spoke of the advantages of swarms of simple robots, each with limited capabilities: they can be designed to be cheap, expendable and simple. However, they require complex coordination in their movements to perform a task cooperatively. This is a fairly new research area in distributed computing, since earlier work focussed on centralized control, which may not be adequate for large swarms. He spoke of the importance of choosing a model of computation for the design of such algorithms that accurately captures reality.

The second keynote talk was given by Giorgio Buttazzo. He spoke about issues related to real-time systems in mobile networks. For example, he described how scheduling algorithms must be "energy-aware" if the components of the system have limited power sources. One strategy is to use elastic scheduling of periodic tasks, using a range of acceptable periods so that the task can be scheduled at times when the system is not overloaded. More generally, one might accept less precise computations during overload conditions. Another issue that he discussed was the scheduling of transmissions in radio networks to avoid collisions that cause information to be lost.

To give an impression of the rest of the technical programme, I shall mention a sample of some of the papers presented. (Although I only give the names of the presenters, in most cases they were presenting work that they did jointly with co-authors; please refer to the conference proceedings.)

Nir Shavit described an algorithm, which he presented as the missing link in the evolution of distributed linked list implementations, which have progressed from "primaeval" coarse-grained locking, through fine-grained locking, all the way to Maged Michael's state-of-the-art algorithm. Prasad Jayanti gave an algorithm that efficiently implements a collection of many load-link/store-conditional objects from compare-and-swap primitives. Burkhard Englert introduced the notion of memory-adaptivity as an analogue (for space complexity) to the idea of adaptive algorithms which require that time complexity is bounded by a function of contention in the system. In a memory-adaptive algorithm, the index of any register accessed by the algorithm can be bounded by a function of the contention in the system. Ling Cheung showed that randomized consensus can be done more efficiently against an adversarial scheduler that is weaker than the ones that have been studied traditionally.

James Aspnes presented self-stabilizing algorithms for the population protocol model, in which very simple agents move around and interact with one another in an unpredictable order to collectively carry out a computation. Toshimitsu Masuzawa gave a self-stabilizing link-colouring algorithm with the goal of fault containment: the distance that the effects of such errors can propagate through the network is bounded.

Fabiano Sarracco discussed malicious black holes, which can suck in mobile agents that wander too close. He described algorithms that can be used to search for black holes that approximate the optimal number of exploratory steps, and gave a lower bound on the achievable approximation ratio if local computation must be done in polynomial time (assuming $P \neq NP$).

Sanjoy Baruah and Michele Cirenei both presented talks on scheduling jobs which must be repeated periodically in a multiprocessor system. Björn Andersson also considered a periodic scheduling problem, but for message transmissions in a radio network, which has the constraint that transmissions must not collide.

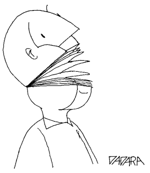
Vincent Gramoli talked about how to reconfigure read/write quora in a dynamic network. Ittai Abraham gave a new dictionary data structure, called a skip B-tree, that can be used in peer-to-peer systems. The data structure is the result of combining ideas from B-trees and from skip graphs. Alex Shvartsman discussed the node discovery problem: when the nodes in a network wish to cooperate on a problem, they must first discover one another's existence.

Héctor Tejeda gave some nice geometric constructions that can be used to build a graph structure (*e.g.* for routing) in an ad hoc network if each process on a plane knows its own coordinates. Mohamed Gouda proposed a model of sensor networks with the goal of being both realistic and mathematically tractable for verification. David Hay spoke about clock synchronization in wireless ad hoc networks. In particular, he discussed how to balance two conflicting goals: minimizing energy consumption and minimizing the difference between clocks. The final talk of the conference, by Kevin Lillis, described how mobile nodes could choose how much power to use while doing radio broadcasts: using too much is wasteful and causes unnecessary collisions, but using too little makes it hard to communicate. In particular, he described how to do this when there are bounded errors in available estimates of the distances between processes.

There were several social events at this well-organized conference. A reception (with live music) was held at Santa Croce in Fossabanda, a former convent dating from the 14th century. Conference attendees went on an excursion to the nearby walled city of Lucca for a walking tour of the historic sites. Afterwards, many of us relaxed in the Caffè di Simo which was the former haunt of Lucca's native son, Giacomo Puccini, and serves the richest hot chocolate I have ever tasted. The excursion was followed by an excellent dinner at Villa Poschi. On their own time, many attendees also explored the scenic city of Pisa, climbed the famous Leaning Tower, and enjoyed Tuscan cuisine.

The next OPODIS conference is to be held in Saint-Emilion, near Bordeaux, France from December 12 to 15, 2006. Details, including the call for papers are available at `www.opodis.net`.

Abstracts of PhD Theses



Abstract of PhD Thesis

Author:	Maria J. Blesa
Title:	Stability in Communication Networks under Adversarial Models
Language:	English
Supervisor:	Prof. Maria J. Serna
Institute:	Universitat Politècnica de Catalunya, Barcelona, Spain
Date:	February 27, 2006

Abstract

The work of A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson from 1996 [Universal stability results for greedy contention-resolution protocols, FOCS:380-389 1996, and JACM, 48(1):39-69 2001], together with the immediate continuation by the work of M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu [Adversarial queueing theory, STOC:376-385 1996, and JACM, 48(1):13-38 2001] can be considered the starting point and basis of the research conducted in this thesis. In their work, they proposed a model for studying the dynamics of packet-switched networks under adversarial traffic providers. This model was given the name of Adversarial Queueing Theory (AQT) and can be considered a pioneering work in studying stability via worst-case analysis.

Before AQT appeared, the existing models were based on more traditional queueing theory and used to make probabilistic assumptions on the traffic arrival pattern. The AQT model replaced those more traditional assumptions by worst-case inputs and approached the topic towards the traditional analysis of algorithms and to more unpredictable network configurations and dynamics.

The AQT model has been mainly applied to the study and determination of the conditions for assuring the stability in packet-switched communication networks. Stability is the property that determines whether the amount of information in the system is always bounded. Three main components are considered to define a networking system: the network (working in a packet-switched mode), the adversarial traffic provider (i.e., an adversary that injects traffic into the network), and the protocols used to schedule the congested packets in the intermediate points of the flow. Indirectly, stability is also referring to the limited size of the buffers at hosts for storing in-route packets and, in many cases, implies the on-time delivery of packets to their destination. A stronger notion studied is that of universal stability both from the network and the protocol point of view. The universal stability

of a network implies the stability of the system under every traffic provider and every protocol. The universal stability of a protocol implies the stability of the system under every network and traffic provider.

The aim of the research conducted in this thesis was to deepen the understanding of the conditions for stability in packet-switched communications networks, such as the Internet, under adversarial traffic service assumptions and to extend the model, as much as possible, in order to cover additional features of nowadays communication networks.

Not much results were available in the area when this thesis was started. The first contributions of this thesis use the AQT model as proposed in those original works. The property of universal stability of networks is fully characterised in terms of forbidden subgraphs. Different characterisations are provided according to different packet trajectory assumptions. The obtained characterisations allow to decide the property in polynomial time. Apart from the importance of the characterisation itself, these results helped on clarifying some confusion existing in the initial literature about the packet trajectory.

The property of universal stability of some protocols was also studied in this thesis. Interestingly enough, for some protocols the characterisation coincides with the characterisation of universal stability in networks. This, apart from assuring a polynomial time decidability, establishes a relation between both properties.

In order to study stability issues in more realistic scenarios, the original AQT model is extended in this thesis to capture additional features existing in nowadays networks. The new evolved models incorporate elements such as (i) prioritised flows, (ii) faulty network topologies and (iii) asynchronous networks dynamics.

Concerning prioritised flows, some extensions of the AQT model are proposed to deal with traffic in which the packets can have different priorities. Both, models for static and variable priorities, are considered. When the priorities of the packets can vary, the adversary become very powerful and the systems tend easily to instability under every greedy protocol. This should warn us about the possible consequences of the loss of control over the priority of the traffic in the network.

Inspired by the growing importance of wireless mobile networks, where some connections between nodes may fail or change quickly and unpredictably, adversarial models for dynamic networks, in which the edges of the underlying graph topology of the network can appear and disappear arbitrarily, are also proposed. In the dynamic models we propose, the adversary controls not only the packet arrivals, but also the link failures. Depending very much on which restrictions are imposed on the adversary in producing failures, and on how the packets involved in those failures are managed, the stability of the system is easier or harder to assure. This is a very influential matter in the study of stability.

The research included in this thesis concludes with a significantly more general adversarial model that includes specific technological features of nowadays networks, namely bandwidths, propagation delays and non-unitary packet sizes. These features determine strongly the dynamics of the communication networks we have nowadays. Additionally, the adversarial model proposed does not assume anymore the inherent synchronism of the AQT model. This is also an important feature to include in the forthcoming models, since the dynamics of real-life communication networks are not synchronous. However, the results considering this more general model do not differ significantly to those obtained with the original AQT model. This brings up the question of how complex a model has to be in order to capture the dynamics of real networking scenarios. Maybe, after all, a relatively simple model is enough to describe the complex dynamics of some communication networks.

Table of Contents (rough)

1 Introduction . . . 1
2 Preliminaries on graphs . . . 9
3 Adversarial Queueing Theory . . . 27
4 Universal stability of networks: directed graphs . . . 45
5 Universal stability of networks: undirected graphs . . . 73
6 Stability under a protocol . . . 89
7 Priority-based and faulty networks . . . 119
8 Failure management in faulty networks . . . 143
9 Continuous network dynamics . . . 157
10 Conclusions . . . 183
Bibliography . . . 193–201

Defense Jury

Prof. Josep Díaz,	Universitat Politècnica de Catalunya, Spain
Prof. Paul Spirakis,	University of Patras and CTI, Greece
Prof. Antonio Fernández,	Universidad Rey Juan Carlos, Spain
Dr. Francesc Comellas,	Universitat Politècnica de Catalunya, Spain
Dr. Zvi Lotker,	CWI, The Netherlands

Author's correspondence address Maria J. Blesa Aguilera
ALBCOM research group
Universitat Politècnica de Catalunya
Jordi Girona 1-3, Omega-213
E-08034 Barcelona, Spain
[mjblesa@lsi.upc.edu]

Abstract of PhD Thesis

Author:	Mohammad GhasemZadeh
Title:	A New Algorithm for the Quantified Satisfiability Problem, Based on Zero-suppressed Binary Decision Diagrams and Memoization.
Language:	English
Supervisor:	Prof. Christoph Meinel
Institute:	Hasso Plattner Institute, University of Potsdam, Germany.
Date:	30 November 2005

Abstract

SAT (SATisfiability of Boolean formula) and QSAT (SATisfiability of Quantified Boolean formula) are central problems in computer science. In this dissertation we present a new algorithm for the QSAT problem. Our algorithm is based on Zero-suppressed Binary Decision Diagrams (ZBDDs or ZDDs), splitting and memoization (a technique which is based on memorization). It is comparable and in some cases much faster than existing solutions.

A QBF is an expression of the form: $Q_1x_1 \ldots Q_nx_n\Phi \quad (n \geq 0)$, where every $Q_i(1 \leq i \leq n)$ is a quantifier, either existential $\exists$ or universal $\forall$; The literals $x_1 \ldots x_n$ are pairwise distinct atoms; and Φ is a propositional formula in CNF on atoms $x_1 \ldots x_n$. The QSAT problem is: Given a QBF formula, prove if is it True?

It can be shown that many scientific problems with numerous applications can be reduced or formulated in the form of the above problem. Therefore any progress in solving the QSAT problem, is a progress in solving a lot of important problems.

Almost all existing QSAT solvers are based on an extended version of the well known DPLL algorithm. The main efforts to improve the DPLL algorithm in recent solvers consist of:

- Clever selection of the branching/splitting variable;
- Clever selection of the next branch to examine;
- Utilizing efficient data structures;
- Making profit from local search and learning.

We realized that ZDDs can represent Boolean formulas given in CNF (Conjunctive Normal Form) efficiently. We also learned that the main drawback of the basic DPLL-for-QBF algorithm is its duplication. These observations led us to

use ZDD as the data structure holding the QBF formula and to embed memoization to the named algorithm in order to avoid solving the same subproblem repeatedly. We implemented our algorithm using CUDD (Colorado University Decision Diagram) package. We called it ZQSAT.

First experimental results showed that ZQSAT is comparable with other improved algorithms. Afterwards we developed and embedded a number of other heuristics to improve our ZQSAT. Finally we showed by experiential results and gave formal proofs to show how ZQSAT was comparable and sometimes much faster than all other existing solutions in solving a set of benchmark problems which are known to be hard for DPLL based QSAT solvers.

We also improved ZQSAT to let it accept QBFs in the form of $Q_1x_1 \ldots Q_nx_n\Phi$, (prenex-NNF) where Φ is a propositional formula in NNF(Negation Normal Form). We showed and proved that this possibility may be exponentially beneficial.

We may summarize our contribution as:

- Using ZDDs to represent the matrix of the QBF. We adopted this idea then established the specific rules suitable for QBF evaluation.
- Embedding memoization to overcome the duplication problem of the ”DPLL for QBF algorithm” (to avoid solving the same subproblem repeatedly).
- Accepting Prenex-NNF in addition to Prenex-CNF formulas.

We can summarize what we learned from ZQSAT as follows:

- Representation of Boolean formulas by ZDDs is beneficial, since this data structure allows compact representation of the formula and lets us store new (sub)formulas with no or a few additional nodes.
- Embedding memoization to the ’DPLL for QBF’ algorithm lets us avoid solving same subproblems repeatedly. Sometimes it was exponentially beneficial. This idea is only feasible along with the above idea.
- Removing subsumed clauses was very useful in simplification of the formula and as a result the speed of the search procedure.
- Transformation of a general Boolean formula into its equivalent NNF can be done efficiently, but transformation of a general Boolean formula into its equivalent in CNF can be done in exponential time. Therefore accepting NNF formulas can be exponentially beneficial.

Table of Contents

1 Introduction **12**

2 Boolean Formulas and Complexity **15**

3 Binary Decision Diagrams **24**

4 Zero-Suppressed Binary Decision Diagrams **40**

5 The QSAT Problem and Existing Solutions **57**

6 ZQSAT: A New Algorithm for the QSAT problem **65**

6.1 Introduction 65

6.1.1 The DPLL Algorithm for the SAT and QBFs 66

6.2 ZQSAT: A ZDD-based algorithm for deciding QBFs 70

6.2.1 Embedding Memoization to QDPLL 71

6.2.2 Using ZDD to represent a CNF formula 73

6.2.3 Benefits of using ZDDs along with our MQDPLL 76

6.2.4 Representing a Boolean formula in NNF by a ZDD 79

6.2.5 Accepting NNF formulas can be Exponentially Beneficial ... 81

6.3 Experimental Results 85

6.3.1 Evaluating ZQSAT over standard prenex-CNF benchmarks .. 86

6.3.2 Evaluating ZQSAT over instances in Prenex-NNF 90

7 Conclusions and future work **93**

REFERENCES **98**

Author's correspondence address Mohammad GhasemZadeh,
Dept. of Computer Engineering. ,
University of Yazd,
P.O.BOX 89195-741,
Yazd, IRAN.

Abstract of PhD Thesis

Author:	Hongmei He
Title:	Algorithms for the Book Crossing Number Problems
Language:	English
Supervisor:	Prof. Ondrej Sýkora and Dr. Ana Salagean
Institute:	Loughborough University , UK
Date:	28 April 2006

Abstract

The crossing minimisation problem is of outmost importance in the field of graph drawings. The aesthetical properties and readability of graphs are heavily dependent on the number of crossings in a drawing. Moreover, crossing minimisation is the most important goal in the linear Very Large Scale Integration (VLSI) circuit design and Quantum-dot Cellular Automata (QCA), since smaller crossing number means lower cost. Another application is in bioinformatics computing. The problem of computing an mRNA sequence of maximal codon-wise similarity to a given mRNA (MRSO problem) is fixed parameter tractable parameterised by the number of crossing edges of the implied structure graph of the source mRNA sequence.

The simplest graph drawing method is that of putting the vertices of a graph on a line and drawing the edges as half-circles on κ-half planes (pages). Such drawings are called κ-page book drawings. In a 1-page book draiwng, all edges are placed on one side of the spine, and in a 2-page book drawing, all edges are placed either above or below the spine. The minimal number of edge crossings in a book drawing is called the book crossing number (BCN). To minimise the BCN, the 1-page book crossing number problem (BCNP) is to find a good order of vertices, and the 2-page BCNP is to further find a good edge distribution in two pages. Both problems are NP-hard. Moreover, the 1-page and 2-page BCNs of a graph provide an upper bound for the standard planar crossing number of the graph. The main objectives were to develop various algorithms for the 1-page and 2-page BCNPs, and further to explore graph theory.

This thesis started with a survey of the previous research on the crossing number problem, especially on the 1-page and 2-page BCNPs. Based on the previous research, the following work on the 1-page and 2-page BCNPs has been done:

(1) presented novel heuristic algorithms to solve the BCNPs, which achieved the results better than or comparable with previous algorithms.

(2) applied genetic algorithms for the BCNPs. The genetic models obtained better results than the latest heuristic algorithms.

(3) created two neural network models for the 1-page and 2-page BCNPs, respectively. The convergence of the neural network models was investigated and good results are achieved. Especially, the model for the 2-page BCNP achieved much better performance in the quality of solutions and running time than previous model.

(4) investigated the complexity of parallel genetic algorithms, and unified the framework of PGA models with the function $PGA(subpopulation\ size, clu\text{–}ster\ size, migration\ period, \ topology)$.

(5) proved some theorems and presented some conjectures about the 1-page and 2-page crossing number for some structural graphs.

Table of Contents

1 Introduction **1**

1.1 Historical background 1
1.2 Motivation of the research 4
1.3 Objectives of the work 8
1.4 An overview of the thesis 9

2 Graph Theoretic Prerequisites **11**

2.1 Graphs 11
2.2 Representing graphs 27
2.3 The complexity of algorithms 29
2.4 NP-complete problems 32

3 Literature Review **35**

3.1 Embeddings and drawings 35
3.2 Extension of the crossing number problems 43
3.3 Previous research achievements 49
3.4 Previous algorithms used for the 1-side problem of bipartite graphs 65
3.5 Heuristic algorithms for the 1-page BCNP 68
3.6 Algorithms for the 2-page BCNP 74

4 Novel Heuristic Algorithms for the BCNPs **82**

4.1 Introduction 82
4.2 AVSDF+ algorithm 83
4.3 2-page crossing minimisation algorithms 97

5 Genetic algorithms for the BCNPs 108

5.1 Introduction 108
5.2 Overview of genetic algorithm 108
5.3 For the 1-page BCNP 111
5.4 For the 2-page BCNP 122

6 Neural Networks and Modeling the BCNPs 135

6.1 Introduction 135
6.2 An review of neural networks 136
6.3 A Hopfield model for the 1-page BCNP 151
6.4 An improved Hopfield model for the 2-page BCNP 161

7 Parallelisation of Genetic Algorithms 172

7.1 Introduction 172
7.2 Parallel Computing 173
7.3 Parallelisation of genetic algorithms 176
7.4 Chapter conclusions 195

8 Novel Exact Results and Conjectures 198

8.1 Introduction 198
8.2 The implementation of an exact algorithm for the 1-page BCNP 199
8.3 The exact 1-page BCNs and experimental data 199
8.4 The exact 2-page BCNs for some structural graphs 211

9 Conclusions 223

A Visualisation Test System 248

Author’s correspondence address Hongmei He
Department of Computer Science
Loughborough University
Loughborough
LE11 3TU
UK

Abstract of PhD Thesis

Author:	Matthias Mann
Title:	A Non-Deterministic Call-by-Need Lambda Calculus: Proving Similarity a Precongruence by an Extension of Howe's Method to Sharing
Language:	English
Supervisor:	Prof. Dr. Manfred Schmidt-Schauß
Institute:	FB Informatik und Mathematik, J.W.Goethe-Universität Frankfurt, Germany
Date:	17. August 2005

Abstract

In this dissertation a non-deterministic lambda-calculus with call-by-need evaluation is treated. Call-by-need means that subexpressions are evaluated at most once and only if their value must be known to compute the overall result. Also called "sharing", this technique is inevitable for an efficient implementation. In the calculus λ_{ND} of chapter 3 sharing is represented explicitely by a `let`-construct. Above, the calculus has function application, lambda abstractions, sequential evaluation and `pick` for non-deterministic choice.

Non-deterministic lambda calculi play a major role as a theoretical foundation for concurrent processes or side-effected I/O. In this work, non-determinism additionally makes visible when sharing is broken. Based on the bisimulation method a notion of equality is developed which respects sharing. Using bisimulation to establish contextual equivalence requires substitutivity within contexts, i.e., the ability to "replace equals by equals" within every program. This property is called congruence or precongruence if it applies to a preorder.

The open similarity of chapter 4 represents a new concept, insofar that the usual definition of a bisimulation is impossible in the λ_{ND}-calculus. So in section 3.2 a further calculus $\lambda_{\approx}$ has to be defined. In section 3.3 the so-called Approximation Theorem is proved which states that the evaluation in λ_{ND} and $\lambda_{\approx}$ agrees. The foundation for the non-trivial precongruence proof is set out in chapter 2 where the trailblazing method of Howe is extended to be capable with sharing. By the use of this (extended) method, the Precongruence Theorem proves open similarity to be a precongruence, involving the so-called precongruence candidate relation.

Joined with the Approximation Theorem we obtain the Main Theorem which says that the open similarity of the $\lambda_{\approx}$ is contained within the contextual preorder

of the λ_{ND}-calculus. However, this inclusion is strict, a property whose non-trivial proof involves the notion of syntactic continuity. Finally, chapter 6 discusses possible extensions of the base calculus such as recursive bindings or case and constructors. As a fundamental study the calculus lambda-ND provides neither of these concepts, since it was intentionally designed to keep the proofs as simple as possible. Section 6.1 illustrates that the addition case and constructors could be accomplished without big hurdles. Since recursive bindings cannot be represented simply by Y, further investigations are necessary.

Table of Contents

1 Introduction 1
- 1.1 Survey and Related Work 4
- 1.2 Outline 12

2 Lazy Computation Systems 17
- 2.1 Language 18
- 2.2 Preorders and the Precongruence Candidate 21
- 2.3 Reduction and Evaluation 26
- 2.4 Simulations 35
- 2.5 Future Work 42

3 Non-deterministic Lambda-calculi 45
- 3.1 The Call-by-need Calculus λ_{ND} 46
- 3.2 The Approximation Calculus $\lambda_{\approx}$ 57
- 3.3 Approximation of λ_{ND}-terms in $\lambda_{\approx}$ 82
- 3.4 Related Work 95

4 Similarity in $\lambda_{\approx}$ is a Precongruence 99
- 4.1 Similarity in the $\lambda_{\approx}$-Calculus 100
- 4.2 Admissibility of the Open Extension $(\cdot)^o$ 119
- 4.3 Proving Open Similarity a Precongruence 124
- 4.4 Future Work 132

5 Contextual and Denotational Semantics 137
- 5.1 Contextual (Pre-) Congruence 138
- 5.2 Syntactic Continuity 141
- 5.3 (In-) Equational Theory 147
- 5.4 Denotational Semantics 150

6 Possible Extensions of the Base Calculus 155
- 6.1 Lambda Calculi with `case` and Constructors 156
- 6.2 Lambda Calculi with recursive `let` 163

7 Conclusion and Future Work 167

Author's EMAIL address `mann@cs.uni-frankfurt.de`
URL: `http://www.ki.uni-frankfurt.de`

European

Association for

Theoretical

Computer

Science

EATCS

E A T C S

EATCS

HISTORY AND ORGANIZATION

EATCS is an international organization founded in 1972. Its aim is to facilitate the exchange of ideas and results among theoretical computer scientists as well as to stimulate cooperation between the theoretical and the practical community in computer science.

Its activities are coordinated by the Council of EATCS, which elects a President, Vice Presidents, a Treasurer and a Secretary. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual **I**nternational **C**olloquium on **A**utomata, **L**anguages and **P**rogramming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the "Bulletin of the EATCS;"
- Publication of the "EATCS Monographs" and "EATCS Texts;"
- Publication of the journal "Theoretical Computer Science."

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: TAPSOFT (Conference on Theory and Practice of Software Development), STACS (Symposium on Theoretical Aspects of Computer Science), MFCS (Mathematical Foundations of Computer Science), LICS (Logic in Computer Science), ESA (European Symposium on Algorithms), Conference on Structure in Complexity Theory, SPAA (Symposium on Parallel Algorithms and Architectures), Workshop on Graph Theoretic Concepts in Computer Science, International Conference on Application and Theory of Petri Nets, International Conference on Database Theory, Workshop on Graph Grammars and their Applications in Computer Science.

Benefits offered by EATCS include:
- Subscription to the "Bulletin of the EATCS;"
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from "EATCS Monographs" and "EATCS Texts;"
- Discount (about 70%) per individual annual subscription to "Theoretical Computer Science;"
- Discount (about 70%) per individual annual subscription to "Fundamenta Informaticae."

(1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July.

Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

Sites of ICALP meetings:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, Great Britain 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, Great Britain 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D.Theses;
- Entertainments and pictures related to computer science.

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at `http://www.eatcs.org/bulletin`. Questions and proposals can be addressed to the Editor by email at `bulletin@eatcs.org`.

(3) EATCS MONOGRAPHS AND TEXTS

This is a series of monographs published by Springer-Verlag and launched during ICALP 1984; more than 50 volumes appears. The series includes monographs in all areas of theoretical computer science, such as the areas considered for ICALPs. Books published in this series present original research or material of interest to the research community and graduate students. Each volume is normally a uniform monograph rather than a compendium of articles. The series also contains high-level presentations of special topics. Nevertheless, as research and teaching usually go hand in hand, these volumes may still be useful as textbooks, too. Texts published in this series are intended mostly for the graduate level. Typically, an undergraduate background in computer science is assumed. However, the background required may vary from topic to topic, and some books may be self-contained. The texts cover both modern and classical areas with an innovative approach that may give them additional value as monographs. Most books in this series will have examples and exercises. Updated information about the series can be obtained from the publisher.

The editors of the series are W. Brauer (Munich), G. Rozenberg (Leiden), and A. Salomaa (Turku). Potential authors should contact one of the editors. The advisory board consists of G. Ausiello (Rome), M. Broy (Munich), C.S. Calude (Auckland), A. Condon (Vancouver), D. Harel (Rehovot), J. Hartmanis (Cornell), T. Henzinger (Lausanne), N. Jones (Copenhagen), T. Leighton (MIT), M. Nivat (Paris), C. Papadimitriou (Athens and San Diego), and D. Scott (Pittsburgh).

EATCS Monographs and Texts is a very important EATCS activity and its success depends largely on our members. If you are a potential author or know one please contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:

Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands

who acknowledges EATCS membership and forwards the order to Springer-Verlag.

(4) THEORETICAL COMPUTER SCIENCE

The journal *Theoretical Computer Science*, founded in 1975, is published by Elsevier Science Publishers, Amsterdam, currently in 20 volumes (40 issues) a year. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

All kinds of papers, introducing or studying mathematical, logical and formal concepts and methods are welcome, provided that their motivation is clearly drawn from the field of computing.

Papers published in *TCS* are grouped in three sections according to their nature. One section, "Algorithms, automata, complexity and games," is devoted to the study of algorithms and their complexity using analytical, combinatorial or probabilistic methods. It includes the fields of abstract complexity (i.e., all the results about the hierarchies that can be defined using Turing machines), of automata and language theory (including automata on infinite words and infinitary languages), of geometrical (graphic) applications and of system performance using statistical models. A subsection is the Mathematical Games Section, which is devoted to the mathematical and computational analysis of games. The second section, "Logic, semantics and theory of programming," is devoted to formal methods to check properties of programs or implement formally described languages; it contains all papers dealing with semantics of sequential and parallel programming languages. All formal methods treating these problems are published in this section, including rewriting techniques, abstract data types, automatic theorem proving, calculi such as SCP or CCS, Petri nets, new logic calculi and developments in categorical methods. The newly introduced third section is devoted to theoretical aspects of "Natural Computing."

The Editors-in-Chief of "Theoretical Computer Science" are:

G. Ausiello, Università di Roma 'La Sapienza', Dip. Inform. e Sistemistica,
via Salaria 113, 00198 Roma, Italy;

D. Sannella, University of Edinburgh, Lab. for Foundations of Computer Science,
Division of Informatics, King's Building, Mayfield Road, Edinburgh, EH9 3JZ, UK

Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands

M.W. Mislove, Tulane University, Dept. of Mathematics, New Orleans, LA 70118, USA.

ADDITIONAL INFORMATION

For further information please visit `http://www.eatcs.org`, or contact the Secretary of EATCS:

Prof. Dr. Branislav Rovan, Department of Computer Science, Comenius University, SK-84248 Bratislava, Slovakia, Email: `secretary@eatcs.org`

EATCS MEMBERSHIP

DUES

The dues are €30 for a period of one year. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for €25 per year. Additional €25 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website `www.eatcs.org`, where you will find an online registration form and the possibility of secure online payment. Alternatively, a subscription form can be downloaded from `www.eatcs.org` to be filled and sent together with the annual dues (or a multiple thereof, if membership for multiple years is required) to the **Treasurer** of EATCS:

Prof. Dr. Dirk Janssens, University of Antwerp, Dept. of Math. and Computer Science
Middelheimlaan 1, B-2020 Antwerpen, Belgium
Email: `treasurer@eatcs.org`, *Tel: +32 3 2653904, Fax: +32 3 2653777*

The dues can be paid (in order of preference) by VISA or EUROCARD/MASTERCARD credit card, by cheques, or convertible currency cash. Transfers of larger amounts may be made via the following bank account. Please, add €5 per transfer to cover bank charges, and send the necessary information (reason for the payment, name and address) to the treasurer.

Fortis Bank, Bist 156, B-2610 Wilrijk, Belgium
Account number: 220–0596350–30–01130
IBAN code: BE 15 2200 5963 5030, SWIFT code: GEBABE BB 18A