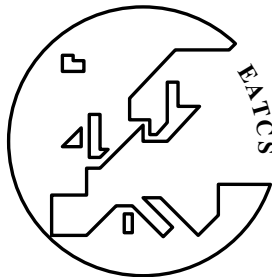


Bulletin

of the

European Association for Theoretical Computer Science

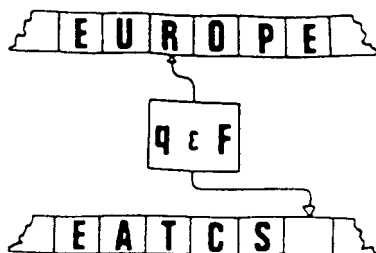
EATCS



Number 93

October 2007

COUNCIL OF THE EUROPEAN ASSOCIATION FOR THEORETICAL COMPUTER SCIENCE



PRESIDENT:	GIORGIO AUSIELLO	ITALY
VICE PRESIDENTS:	MOGENS NIELSEN	DENMARK
	PAUL SPIRAKIS	GREECE
TREASURER:	DIRK JANSSENS	BELGIUM
BULLETIN EDITOR:	VLADIMIRO SASSONE	UNITED KINGDOM

LUCA ACETO	ICELAND	MADHAVAN MUKUND	INDIA
KRZYSZTOF APT	THE NETHERLANDS	CATUSCIA PALAMIDESSI	FRANCE
PIERRE-LOUIS CURIEN	FRANCE	DAVID PELEG	ISRAEL
JOSEP DÍAZ	SPAIN	DON SANNELLA	UNITED KINGDOM
ZOLTÁN ÉSIK	HUNGARY	Jiří SGALL	CZECH REPUBLIC
FEDOR FOMIN	NORWAY	ANDRZEJ TARLECKI	POLAND
GIUSEPPE F. ITALIANO	ITALY	WOLFGANG THOMAS	GERMANY
JUHANI KARHUMÄKI	FINLAND	INGO WEGENER	GERMANY
RICHARD E. LADNER	USA	EMO WELZL	SWITZERLAND
JAN VAN LEEUWEN	THE NETHERLANDS	GERHARD WÖEGINGER	THE NETHERLANDS
EUGENIO MOGGI	ITALY		

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)
MOGENS NIELSEN	(2002–2006)		

EATCS COUNCIL MEMBERS

EMAIL ADDRESSES

Luca Aceto luca@ru.is
Giorgio Ausiello ausiello@dis.uniroma1.it
Krzysztof Apt apt@cw.nl
Pierre-Louis Curien curien@pps.jussieu.fr
Josep Díaz diaz@lsi.upc.es
Zoltán Ésik ze@inf.u-szeged.hu
Fedor Fomin fomin@ii.uib.no
Giuseppe F. Italiano italiano@disp.uniroma2.it
Dirk Janssens Dirk.Janssens@ua.ac.be
Juhani Karhumäki karhumak@cs.utu.fi
Richard E. Ladner ladner@cs.washington.edu
Jan van Leeuwen jan@cs.uu.nl
Eugenio Moggi moggi@disi.unige.it
Mogens Nielsen mn@brics.dk
Madhavan Mukund madhavan@cmi.ac.in
Catuscia Palamidessi catuscia@lix.polytechnique.fr
David Peleg peleg@wisdom.weizmann.ac.il
Don Sannella dts@dcs.ed.ac.uk
Vladimiro Sassone vs@ecs.soton.ac.uk
Jiří Sgall sgall@math.cas.cz
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 Woeginger g.j.woeginger@math.utwente.nl

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

The bulletin is entirely typeset by PDF_{TEX} and $\text{CON}_{\text{TEX}}\text{T}$ in TX_{FONTS} . 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 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ using the class `beatcs.cls` (a version of the standard $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}_{2_{\epsilon}}$ 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	5
REPORT FROM THE EATCS GENERAL ASSEMBLY	6
THE EATCS AWARD 2007	10
THE EATCS AWARD 2008	13
THE GÖDEL PRIZE 2008	14
NADIA BUSI (1968–2007)	18

EATCS NEWS

THE JAPANESE CHAPTER, <i>by K. Makino</i>	25
NEWS FROM INDIA, <i>by M. Mukund</i>	27
NEWS FROM LATIN AMERICA, <i>by A. Viola</i>	29
NEWS FROM NEW ZEALAND, <i>by C.S. Calude</i>	32

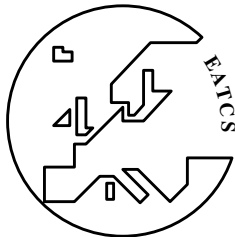
THE EATCS COLUMNS

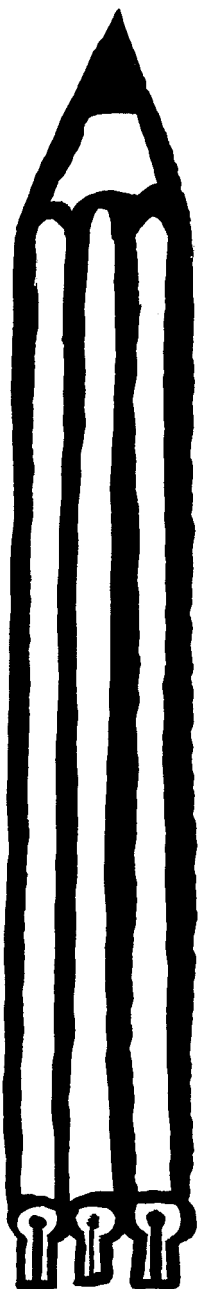
THE COMPLEXITY COLUMN, <i>by J. Torán</i>	
QUANTUM COMPUTING AND THE HUNT FOR HIDDEN SYMMETRY, <i>by</i> <i>G. Alagic and A. Russell</i>	53
THE CONCURRENCY COLUMN, <i>by L. Aceto</i>	
DYNAMIC WEB DATA AND PROCESS CALCULI, <i>by S. Maffei</i>	76
THE DISTRIBUTED COMPUTING COLUMN, <i>by M. Mavronicolas</i>	
AN INTRODUCTION TO POPULATION PROTOCOLS, <i>by J. Aspnes and E. Ruppert</i>	98
THE FORMAL LANGUAGE THEORY COLUMN, <i>by A. Salomaa</i>	
DECISION ALGORITHMS FOR SUBFAMILIES OF REGULAR LANGUAGES USING STATE-PAIR GRAPHS, <i>by Y.-S. Han</i>	118
THE FORMAL SPECIFICATION COLUMN, <i>by H. Ehrig</i>	
MODEL TRANSFORMATIONS BY GRAPH TRANSFORMATIONS ARE FUNCTORS, <i>by H. Ehrig, K. Ehrig, C. Ermel and U. Prange</i>	134
THE LOGICS IN COMPUTER SCIENCE COLUMN, <i>by Y. Gurevich</i>	
PROOF INTERPRETATIONS AND THE COMPUTATIONAL CONTENT OF PROOFS IN MATHEMATICS, <i>by U. Kohlenbach</i>	143
THE NATURAL COMPUTING COLUMN, <i>by G. Rozenberg</i>	
MACHINES OF SYSTEMS BIOLOGY, <i>by L. Cardelli</i>	176

TECHNICAL CONTRIBUTIONS

A SIMPLE COMPLETENESS PROOF FOR THE AXIOMATISATIONS OF WEAK BEHAVIOURAL EQUIVALENCES, <i>by Y. Deng</i>	207
THE DOMINO PROBLEM OF THE HYPERBOLIC PLANE IS UNDECIDABLE, <i>by</i> <i>M. Margenstern</i>	220
THE PUZZLE CORNER, <i>by L. Rosaz</i>	239
REPORTS FROM CONFERENCES	
ICALP 2007	243
AGT 2007	266
WG 2007	269
ABSTRACTS OF PHD THESES	271
EATCS LEAFLET	281

EATCS MATTERS



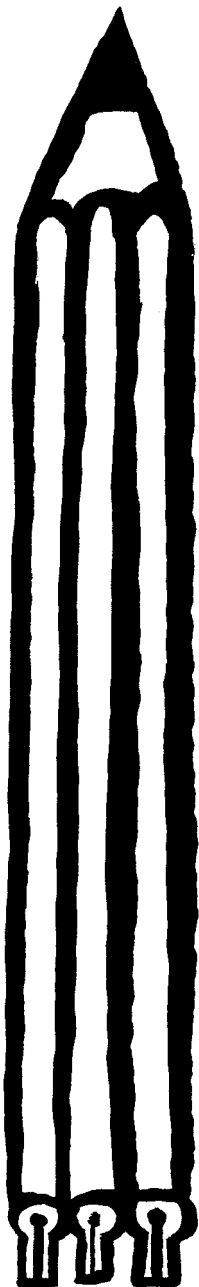


Letter from the President

Dear EATCS members,

first of all let me report you that the round of Council elections has just taken place and we are glad to give our greetings to the ten Council members that have been elected for the term 2007-2009. You may see the result of elections in this issue of the Bulletin. I would like to express my thanks to all the candidates and in particular to the six old members that have been reconfirmed and to the four members (Krzysztof Apt, Pierre-Louis Curien, Fedor Fomin, and Madhavan Mukund) that have been elected for the first time. We have several important issues to tackle and I am sure that the new members will bring new ideas and strong contributions. At the same time I would like to give a special thank to Grzegorz Rozenberg who is leaving our Council after having served for many years and after having given so many important contributions to the life of the Association, both as President, and as Bulletin Editor, Monograph Series Editor, and in uncountably many other ways. EATCS is deeply indebted with him and I hope EATCS can always count on his advice and support. I am sure you all share my feelings.

Another successful ICALP has been held this year. As you know, this year ICALP has been co-located with various important logic conferences (LICS, PPDP, and Logic Colloquium). Altogether ten Satellite Workshops have also taken place, giving rise to a memorable scientific event. I wish again to thank all the organizers in Wroclaw and in particular Jerzy



Marcinkowski and Thomas Jurdzinski for their excellent work.

During ICALP the Council and the General Assembly have taken place as usual. Several items have been addressed, regarding all aspects of the life of our Association: organization of future ICALP, publications, awards. Many decisions have been taken but several issues still need more in depth discussions. You can find the report on the General Assembly in this Bulletin.

In the meantime, in Reykjavik, Luca Aceto, Magnus Halldorson, Anna Ingolsfdottir are working hard for the organization of the 35th International Colloquium on Automata, Languages and Programming. Please look at the web site <http://www.ru.is/icalp08/> and be prepared to submit your best papers by February 10, 2008. Also the deadline for submitting Workshop proposals is still open. It will close on October 31, 2007.

Finally let me remind you that the Call for nominations for the Joint SIGACT-EATCS Goedel Prize is open. This year the Goedel Prize will be awarded during ICALP 2008 in Reykjavik. The Chairman of the Committee is Volker Diekert. Also the Call for Nomination for the EATCS Award is open. The Chairman of the Committee is David Peleg. Both Calls appear in this issue of the Bulletin (and of course in the EATCS web site).

Giorgio Ausiello, Roma
October 2007

Letter from the Bulletin Editor

Dear Reader,

Welcome to the October 2007 issue of the Bulletin of the EATCS. As usual for October issues, this volume contains reports from the annual EATCS gathering at ICALP. As you will know, the EATCS Distinguished Achievements Award has been presented to Dana Scott. Although we cannot report Dana's acceptance speech, we publish a transcript of the laudatio, prepared by the Award Committee on the basis of the nomination letters they received. The scientific reportage from ICALP is completed by Manfred Kudlek's report -as thorough as always- including the customary barrage of photographs, which you will find in the reports section. Regarding EATCS businesses, the President's report on the Annual General Assembly appears in the 'EATCS MATTERS' section.

Looking beyond reporting, the scientific contents of this issue is -I hope- as interesting as ever, presenting a blend of original contributions and some beautiful surveys. In particular, I would like to draw your attention on ULRIK KOHLENBACH's piece on the computational contents of mathematical proofs and on LUCA CARDELLI's paper on machines for systems biology.

In concluding, I would like to take a second to reflect on the much premature and shocking death of our colleague Nadia Busi. Roberto Gorrieri has prepared a beautiful IN MEMORIAM, which I refer you to.

Enjoy

Vladimiro Sassone, Southampton
October 2007



ICALP 2007

REPORT ON THE EATCS GENERAL ASSEMBLY 2007

The 2007 General Assembly (GA) of EATCS took place on Tuesday, July 10th, 2007, in Wroclav, at the Institute of Computer Science of the University, site of ICALP 2007. The President Giorgio Ausiello opened the GA at 17:00.

The agenda consisted of the following items.

REPORT OF THE EATCS PRESIDENT. The President reported briefly on the EATCS activities between ICALP 2006 and ICALP 2007. He referred to the more detailed report posted a couple of weeks before the GA on the EATCS web page at www.eatcs.org. Giorgio Ausiello explicitly mentioned and emphasized several items. First of all, some figures concerning the status of EATCS membership were given, followed by an overview of financial matters. Currently the EATCS members are 882, with a slight decrease with respect to the previous year. The President reminds all members to renew their membership as soon as it expires. The membership status can be checked through the EATCS web site. The financial situation of EATCS is stable. In the future some financial benefits are expected due to the decision to print and send paper copies of the Bulletin only to members explicitly asking for hardcopies. The money that will be saved in this way will be used for other initiatives that the Council is currently considering. During the period of the report 10 major scientific events took place under the auspices of EATCS. Besides EATCS has contributed to a variety of awards, namely, beside the EATCS Award 2007 that has been assigned to Professor Dana Scott for his fundamental contributions to the theory of computing, EATCS has contributed to assigning the Goedel Prize and Best Paper Awards at major conferences in theoretical computer science (ICALP, ETAPS, ESA). Finally, through an agreement between the ACM Symposium on Principles of Distributed Systems (PODC) and the EATCS Symposium on Distributed Computing (DISC), SIGACT and EATCS will also cooperate in awarding the Dijkstra Prize in Distributed Computing. The President reported on the composition of the award committees for 2008. For the EATCS Award Emo Welzl has been appointed to replace Wolfgang Thomas (whose three years term has expired). Emo Welzl will supplement Catuscia Palamidessi and David Peleg (who will chair the Committee). For the Goedel Prize Jean-Pierre Jouannaud has been appointed to replace Paul Vitanyi (whose three years term has expired). In 2008 the Committee will be chaired by Volker Diekert and the award ceremony will take place at ICALP. Subsequently the President reported on EATCS publications, again referring to the annual report for details. In the EATCS Texts and Monographs series, a total of three volumes

have been published in the last year. The Bulletin continues to be the flagship of EATCS publications. Posting it in the EATCS web pages for open access has been an important contribution of our association to the theoretical computer science community.

COUNCIL ELECTIONS. The President reminded that 2007 is an election year: ten members of the Council have finished their term and ten new members have to be elected. The members whose term has expired are: Josep Diaz, Zolt n Esik, Jean-Pierre Jouannaud, Juhani Karhum ki, David Peleg, Grzegorz Rozenberg, Branislav Rov n, Andrzej Tarlecki, Gerhard Woeginger, Uri Zwick. Subsequently the list of nominations received by the President for the future elections has been presented to the GA and integrated with further nominations. In conclusion the GA approved the following list of candidates: Noga Alon, Krzysztof Apt, Hagit Attiya, Yossi Azar, Gerth Brodal, Luis Caires, Pierre-Louis Curien, Xiaotie Deng, Josep Diaz, Zolt n Esik, Fedor Fomin, Pierre Fraigniaud, Juhani Karhum ki, Yoshiki Kinoshita, Rastislav Kralovic, Yassine Lakhnecht, Madhavan Mukund, David Peleg, Andrzej Tarlecki, Jiri Wiedermann, Gerhard Woeginger, Moti Yung. Before elections the President will check that all nominees indeed accept being candidates. The electronic ballot will be held during the month of September. Before closing this item in the Agenda the President reported that the Council has discussed about the possibility that in future Council election the candidates formulate a short statement that synthetically expresses the objectives they would like to pursue, if elected. The Council will investigate this issue further.

REPORT ON ICALP 2007. First of all Jerzy Marcinkowski (Institute of Computer Science, University of Wroclaw) presented a report on the organization of ICALP 2007 on behalf of the Organizing Committee. The scientific program was divided in three tracks: Track A – Algorithms, Automata, Complexity and Games (PC Chair Lars Arge, University of Aarhus, Denmark, Track B - Logic, Semantics, and Theory of Programming (PC Chair Andrzej Tarlecki, Warsaw University, Poland, Track C - Security and Cryptography Foundations (PC Chair Christian Cachin, IBM Zurich Research Laboratory, Switzerland. ICALP 2007 was co-located with LICS 2007, LC 2007, and PPDP 2007. The following satellite workshops were organized: Cryptography for Ad-hoc Networks (WCAN), Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA), Group-Oriented Cryptographic Protocols (GOCP), Structural Operational Semantics (SOS), Probabilistic Automata and Probabilistic Logics (PAuL), Theory of Randomized Search Heuristics (TRSH), Development of Computational Models (DCM), Logic and Computational Complexity (LCC), Traced Monoidal Categories, Network Algebras, and Applications (TM-

CNAA), Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS). The GA expressed its appreciation for the excellent organization of ICALP 2007, of ICALP Workshops and of all other co-located events. Subsequently the three ICALP PC Chairs, Lars Arge, Andrzej Tarlecki, and Christian Cachin presented their reports containing statistics of the three tracks. Overall 242 papers were submitted at ICALP 2007 (149, 59, 34 respectively for the three tracks); 76 were accepted. Details can be found in this issue of the Bulletin in the usual report contributed by Manfred Kudlek. Again the GA expressed its appreciation for the excellent work done by the Program Committees. Giorgio Ausiello continued the tradition to present the ICALP organizers Jerzy Marcinkowski and Thomas Jurdzinski with small gifts, thanking them for their efforts.

REPORT ON ICALP 2008. M. Mousavi, Workshop Chair of ICALP 2008 reported on the organization of ICALP 2008 in Reykjavik, Iceland, on behalf of himself and of the team of organizers (consisting of Luca Aceto, Magnus Halldorsen, Anna Ingolsfottir). ICALP 2008 will be organized in three tracks. The traditional tracks A and B will be respectively chaired by Leslie Ann Goldberg and by Igor Walukiewicz. Track C will be again devoted to Security and Cryptography Foundations and will be chaired by Ivan Damgaard. After presenting logistic and organizational aspects of ICALP 2008 Mousavi reminded that the Call for Papers was ready and had been posted in the web pages of the Conference. Paper copies were available at ICALP 2007 for all interested participants. The GA thanked Mousavi and the whole group from Reykjavik for their organizational efforts.

VENUE OF ICALP 2009. Giorgio Ausiello announced that for the organization of ICALP 2009 two bids had been formulated, the first one by Catuscia Palamidessi and Jean-Pierre Jouannaud (Ecole Polytechnique, Paris), the second one by Paul Spirakis (CTI, Patras) and Elias Koutsoupias (U. of Athens). Subsequently the bid from Paris has been postponed to 2010 and therefore, at the time of GA, the only valid proposal for 2009 was the one put forward by Patras and Athens. No other proposal was brought up by those present at GA. On invitation from the President, Paul Spirakis illustrated the proposal. At the end the President thanked Spirakis for his presentation and asked the Assembly to vote. Although the precise location was not decided yet (Rhodes being one option) the GA unanimously approved that in 2009 ICALP will be organized in Greece, jointly by CTI and University of Athens. Before concluding this item in the Agenda, the President started a brief discussion on future ICALP conferences. First of all, in view of the limited number of papers submitted to Track C in 2007 Giorgio Ausiello reported that the Council is considering the opportunity to identify new topics for Track C, possibly starting with ICALP 2009. In second place, Giorgio Ausiello

reminded that both STOC and FOCS, the two main US conferences in theoretical computer science, have been held once in Europe and suggested to explore the possibility to bring ICALP once to North America in the next few years, possibly in connection with the opportunity to co-locate ICALP with LICS. Also this issue will be investigated further by the Council.

EU RESEARCH MATTERS. The President invited Paul Spirakis to report about recent developments in EU research and about the initiatives taken to improve EATCS visibility toward the EU Commission. At the end of the report the President thanked Paul Spirakis for his presentation. The President announced that the slides of Spirakis's report would have been posted in the "EU Matters" Section of EATCS web site and reminded to the Assembly that such new section of the EATCS web site is devoted to reports and news regarding EU research and may be entered only by EATCS members.

SPECIALS. At this point Giorgio Ausiello gave the floor to Manfred Kudlek who illustrated the statistics of the authors who published repeatedly at ICALP and presented the special EATCS badge to those having reached 5 or more full papers at ICALP. By tradition Kudlek also presented the EATCS badges to the Editors of ICALP 2007 Proceedings.

At 18:30, since no other matter had to be addressed, the President thanked all present and closed the 2007 General Assembly of EATCS.

Giorgio Ausiello

THE EATCS AWARD 2007

LAUDATIO FOR PROFESSOR DANA SCOTT

AS READ BY THE CHAIR OF THE COMMITTEE ON 10TH JULY 2007 IN WROCLAW

Dear Professor Scott, dear Dana, dear Mrs. Scott,

dear colleagues and friends,

as stated in the announcements of EATCS, the EATCS Award is awarded annually, to honor scientists for extensive and widely recognised contributions to theoretical computer science over a life long scientific career.

After a careful consideration of the nominations for 2007, the award committee and the council of EATCS have decided in unanimous votes to honor

PROFESSOR DANA SCOTT

with the EATCS award 2007, in recognition of his deep and influential achievements and his lasting merits to our field. And — as the president of EATCS mentioned in one of the letters exchanged on the matter — it is a privilege for EATCS to grant its award to this outstanding scientist.

As chairman of the 2007 award committee, I will try to say some very short words about Dana's work, which is not so easy due to the enormous spectrum of subjects to which he has contributed, often far beyond theoretical computer science. In fact, the first steps of his career were in mathematical logic, with his PhD thesis work in Princeton on "Convergent Sequences of Complete Theories" under the supervision of Alonzo Church.

Soon after his PhD in 1958 we see his first contribution to computer science, when he wrote a celebrated paper with Michael Rabin, a colleague from Princeton, entitled "Finite Automata and Their Decision Problems." This paper introduced the idea of nondeterministic machines to automata theory. It belongs to the fundamentals of computation theory as taught today to every computer science student

— a pioneering work which earned Rabin and Scott also the 1976 ACM Turing Award.

Dana Scott took up his first posts as instructor and professor of mathematics and logic at Chicago, the University of California, Berkeley, and Stanford. At that time he worked on a great variety of problems in set theory and model theory, producing for instance a new and very elegant proof of the independence of the continuum hypothesis. After some time again in Princeton he finally came to Europe in 1972, when he took up the post of Professor of Mathematical Logic at the University of Oxford which he held for nine years. By coincidence or not — I do not know — this move to Europe came with a breakthrough he achieved in theoretical computer science.

In Oxford, Dana developed in cooperation with Christopher Strachey a mathematical foundation to the denotational semantics of programming languages, also known as the Scott-Strachey approach to semantics. This work constitutes one of the most influential pieces of work in theoretical computer science, and can be regarded as founding one of the major schools of computer science. Dana's major contribution is his development of domain theory that allows programs involving recursive functions and complex control constructs to be given a denotational semantics. Additionally he provided a foundation for the understanding of infinitary and continuous information through domain theory. The impact of this work was enormous: for instance, Milner developed the Stanford LCF based on a logic developed by Scott.

Let me add a remark from a more historical perspective. In the 1930's, the work of mathematicians such as Gödel, Turing, Tarski, and Church had profound consequences on Mathematical Logic and gave origin to Computability Theory. Dana Scott is among the top researchers who carried that mathematical tradition into Computer Science. Denotational semantics is the natural evolution of Tarskian (compositional) semantics for logical languages into the realm of programming languages, and Domain Theory provides the needed semantic objects. In particular, effectively-given domains (and Information Systems) extend the notion of computability well beyond natural numbers and strings.

In 1981, Dana went to Carnegie Mellon University to become Hillman Professor of Computer Science, Mathematical Logic, and Philosophy. His more recent work has further advanced the foundations of denotational semantics in two directions: Synthetic Domain Theory (a way to view domains as sets in a non-standard set theory) and Equilogical Spaces, which represent a natural extension of domains.

Dana Scott has obtained further fundamental results also in Set Theory, Model Theory, Modal Logic, Topology, Category Theory, and Realizability, a range much too vast to be discussed here. Let me mention only one aspect: His work on

constructive mathematics, and also his lucid expositions of the work of others, had a great influence on the uses non-classical logics in computing — for example, in constructive reasoning, modal logics for concurrency, and propositions-as-types. He made a whole generation see the potential of non-standard logics, an idea as revolutionary as the one of non-standard geometries had been.

His brilliant and masterful style of writing is a standard that is guiding — or should be guiding — researchers in computer science and mathematics.

In summary, Dana Scott's research career has been characterized by a concern for elucidating fundamental concepts, with a focus on mathematically hard problems that bear on these concepts. His influence in forming mathematical foundations for Computer Science is now visible over a period of over 40 years. The European research community of Theoretical Computer Science owes a lot to his thoughts and is flourishing while pursuing the directions he started.

THE EATCS AWARD 2008

CALL FOR NOMINATIONS

EATCS annually honors a respected scientist from our community with the prestigious **EATCS DISTINGUISHED ACHIEVEMENTS AWARD**. The award is given to acknowledge extensive and widely recognized contributions to theoretical computer science over a life long scientific career.

For the EATCS Award 2008, candidates may be nominated to the Awards Committee. Nominations must include supporting justification and will be kept strictly confidential. The deadline for nominations is: **December 1, 2007**.

Nominations and supporting data should be sent to the chairman of the EATCS Awards Committee:

Professor David Peleg
Department of Computer Science and Applied Mathematics
The Weizmann Institute of Science
Rehovot 76100
Israel
Email: david.peleg@weizmann.ac.il

Previous recipients of the EATCS Award are

R.M. Karp	(2000)	C. Böhm	(2001)
M. Nivat	(2002)	G. Rozenberg	(2003)
A. Salomaa	(2004)	R. Milner	(2005)
M. Paterson	(2006)	D.S. Scott	(2007)

The next award is to be presented during ICALP'2008 in Reykjavik.

GÖDEL PRIZE 2008

CALL FOR NOMINATIONS

deadline: **JANUARY 31, 2008.**

The Gödel Prize for outstanding papers in the area of theoretical computer science is sponsored jointly by the European Association for Theoretical Computer Science (EATCS) and the Association for Computing Machinery Special Interest Group on Algorithms and Computation Theory (ACM-SIGACT). This award is presented annually, with the presentation taking place alternately at the International Colloquium on Automata, Languages, and Programming (ICALP) and the ACM Symposium on Theory of Computing (STOC). The sixteenth presentation will take place during ICALP 2008, Reykjavik, Iceland, July 6 to 13, 2008. The Prize is named in honor of Kurt Gödel in recognition of his major contributions to mathematical logic and of his interest, discovered in a letter he wrote to John von Neumann shortly before Neumann's death, in what has become the famous "P versus NP" question. The Prize includes an award of \$ 5.000 (USD).

AWARD COMMITTEE: The winner of the Prize is selected by a committee of six members. The EATCS President and the SIGACT Chair each appoint three members to the committee, to serve staggered three-year terms. The committee is chaired alternately by representatives of EATCS and SIGACT, with the 2008 Chair (Volker Diekert) being a EATCS representative.

The 2008 Award Committee consists of Volker Diekert (Universität Stuttgart), Shafi Goldwasser (MIT and Weizmann Institute), Johan Håstad (KTH Stockholm), Jean-Pierre Jouannaud (École Polytechnique and Université Paris-Sud), Christos Papadimitriou (UC Berkeley), and Colin Stirling (University of Edinburgh).

ELIGIBILITY: (The last change of rules goes back to the 2005 Prize.) Any research paper or series of papers by a single author or by a team of authors is deemed eligible if the paper was published in a recognized refereed journal before nomination but the main results were not published (in either preliminary or final form) in a journal or conference proceedings before 1994. Hence, if *JP*

(respectively *CP*) is the journal publication date (respectively the conference proceedings date) of a nominated paper, and if n denotes the year of the next Award (here $n = 2008$) then the following constraints should be respected:

$$JP \leq (\text{January } 31, n) \quad \text{and} \quad \min(JP, CP) \geq (\text{January } 1, (n - 13))$$

Here, choosing $n - 13$ is meant as a recognition of the fact that the value of fundamental work cannot always be immediately assessed, and *CP* is taken into account because a conference publication often is the most effective means of bringing new results to the attention of the community.

The research work nominated for the award should be in the area of theoretical computer science. The term “theoretical computer science” is meant to encompass, but is not restricted to, those areas covered by ICALP and STOC. Nominations are encouraged from the broadest spectrum of the theoretical computer science community so as to ensure that potential award-winning papers are not overlooked. The Award Committee shall have the ultimate authority to decide whether a particular paper is eligible for the Prize.

NOMINATIONS: Nominations for the award should be submitted to the Award Committee Chair at the following address:

Volker Diekert
Institute of Formal Methods in Computer Science
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany
email: diekert@informatik.uni-stuttgart.de
tel: ++49 711 7816329
fax: ++49 711 7816310

To be considered, nominations for the 2008 prize must be received by January 31, 2008. Nominations may be made by any member of the scientific community. A nomination should contain a brief summary of the technical content of the paper(s) and a brief explanation of its significance. A copy of the research paper or papers should accompany the nomination. The nomination must state the date and venue of the first conference publication or state that no such publication has occurred. If at all possible, we request that the nomination letter and nominated paper(s) be transmitted via email.

The work may be in any language. However, if it is not in English, a more extended summary written in English should be enclosed. Additional recommendations in favor of the nominated work may also be enclosed. To be considered for the award, the paper or series of papers must be recommended by at least

two individuals, either in the form of two distinct nominations or one nomination including recommendations from two different people.

Those intending to submit a nomination are encouraged to contact the Award Committee Chair by email well in advance. The “Subject” line of all related messages should begin with “Goedel08.”

SELECTION PROCESS: Although the Award Committee is encouraged to consult with the theoretical computer science community at large, the Award Committee is solely responsible for the selection of the winner of the award. The prize may be shared by more than one paper or series of papers, and the Award Committee reserves the right to declare no winner at all. All matters relating to the selection process that are not specified here are left to the discretion of the Award Committee.

PAST WINNERS:

2007: ALEXANDER A. RAZBOROV AND STEVEN RUDICH, “Natural Proofs,” *Journal of Computer and System Sciences*, **55** (1997), 24–35.

2006: MANINDRA AGRAWAL, NEERAJ KAYAL, AND NITIN SAXENA, “PRIMES is in P,” *Annals of Mathematics*, **160** (2004), 1–13.

2005: NOGA ALON, YOSHI MATIAS AND MARIO SZEGEDY, “The space complexity of approximating the frequency moments,” *Journal of Computer and System Sciences*, **58** (1999), 137–147.

2004: MAURICE HERLIHY AND NIR SHAVIT, “The Topological Structure of Asynchronous Computation,” *Journal of the ACM*, **46** (1999), 858–923.

MICHAEL SAKS AND FOTIOS ZAHAROGLOU, “Wait-Free k -Set Agreement Is Impossible: The Topology of Public Knowledge,” *SIAM Journal of Computing*, **29** (2000), 1449–1483.

2003: YOAV FREUND AND ROBERT SCHAPIRE, “A Decision Theoretic Generalization of On-Line Learning and an Application to Boosting,” *Journal of Computer and System Sciences* **55** (1997), 119–139.

2002: GÉRAUD SÉNIZERGUES, “ $L(A)=L(B)$? Decidability results from complete formal systems,” *Theoretical Computer Science* **251** (2001), 1–166.

2001: URIEL FEIGE, SHAFI GOLDWASSER, LÁSZLÓ LOVÁSZ, SHMUEL SAFRA, AND MARIO SZEGEDY, “Interactive proofs and the hardness of approximating cliques,” *Journal of the ACM* **43** (1996), 268–292.

SANJEEV ARORA AND SHMUEL SAFRA, “Probabilistic checking of proofs: a new characterization of NP,” *Journal of the ACM* **45** (1998), 70–122.

SANJEEV ARORA, CARSTEN LUND, RAJEEV MOTWANI, MADHU SUDAN, AND MARIO SZEGEDY, “Proof verification and the hardness of approximation problems,” *Journal of the ACM* **45** (1998), 501–555.

- 2000:** MOSHE Y. VARDI AND PIERRE WOLPER, “Reasoning about infinite computations,” *Information and Computation* **115** (1994), 1–37.
- 1999:** PETER W. SHOR, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Journal on Computing* **26** (1997), 1484–1509.
- 1998:** SEINOSUKE TODA, “PP is as hard as the polynomial-time hierarchy,” *SIAM Journal on Computing* **20** (1991), 865–877.
- 1997:** JOSEPH HALPERN AND YORAM MOSES, “Knowledge and common knowledge in a distributed environment,” *Journal of the ACM* **37** (1990), 549–587.
- 1996:** ALISTAIR SINCLAIR AND MARK JERRUM, “Approximate counting uniform generation and rapidly mixing Markov chains,” *Information and Computation* **82** (1989), 93–133.
- MARK JERRUM AND ALISTAIR SINCLAIR, “Approximating the permanent,” *SIAM Journal on Computing* **18** (1989), 1149–1178.
- 1995:** NEIL IMMERMANN, “Nondeterministic space is closed under complementation,” *SIAM Journal on Computing* **17** (1988), 935–938.
- RÓBERT SZELEPCSÉNYI, “The method of forced enumeration for nondeterministic automata,” *Acta Informatica* **26** (1988), 279–284.
- 1994:** JOHAN HÅSTAD, “Almost optimal lower bounds for small depth circuits,” *Advances in Computing Research* **5** (1989), 143–170.
- 1993:** LÁSZLÓ BABAI AND SHLOMO MORAN, “Arthur-Merlin games: a randomized proof system and a hierarchy of complexity classes,” *Journal of Computer and System Sciences* **36** (1988), 254–276.
- SHAFI GOLDWASSER, SILVIO MICALI AND CHARLES RACKOFF, “The knowledge complexity of interactive proof systems,” *SIAM Journal on Computing* **18** (1989), 186–208.

OBITUARY



NADIA BUSI **(1968 – 2007)**

Nadia Busi passed away, unexpectedly, on September 5, after a brief illness, at the age of 39. She leaves a 5-year old son, Luca, and many friends and colleagues in deep sadness.

Nadia studied first in Bologna (Master in Computer Science in 1993), then in Siena (PhD in Logic and Theoretical Computer Science in 1997). In 1998 she received the annual EATCS (Italian Chapter) prize for the best Italian PhD thesis for her doctoral dissertation “Petri Nets with Inhibitor and Read Arcs: Semantics, Analysis and Application to Process Calculi”. In 1997 she was appointed as an assistant professor at the University of Bologna, and in 2001 she became an

associate professor at the same university.

It is usually difficult to draw a picture of the scientific contribution of a person, especially when the person is so young. This is even more true for me since, as her PhD thesis advisor, I run the risk of not being objective in setting her contributions in the right light. Nadia's main interest was the study of expressiveness problems in concurrency theory, also developing novel proof techniques to do this. She started in her PhD thesis by singling out a particular subclass of Petri nets with inhibitor arcs (a Turing-complete formalism), called primitive nets, for which some interesting properties are decidable. She showed that it was possible to map an interesting fragment of the pi-calculus on primitive nets, hence proving that such a fragment enjoys the same decidability results of primitive nets. In her thesis, in joint work with Gianluigi Zavattaro (who co-authored most of her papers on expressiveness), she also provided a proof that a core process calculus, built around the coordination primitives of Linda, is Turing-complete or not depending on the actual definition of the output operation (synchronous vs asynchronous). Starting from that basic experiment, she then studied several calculi, many of which inspired by coordination languages (with shared data space, with publish/subscribe mechanisms, or with event-driven mechanisms), for which she offered, in a series of papers, a thorough study of their relative expressive power. A common theme in these studies was the use of Petri nets (or well-structured transition systems) semantics for such calculi as a medium to prove different decidability and undecidability results. It is indeed a pity that such a long series of papers was not complemented with a comprehensive overview for easy access to the interested scholar. Nadia also developed deep foundational work in studying the relative expressive power of recursion, replication and iteration in basic process calculi, as well as some work on the expressiveness of fragments of the ambient calculus. In joint work with Michele Pinna, she also studied some aspects of Petri nets with read and inhibitor arcs, e.g., providing this class of nets with a truly concurrent semantics. Some of her secondary interests included service-oriented computing, security and stochastic Petri nets.

More recently, she was quite fascinated with bio-inspired models of computation, both coming from process calculi (such as Cardelli's brane calculi) or from automata theory (such as Paun's P systems). She developed new classes of models, such as Genetic P systems (in joint work with Claudio Zandron), as well as foundational studies on decidability properties of other well-known formalisms (e.g., brane calculi). This line of research was the one that most excited Nadia's interest in the last year of her life. She talked to me about her deep interest in such problems, and I was astonished by her ability to connect different aspects of different formalisms in terms of their relative expressive power. But, all of a sudden, silence: an irreparable loss.

Roberto Gorrieri



INSTITUTIONAL SPONSORS

BRICS, Basic Research in Computer Science,
Aarhus, Denmark

Elsevier
Amsterdam, The Netherlands

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

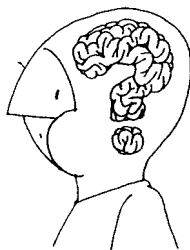
Microsoft Research,
Cambridge, United Kingdom

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



REPORT FROM THE JAPANESE CHAPTER

K. Makino (Tokyo Univ.)

EATCS-JP/LA Workshop on TCS

The *seventh EATCS/LA Workshop on Theoretical Computer Science* will be held at Research Institute of Mathematical Sciences, Kyoto Univ., January 28 ~ 30, 2008. The workshop will be jointly organized with *LA*, Japanese association of theoretical computer scientists. Its purpose is to give a place for discussing topics on all aspects of theoretical computer science.

A formal call for papers will be announced at our web page early November, and a program will be announce early January, where we are also planning to announce a program in the next issue of the Bulletin. Please check our web page around from time to time. If you happen to stay in Japan around that period, it is worth attending. No registration is necessary for just listening to the talks; you can freely come into the conference room. (Contact us by the end of November if you are considering to present a paper.) Please visit Kyoto in its most beautiful time of the year !

6th EATCS-JP/LA Presentation Award

The sixth EATCS/LA Workshop on Theoretical Computer Science was held at Research Institute of Mathematical Sciences, Kyoto Univ., January 29 ~ 31, 2007. **Mr. Keita Xagawa** (Tokyo Inst. of Tech.) who presented the following paper, was selected as the 6th EATCS/LA Presentation Award.

A lattice-based cryptosystem and proof of knowldge on its secret key
by K. Xagawa, A. Kawachi, K. Tanaka (Tokyo Inst. of Tech.)

The award was given to him at the Summer LA Symposium held in August 2007. *Congratulations!* Please check our web page for the detail information and the list of presented papers.

On TCS Related Activities in Japan:

TGCOMP Meetings, January ~ June, 2007

The *IEICE*, Institute for Electronics, Information and Communication Engineers of Japan, has a technical committee called *TGCOMP*, Technical Group on foundation of COMPuting. During January ~ June of 2007, *TGCOMP* organized 4 meetings and 43 papers (including two tutorials) were presented there. Topics presented are, very roughly, classified as follows.

Algorithm: On Graphs (12)	Computational Complexity (2)
Algorithm: On Strings (6)	Cryptography (1)
Algorithm: On Other Objects (11)	Distributed Computing (5)
Combinatorics / Probabilistic Analysis (1)	Formal Languages and Automata (5)

See our web page for the list of presented papers (title, authors, key words, email).

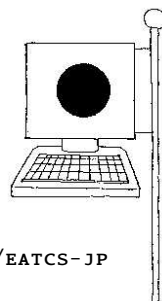
Forthcoming Events in Japan:

ISAAC 2007 The 18th International Symposium on Algorithms and Computation (ISAAC 2007) will be held in Sendai Excel Hotel Tokyu, Sendai, Japan, December 17-19, 2007. The symposium is intended to provide a forum for researchers working in algorithms and theory of computation. The invited speakers for ISAAC 2007 are Pankaj K. Agarwal (Duke University) and Robin Thomas (Georgia Institute of Technology). The list of accepted papers is available via the conference website <http://www.nishizeki.ecei.tohoku.ac.jp/isaac07/>.

IWAAG 2007 The International Workshop on Algorithms and Graphs (IWAAG 2007) will be held in Sendai Excel Hotel Tokyu, Sendai, Japan on December 16, 2007. This workshop honors Professor Takao Nishizeki on the occasion of his 60th birthday. The list of the invited speakers for ISAAC 2007 includes Francis Y. L. Chin (U. Hong Kong), Ding-Zhu Du (U. Texas), Peter Eades (U. Sydney), Seokhee Hong (NICTA), Wen-Lian Hsu (Academia Sinica), Der Tsai Lee (Academia Sinica), Md. Saidur Rahman (BUET), Dorothea Wagner (U. Karlsruhe), and Hsu-Chun Yen (National Taiwan U.). See also the website <http://www.nishizeki.ecei.tohoku.ac.jp/iwaag07/>.

THE JAPANESE CHAPTER

CHAIR:	KAZUO IWAMA
V. CHAIR:	OSAMU WATANABE
SECRETARY:	KAZUHISA MAKINO
EMAIL:	EATCS-JP@IS.TITECH.AC.JP
URL:	HTTP://WWW.IS.TITECH.AC.JP/~WATANABE/EATCS-JP



NEWS FROM INDIA

BY

MADHAVAN MUKUND

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

Winter is approaching and, as usual, this is the busy season for conferences for India, so look ahead to some of the conferences and workshops coming up in late 2007–early 2008.

FSTTCS 2007 The 27th edition of FSTTCS, the annual conference of the Indian Association for Research in Computing Science (IARCS) will take place at the India International Centre in New Delhi during December 12–14, 2007. Two one day workshops are planned in conjunction with the conference, on December 11 and 15.

The Program Committee is chaired by V. Arvind (IMSc, Chennai) and Sanjiva Prasad (IIT Delhi). The invited speakers for FSTTCS 2007 are Maurice Herlihy, Richard Karp, Benjamin Pierce, Thomas Reps, Salil Vadhan and Andrew Yao. The list of accepted papers is available at the conference website, <http://www.fsttcs.org>.

Indocrypt 2007 The 8th International Conference on Cryptology in India, Indocrypt 2007, will be held in Chennai during December 10–12, 2007. The Program Committee is chaired by C. Pandu Rangan (IIT Madras), Moti Yung (Columbia, USA) and Kannan Srinathan (IIIT, Hyderabad). The invited speakers are S.V. Raghavan (IIT Madras), Ramarathnam Venkatesan (Microsoft Research) and Jonathan Katz (Maryland). There will be a preconference tutorial on *Side Channel Attacks* by Ingrid Verbauwhede (Katholieke Universiteit Leuven, Belgium) and two postconference tutorials on *Theoretical Foundations of Public-Key Encryption* and *Robust Combiners* by Manoj M. Prabhakaran (UIUC, USA) and Krzysztof Pietrzak (CWI, Amsterdam, the Netherlands), respectively.

The list of accepted papers and other details are available at the conference website, <http://www.cs.iitm.ernet.in/~indocrypt2007>.

QIP 2008 The Eleventh Workshop on Quantum Information Processing, QIP 2008, is being organized in New Delhi during December 17–21, 2007 by IARCS, with support from the Indian Institute of Technology, New Delhi, and the Tata Institute of Fundamental Research, Mumbai. The QIP workshop series began in Aarhus in 1998 and is the premier annual meeting in the field. QIP 2008, like its previous editions, will feature invited talks, contributed talks and a poster session. The workshop website is at <http://qipworkshop.org>.

TECS Week 2008 The Tata Research Development and Design Centre (TRDDC), Pune, India continues its annual TCS Excellence in Computer Science (TECS) Week series of instructional workshops with TECS Week 2008, to be held at TRDDC during January 7-11 2008. TECS Week is jointly conducted by TRDDC, the International Institute for Software Technology (IIST), United Nations University and IARCS.

This year's talks are on the topics *Human Computer Interaction*, *Virtual Reality*, and *Data Visualization*. The speakers for TECS Week 2008 are Stephen Ellis (NASA AMES, USA), Tamara Munzner (U British Columbia, Canada), Sethuraman Panchanathan (Arizona State, USA), Harold Thimbleby (Swansea, UK) and Kentaro Toyama (Microsoft Research, India. For more details, see http://www.tcs-trddc.com/Tecs'08/tcs_excellence_in_computer_scienc.htm

Madhavan Mukund, Chennai Mathematical Institute
Secretary, IARCS (Indian Association for Research in Computing Science)
<http://www.cmi.ac.in/~madhavan>

NEWS FROM LATIN AMERICA

BY

ALFREDO VIOLA



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

In this issue I present SISAP 2008, AMW 2007, LATIN 2008, and the invitation for participation in SPIRE 2007. 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.

SISAP 2008

The International Workshop on Similarity Search and Applications (SISAP) is a new conference devoted to similarity searching, with emphasis on metric space searching, and will be held on April 11 - 12, 2008 in Cancún, Mexico. It aims to fill in the gap left by the various scientific venues devoted to similarity searching in spaces with coordinates, by providing a common forum for theoreticians and practitioners around the problem of similarity searching in general spaces (metric and non-metric) or using distance-based (as opposed to coordinate-based) techniques in general. SISAP aims to become an ideal forum to exchange real-world, challenging and exciting examples of applications, new indexing techniques, common testbeds and benchmarks, source code, and up-to-date literature through a Web page serving the similarity searching community. Authors are expected to use the testbeds and code from the SISAP Web site for comparing new applications, databases, indexes and algorithms. Contributions to the conference should fall into the following categories:

- Basic techniques: general methods that apply to arbitrary metric spaces, nonmetric or (dis)similarity spaces, or high-dimensional vector spaces.
- Applied techniques: methods that apply to specific similarity search problems.
- Spaces and similarities: poster papers that present challenges, in the form of searching in new spaces.

The Proceedings will be published by IEEE Computer Science Press. A special issue with the extended versions of the best SISAP 2008 papers will appear in the Journal of Discrete Algorithms (Elsevier). For more information visit <http://www.sisap.org>.

AMW 2007

The South American Database School and II Workshop on Foundation on Databases and the Web (AMW 2007) will take place in Punta del Este, Uruguay, from October 23 to 26, 2007. This event continues the previous edition of the Workshop on Foundations of Databases and the Web. Together with the workshop this year we will organize a School consisting in four tutorials of three hours of duration each and a Graduate Colloquium.

The School is oriented to graduate students, young researchers, researchers from neighboring disciplines, as well as industrial researchers and practitioners. The Graduate Colloquium is aimed at bringing together students within the Database and Web fields to discuss their research in an international forum. For more information, visit <http://www.fing.edu.uy/inco/grupos/csi/AMW07/>.

LATIN 2008

The 8th Latin American Theoretical Informatics Symposium will be held on April 7-11, 2008 in Búzios, Rio de Janeiro, Brazil. LATIN was launched in 1992 to foster the interaction between the Latin-American community and computer scientists around the world. LATIN'08 will be the eighth of a series, after São Paulo, Brazil (1992); Valparaíso, Chile (1995); Campinas, Brazil (1998) and Punta del Este, Uruguay (2000), Cancun, Mexico (2002), Buenos Aires, Argentina (2004), Valdivia, Chile (2006).

The invited speakers are Claudio Lucchesi (Brazil), Moni Naor (Israel), Wojciech Szpankowski (USA), Eva Tardos (USA) and Robert Tarjan (USA). For more information visit <http://www.latin08.org/>.

SPIRE 2007

SPIRE 2007 is a Symposium on String Processing and Information Retrieval in its fourteenth edition, that will be held in Santiago, Chile on October 29 - 31, 2007. SPIRE has its origins in the South American Workshop on String Processing which was first held in Belo Horizonte (Brazil, 1993). Starting in 1998, the focus of the workshop was broadened to include information retrieval due to its increasing relevance and its inter-relationship with the area of string processing. In addition, since 2000, the conference venue has been in Europe in even years. The last conferences have been in Padova (Italy), Buenos Aires (Argentina), and Glasgow (UK). As in past editions, the proceedings of SPIRE 2007 will be published by Springer in the Lecture Notes in Computer Science series.

The invited talks are "Measures of Measurements: Robust Evaluation of Search Systems" by Justin Zobel (Australia), "Trends in Online Social Interactions" by Andrew Tomkins (USA) and "Near Space-Optimal Perfect Hashing Algorithms" by Nivio Ziviani (Brazil). For more information visit <http://www.cwr.cl/spire2007/>.

Regional Events

- October 9 - 12, 2007, Valparaíso, Chile: 10th Information Security Conference (IST'07) <http://www.isc07.cl/>.
- October 23 - October 26, 2007, Punta del Este, Uruguay: AMW 2007 <http://www.fing.edu.uy/inco/grupos/csi/AMW07/>.
- October 29 - October 31, 2007, Santiago, Chile: SPIRE 2007. <http://www.cwr.cl/spire2007/>.
- October 31 - November 2, 2007, Santiago, Chile: LA-Web 2007. <http://www.cwr.cl/la-web2007/>.
- November 5 - 10, 2007, Iquique, Chile: SCCC 2007 — Chilean Database Workshop. <http://prat.unap.cl/jcc2007>.
- April 7 - 11, 2008, Rio de Janeiro, Brazil: Latin American Theoretical Informatics (LATIN'08). <http://www.latin08.org/>.
- April 11 - 12, 2008, Cancún, Mexico: SISAP 2008. <http://www.sisap.org>.

NEWS FROM NEW ZEALAND

BY

C.S. CALUDE



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

1 Scientific and Community News

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

- 306. C.S. Calude and J. Gruska. Quantum Informatics and the Relations Between Informatics, Physics and Mathematics: A Dialogue. 05/2007
- 307. A. Raichev and M.C. Wilson. Asymptotics of Diagonal Coefficients of Multivariate Generating Functions. 05/2007
- 308 M. Zimand. Two Sources Are Better than One for Increasing the Kolmogorov Complexity of Infinite Sequences. 05/2007
- 309. C.S. Calude, E. Calude and S. Marcus. Proving and Programming. 06/2007
- 310. S. Drape. The Suitability of Different Binary Tree Obfuscations. 06/2007
- 311. S. Drape and A. Majumdar. Design and Evaluation of Slicing Obfuscation. 06/2007

2 Quantum Informatics and the Relations Between Informatics, Physics and Mathematics: A Dialogue with Professor Jozef Gruska (second part)

We continue our discussion reported in the previous column.

CC: To explore the relations between the classical and quantum worlds is another challenge.

JG: Well, views on these two worlds can be very different. Niels Bohr said “There is no quantum world. There is only an abstract quantum physical description. It is wrong to think that the task of physics is to find out how Nature is. Physics concerns what we can say about Nature.” A. Zeilinger noted “The border between classical and quantum phenomena is just a question of money.” I find very interesting D. Greenberger’s position who said “I believe there is no classical world. There is only quantum world. Classical physics is a collection of unrelated insights: Newton’s laws. Hamilton’s principle, etc. Only quantum theory brings out their connection. An analogy is the Hawaiian Islands, which look like a bunch of islands in the ocean. But if you could lower the water, you would see, that they are the peaks of a chain of mountains. That is what quantum physics does to classical physics.”

CC: This is interesting.

JG: The search for such borders between classical and quantum worlds came recently to the experimental level. An important current research agenda is to find out for what kind of macroscopic objects such phenomena as superposition or entanglement hold. For example, these phenomena have been demonstrated already on an ensemble of 10^{14} atoms and on large molecules.

CC: Do you believe in Greenberg’s position, or do you think that quantum mechanics holds only in certain parts of the physical world? Certainly your car mechanic is a classical mechanic, not a quantum one.

JG: Quantum mechanics surely brought a revolution in our view of the physical world. I would like to join those expecting that this revolution is not finished. One reason is that all attempts to get within this theory a unified understanding of time and space, cosmology and gravitation failed. One has therefore to admit that this is one of the reasons quantum mechanics is still not “the theory” of the physical world. Smolin, see *quant-ph/0609109*, has recently explored the hypothesis that quantum mechanics is an approximation of another, cosmological theory, that is accurate only for the description of subsystems of the universe. He found conditions under which quantum mechanics could be derived

from the cosmological theory by averaging over variables that are not internal to the subsystem (and can be seen as non-local hidden variables of a new type).

CC: Should informatics get involved in such megastar problems?

JG: I even argue that this is one of the main challenges and tasks for theoretical informatics. These are really the *problems* theoretical informatics should try to deal with instead of being concerned so much with numerous attempts to close various $\log^* n$ and $\log \log n$ gaps, to say it metaphorically.

CC: This makes us to come to the question: what are really the main reasons to pursue quantum information processing and communication?

JG: On a common sense level, I see, as it was already mentioned, that QIPC is the result of a marriage between perhaps the two most important areas of science of 20th century: quantum physics and informatics. It would therefore be very surprising that such a marriage would not bring important outcomes for the whole science and technology. On a more technical level, I see the following reasons:

- QIPC is believed to lead to a new quantum information processing technology that will have deep and broad impacts, on science, technology and society in general.
- Several sciences and technologies are approaching the point at which they badly need expertise with isolation, manipulation and transmission of particles.
- It is increasingly believed that new, quantum information processing based tools for understanding quantum phenomena can be developed.
- Quantum cryptography seems to offer higher levels of security and could be soon feasible.
- QIPC has been shown to be more efficient in interesting/important cases.

CC: May I observe that with the exception of the last “reason” everywhere else you have used terms like “it seems”, “it is believed”. Could you give us the simplest example of a *provable* QIPC solution which is more efficient than any classical solution (except Grover’s algorithm)?

JG: Several: quantum teleportation cannot be made classically; if communicating parties share entanglement this may increase the capacity of their classical channel; in the area of quantum communication complexity, exponential separation has been proven for some problems. Finally, let me mention Simon’s problem: Check whether a given finite function is one-to-one or two-to-one.

In this case it has been proven, in a reasonable sense, that quantum solution is exponentially faster than any probabilistic solution—the weak point of this result is, however, that it is a promise problem and we work with query complexity.

CC: Deutsch’s problem—test whether a bit-function is constant or not—was considered for many years the simplest example of a problem in which the quantum solution is superior to any classical solution; apparently nobody really checked this claim. In *quant-ph/0610220* I showed that classical solutions as efficient as the quantum one exist. Is any of the above listed examples in the same category?

JG: Your classical solution of the Deutsch problem has been a big surprise. People have realised that what has been claimed to be a more efficient quantum solution of the Deutsch problem is actually a solution of a different problem, with a different black box and inputs. It was only believed that this is not something essential, but you have shown that it is. I tend to believe that this will not be the situation in the cases mentioned above, but I have to admit that I am not fully sure. Another surprising recent result along these lines is the recent discovery, , *quant-ph/0611156*, that Quantum Fourier Transform over \mathbf{Z}_q , which has been thought to be the key quantum ingredient of Shor’s algorithms, can be simulated on classical computers in polynomial time. All that actually demonstrates how little we actually know about the computational power of quantum phenomena—entanglement, superposition and measurements.

CC: In connection with that, it is perhaps useful to mention that many solutions offered by quantum information processing make a crucial use of several hard to accept, counterintuitive phenomena as randomness of quantum measurement, the existence of entangled states and quantum non-locality. In some other cases, even more weird phenomena are used, for example, quantum counterfactual phenomena. Could we now turn our attention to them and perhaps start with quantum measurement.

JG: Results, both classical and quantum, of the basic quantum projection measurement should be random and should result, in general, in a collapse of the state being measured. Already Erwin Schrödinger had problems to accept it and his position is well known: “Had I known that we are not going to get rid of this damned quantum jumping, I never would have involved myself in this business.” Albert Einstein famous claim “God does not play dice” got a superb response from Niels Bohr: “The true God does not allow anybody to prescribe what he has to do.” However, experiments seem to confirm randomness—summarised by Nicolas Gisin in his “God tosses even non local dices”—to emphasise the existence of the shared randomness the quantum measurement of entangled

states produces. Interestingly enough, physicists seem to have more problems to accept randomness than informaticians, because informatics has a lot of technical results showing the power of randomness for computation and communication. I would therefore like to say that *God is not malicious and provides us with useful randomness*. One should notice there are still very prominent old physicists and some bright young physicists having problems to accept randomness at quantum measurement. However, I have different problems concerning quantum measurement. A key step in some quantum algorithms is the measurement at which Nature finds, in a single step, for a given (actually any) integer function f , and randomly chosen y from the range of f , all x such that $f(x) = y$, and then incorporates all such x into a superposition of basic states. I have really problems to believe it fully in spite of the fact that the mathematics behind it is perfect once we accept the principles of quantum projective measurement.

CC: Entanglement is even a more esoteric phenomenon.

JG: Again, mathematically the existence of entangled states is very easy to understand. However, if physical consequences are considered, the situation is different. Look, a very simple CNOT-gate should be able to process two independent particles in such a way that they get entangled and stay entangled no matter how far away they move. This is extremely hard to believe, though experiments (not really perfect) confirmed entanglement already among very different physical objects, as, for example, photons and atoms, and even for the distance of 144 km as recently demonstrated by Weinfurter's group in open space in Canary Islands—what has been a quite shocking recent experimental outcome. In addition, using the process called entanglement swapping, one can make entangled particles that have never been interacting.

CC: In spite of that entanglement is an important information processing resource.

JG: Some even say that it is a new gold mine of the physical world because entanglement allows to create such events, impossible in the classical world, as quantum teleportation, to create quantum algorithms that are faster than any known classical algorithm (for the same problem). Entanglement is a key tool to make some communications even exponentially more efficient than what one can classically achieve; to increase the capacity of communication channels; to act as a catalyst and so on. In addition, entanglement allows to create pseudo-telepathy. Asher Peres pointed out nicely that “entanglement allows quantum magicians to do things no classical magician can do”.

CC: There are quite different views on entanglement.

JG: Indeed, entanglement has many faces. One can see it as a bridging notion between QIPC science and fields so different as condense-matter physics, quantum gravity and so on. There are various approaches to generalise this concept. A recent one is based on the idea that quantum entanglement may be directly defined through expectation values of preferred observables—without reference to preferred subsystem decomposition. Such a framework allows the existence of non-trivial entanglement within a single indecomposable quantum system ...

CC: Possibly the most controversial issue concerning entanglement is the existence of non-local correlations created by a measurement of entangled states. Could we discuss this counterintuitive phenomenon in more details?

JG: Not many realise that “physics was non-local since Newton times with the exception of the period 1915–25”, as recently Nicolas Gisin pointed out. In other words, since Newton’s time the main physical theories implied the existence of non-local phenomena with the exception of the above period. In 1915, Albert Einstein came with the theory of relativity that denies the existence of immediate non-local effects, but quantum mechanics then brought certain non-local effects back into the mainstream physics.

CC: Newton himself had noticed counterintuitive consequences of his theory of gravity.

JG: Yes, for example, Newton realised that according to his theory if a stone is moved on the moon, then weights of all of us, here on the earth, are immediately modified. However, he actually believed that the reason is an imperfection of his theory. His words on this subject are very interesting: “That Gravity should be innate, inherent and essential to Matter, so that one Body may act upon another at a Distance thro a Vacuum, without the Mediation of any thing else, by and through which their Action and Force may be conveyed from one to another, is to me so great an Absurdity, that I believe no Man who has in philosophical Matters a competent Faculty of thinking, can ever fall unto it. Gravity must be caused by an Agent acting constantly according to certain Laws, but whether this Agent be material or immaterial, I have left to the Consideration of my Readers.”

CC: Newton’s observation may further complicate the attempts of losing weight ... More seriously, Leibniz criticised Newton’s theory of gravity as a revival of the “occult properties” of medieval philosophy. However, quantum non-locality is different. It does not allow superluminal communication and therefore it does not contradict relativity. Did Einstein realise that? Can we have stronger correlations than those induced by entanglement without contradicting relativity theory?

JG: There have been many interesting recent developments in entanglement and non-locality. For example, Methot and Scarani in *quant-ph/0601210* pointed out that there are good reasons to consider quantum entanglement and quantum non-locality as two independent resources. Namely, they have shown that for the main known measures of non-locality, not maximally entangled states are maximally non-local. However, the main new impulse for the study of non-locality came from the introduction of so-called PR-boxes by Popescu and Rohrlich, in 1997, in a paper that started to attract attention only fairly recently.

CC: The introduction of PR-boxes was an unexpectedly stimulating idea.

JG: They were intended as a toy tool that demonstrates (in a reasonable sense) a non-locality stronger than quantum non-locality which does not contradict relativity. PR-boxes are easy to describe. Indeed, a PR-box can be seen as consisting of two black boxes operated by two (very) distant parties that cannot have any direct communication. If one party, say A , puts on the input of its sub-box a (random) bit x_A , then it gets, immediately, as an output, a random bit y_A . The same for other party B . However, in spite of the fact that each of the inputs and outputs are random, the outputs should be always correlated with inputs as follows: $x_A \cdot x_B = (y_A \oplus y_B)$.

PR-boxes could be very powerful. Indeed, having enough of them we could have unconditionally secure bit commitment and, moreover, each distributed computation of a Boolean function could be done using only one bit of communication, something no one could believe. This quantum communication complexity result implies that PR-boxes cannot exist physically. This result was again one of the impressive contributions of the complexity theory to quantum mechanics.

CC: In spite of that PR-boxes keep being investigated.

JG: Yes, because they play a central role in the study of non-locality that does not contradict relativity theory. For example, an interesting question is how well we can approximate PR-boxes. It was shown that with shared entanglement one can approximate PR-boxes with success probability 0.854 and in no physical world this can be done with success probability of more than 0.908.

CC: Quite interesting! Counterfactual effects are other mysterious phenomena.

JG: I see them as another indication that something may not be OK in our understanding of the physical world. Counterfactual effects allow, for example, for the possibility to get the result of a quantum computation without actually

performing the computation. Recently, Paul Kwiat has presented the first demonstration of counterfactual computation using an optical-based quantum computer (see the Feb. 23 issue of *Nature*).

CC: A deeper understanding of all these phenomena is a challenge for physics.

JG: And for informatics too. There are of course many other very big challenges. Let me mention some of them: Is our universe computable? Efficiently computable? Is our world a polynomial or an exponential place? (As pointed out by Scott Aaronson.)

CC: Could our world be exponential?

JG: Well, without believing in exponentiality of our physical world we could have problems to explain some experiments already done. We do not consider as feasible a computation requiring exponentially growing number of steps, but no one actually seem to complain to have exponentially large probability distributions. The situation with exponentiality is therefore far from obvious and far from simple. As pointed out by Goldreich, we may need more realistic complexity models of quantum computations and, I think, also of communication.

CC: Can we really have a powerful quantum computer?

JG: This challenge is an important current research agenda for physics and informatics. Could it happen that quantum mechanics “breaks down” before factoring large integers? Landauer was perhaps the first sceptic and his statement “One will need more than rain to stop this parade” reflects feelings of his time, but these feelings keep coming back again and again.

CC: Why “quantum mechanics can break down before factoring very large integers”?

JG: There are many arguments. From the history of physics one can extrapolate that each theory has its limits and therefore one could expect that current quantum mechanics does not hold for too small and too large scales. Some believe that the size of measuring devices will have to grow exponentially. In addition, there are people believing that we cannot fight decoherence or theoretical results that claim that if the reliability of elementary gates and “wires” reaches a certain threshold, then quantum information processing can be done in any time and space distance, are wrong or improperly interpreted.

CC: On a more general level an important problem is that of feasibility in physics.

JG: Feasibility in physics was for a long time determined by the following statement of Dirac: “Can every observable be measured? The answer theoretically is yes. In practice it may be very awkward, or perhaps even beyond the ingenuity of the experimenter, to design an apparatus which could measure some particular observable, but the theory always allows one to imagine that the measurement can be made.” Nowadays, it is obvious that this is not so. Theoretically, we can have quantum states with uncomputable amplitudes. It is therefore clear that not everything one can find in quantum theory is feasible in practice.

Informatics is already quite far in its attempts to develop important concepts of feasibility. It first realised that there are non-computable numbers and later that there are unfeasible tasks and hard to compute computable numbers. It seems to me that physics is still behind the goal to get a very reasonable concept of feasibility. And it is an important task for both physics and informatics to work on it.

CC: Dirac’s idea was that unitaries and projective measurement in Hilbert space exist in Nature. The fact that there are uncomputable numbers and unsolvable problems can be seen as implying that not all unitaries and measurements can be constructed. Does it mean that they cannot exist in Nature?

JG: This is an interesting and fundamental question. I do not have a sharp view on this issue. Is the question about the existence of a different category than the existence of uncomputable numbers? I guess yes.

CC: Let us go back to the possibility of constructing universal quantum computers. This is a much discussed question. What do you think?

JG: First of all, it is far from clear whether we would really need them for usual computations. The number of cases they may be more efficient can be practically small. That can be seen from the fact that we still have relatively few impressive quantum algorithms. A need for a general purpose quantum computer is therefore questionable. Another issue is the need to have powerful quantum special purpose processors or devices to simulate quantum phenomena and processes. To make a long story short, I believe that either we will have quantum computers or we will discover some new important limitations of the physical world.

CC: This brings us to the controversial issue of interpretations of quantum theory. There are various sophisticated interpretations and a lot of articles have been written on this issue by scientists and philosophers of science.

JG: I think that there is a sophisticated mess concerning interpretations. I like the observation that not only philosophers of science cannot agree on a particular interpretation, but they have even a problem to agree on what is an interpretation.

On the other hand, I believe that outcomes of quantum informatics, especially in the area of quantum computation and communication theory, but also in cryptography, broadly understood, can bring more light into various interpretations and into the relations between them.

CC: By the way, how did you get involved in quantum information processing?

JG: In 1989, after being appointed as chairman of the newly created IFIP Specialist Group on Foundations of Computing (SGFCS 14), I worked out a very ambitious, and idealistic, program how SGFCS 14 could support the development of TCS. One of my suggestions was to create a working group Informatics and Physics. No one complained, but such an idea turned out to be too much ahead of time. In 1992 and 1993, during my three years stay at the University of Hamburg, I run, together with Manfred Kudlek, a physicist by education, a seminar “Informatics and physics”. One of the papers we discussed was Deutsch’s paper where the model of quantum Turing machine was introduced ... To make a long story short, in 1997–99 I wrote, partly on the beach in Nice, my book *Quantum Computing*.

CC: Could we now discuss the relations between physicists and informaticians. My first question is what should informaticians learn from physicists?

JG: I see three main directions: (1) physics sells itself better and in a more mature way; (2) physics is better organised; (c) physics has a better and mature publication policy.

CC: Of course, physics is much older than modern informatics.

JG: Correct, but still differences concerning the quality of selling are enormous. The main impulse for enormous support for physics came from the needs of the Second World War, and later of the cold war. Informatics has made in the last 50 years arguably larger contributions to science and society, but still the amount of money going into physics is much larger than the amount informatics gets. I think physics leaders have realised that the society support of science is not mainly due to the fact that it brings important outcomes, but that generates mysterious problems and phenomena and solves some of them. Physicists have made some great marketing moves as making one of their goals to create a *theory of everything* (S. Hawking). Who could really believe in it? But this idea brought much support for physics in general and in UK in particular.

CC: Ignoring age, do you see any specific reasons why informatics is not as well organised as physics?

JG: Informaticians should start to understand that the way a field performs as the whole depends not only on how many clever young people it has, but even more on how many wise people it has in its leadership. The current value system in informatics, with such emphasis on accepted papers at “prestigious conferences”, prefers young bright people and even the middle generation is very soon “out”. This seems to be true especially in theoretical informatics. As a consequence, the field is scientifically doing very well, concerning solving hard open problems, but far less in the attempts to attack important new problems of the field and of science in general. The overall standing of theoretical informatics within the informatics community is quite low and goes actually down, quite fast, I think. Those that should be and could be leaders are put much too soon aside. Everybody in the field is then paying for that, in long terms.

CC: It seems that an emphasis on 3-4 page long papers dominates the publication policy in physics ...

JG: From the point of view of physics that was a clever idea. It is now embraced by a large number of authors. By writing two pages a scientist can have 10 publications (with 10 authors each) and if each such paper is cited, then physics has 100 citations. I am a bit dramatising situation, but not essentially. In any global evaluation of sciences, physics (and other natural sciences) dominate, to a large extend due to such publication policy, and, as a consequence, their interests dominate the current science in spite of the fact that the global interests of society would need to put the focus to other areas of science.

CC: Since there are now large possibilities for electronic publishing, informatics should be a leader.

JG: But it is not, and it is getting, again, far behind physics. Look how much (and how cleverly) physicists use the Los Alamos archive. However, this is not all. Informatics clearly needs a more mature publishing policy. The current publishing policy in informatics, inherited, to a very large extend, from mathematics, is not well suited for informatics. As a consequence, once the number of publications, citations and impact factors start to be counted, informatics looks like performing not so well than areas of science with arguably smaller current impact on science, technology and society.

CC: On the other side, what physics can learn from informatics?

JG: On a very general level, one can say that informatics offers for (quantum) physics paradigms, concepts, and results that can allow physics to see sometimes faster what is impossible, to formulate and to sharp better results and to see deeper into the physical world.

One can say that quantum information processing concepts, paradigms, models and results forced physicists to reshape their ideas of reality, to rethink the nature of things at the deepest level, to revise their concepts of position and speed, their notion of cause and effect, ...

CC: And what physicists should learn from informaticians?

JG: Many things. In the area of quantum information processing, the use of the big-O notation to express scalability and feasibility. Then, an understanding that after learning an issue for small cases one should try to understand the general case, and to replace hand-waving arguing by precise proofs. Physicists are starting to learn that using complexity-theoretic models and results one can learn that certain phenomena are (likely) impossible and to understand the power of various quantum information processing resources, as entanglement, and non-locality.

One should realise that informatics has brought new views on old phenomena, and that is perhaps its main contribution.

CC: Complexity theory seems to be the main area of theoretical informatics physicists may find useful ...

JG: Correct. One can even say that the main reason why already von Neumann did not come with the idea of quantum information processing was the fact that in his time science couldn't see that quantum information processing would pay off. It was mainly due to (quantum) complexity results that made clear that quantum computing could pay off. Moreover, the main killer-applications for the whole field were actually Shor's algorithms motivated by (quantum) structural complexity results.

However, I believe that other areas of theoretical computer science can significantly contribute to our understanding of the physical world. For example a recently emerging *quantum programming, specification and reasoning theory*.

CC: What else in informatics may be useful to physicists?

JG: For an informatician it is almost shocking the level of referencing in physics. For example, they do not write titles of the articles in references and the paper size (last page), though this is often a very important information.

CC: There has to be a reason for that.

JG: As I see it, a well-done dissemination of knowledge is still not the main goal of publications in physics as it is in informatics. The main goal of the whole publication system in physics seems to be a fast documentation of the priority of new discoveries. Physics was for centuries influenced by goals and customs that dominated when the science started. I think most of physicists even do not

realise what is behind the rules that are imposed on them by journals and their publication culture. The emphasis on ensuring the priority of authors as the main goal of publications has as a consequence the fact that papers are (actually have to be) written in such a way that they are not easy to be read, checked and understood. This does not seem to be desirable. The main goal seems to be able to say (for authors): I was (we were) first to do that and that, it was published in ... Another goal of such publications, again inherited from old times, seems to prevent, as much as possible, the reader to make very fast use of the published results. One can say that physics papers are, at least to a large extent, “readers-unfriendly”. The younger generation of physicists started to be different. While they may not realise why the publication policy is as it is, they still do not have enough power to change long time ago established policies and traditions. However, I should notice that a similar publication policy was used in former Soviet Union in mathematics. Easy to read papers had small chances to get accepted. Authors were under pressure to establish heavy formalism and to compress the paper as much as possible.

CC: I see this kind of tendency in many papers in theoretical computer science ...

JG: This is true, but this is more due to the abstraction the field goes into, the difficulty to describe informally formal systems and reasoning about them. Fortunately, nowadays we do not depend so much on powerful editors. You can just put the complete version on your web page. Moreover, journals compete (and put prices) for best papers, ...

CC: Are the two communities of quantum information processing, one coming from physics and one coming from informatics, getting closer?

JG: Significantly, and I think that many physicists in QIPC have started to fully realise the power of informatics methods, and use them. (Actually, I do not know any other area of science, where TCS outcomes would be so useful and had such big impacts as in QIPC. QIPC can be seen as a really big success story of TCS.) Informaticians have started to appreciate the importance of many related theoretical problems of physics.

CC: Many have the feeling that after a big boom during 1993–96, the progress in the area of QIPC has been recently far less spectacular. Is it true?

JG: Comparing with 1994, the number of papers submitted to quantum archive has increased more than 10 times. This characterises pretty well the increase of the research in this area. Both theory and experiments have made an enormous progress. Theory results seem to be more technical and less

spectacular. Experimental results, especially in quantum cryptography, may not be spectacular for a non-specialist, but for experts it has been achieved recently far more than one could have expected 10 years ago. For example, the first experiment in quantum cryptography was based on the transmission of photons for a distance of 32.5 cm. Nowadays the maximum is approaching 200 km, when fibres are used and was done for 27 km, from one peak to another, in Alps, and for 144 km, from one island (La Palma) to another (Tenerife), via an optical free-space link. Some foresee even transmission to 1000 km using quantum repeaters. Hardly someone could have believed in such achievements 10 years ago. Experimental cryptographic networks, for example the DARPA network in Boston, are remarkable achievements.

CC: The situation seems to be very different in the attempts to design more powerful quantum processors. Factorisation of the number 15, and an experiment with 8 qubits, are still the most publicised results and that is far from being impressive.

JG: It is correct that impressive results in this direction are missing. The field is in the process of exploring various technologies and searching for various primitives and from this point of view the field is in cumulating state of knowledge, methods and experience. There have been many surprising outcomes, as, for example, an understanding that quantum computation can be performed by measurements only, or the idea of one-way computing. It is true that there are still many pessimists not believing that we can win our fight with decoherence. However, while progress in science is often done by pessimists, progress in technology is always done by optimists. I like to remember the famous story of the Colossus, of our first really powerful electronic computer, designed by Tommy Flowers, in a post office laboratory, in spite of the fact that his proposal was rejected by a panel of experts as unfeasible. He did that because he knew that thermionic valves are reliable provided they are not turned on and off too often.

CC: Could a discovery of a simple technology make a miracle for quantum computing?

JG: Who knows, I believe so, I am an optimist.

CC: It takes a long time until new ideas make a real impact.

JG: Of course. For example, in the development of the last three centuries we can notice, from the science and technology point of view, the following common scenarios:

19th century was mainly influenced by the first industrial revolution that had its basis in the classical mechanics discovered, formalised and developed in the

18th century.

20th century was mainly influenced by the second industrial revolution that had its basis in the electrodynamics discovered, formalised and developed in the 19th century.

21th century can be expected to be mainly developed by quantum mechanics and informatics discovered, formalised and developed in the 20th century.

To summarise, it used to take about a century for new discoveries in science and technology to have a decisive and global impact on the society developments, and usually in a way no one could image at the very beginning.

CC: We have started our discussion with an observation that the concept of information plays such an important role nowadays for physics and our understanding of the physical world. A similar situation may apply to the security and related cryptographic concepts.

JG: Indeed, I am expecting, more and more, that basic cryptographic (in a broad sense) concepts will play a very important role in our understanding of both information processing and physical worlds. They may play an even more important role (than the concepts related to information processing and transmission themselves) in our understanding of the laws and limitations of the physical and information worlds. Moreover, the impact of cryptography, again in a broad sense, goes fast even much farther. Growing needs to provide security, privacy, anonymity and authentication, especially in connection with one of the “ultimate, and never fully reached, goals of science and technology”—the design of global computation and communication networks, called usually grid networks, will create big and important specialised industries. This can be even one of the important driving forces of many industries and of the overall development of society.

CC: You seem to have again a very strong position . . .

JG: I would also like to foresee, as another important area of science and technology, the emerging security science and technology. A science not only for security providing technologies, but as a really fundamental science. And not only that.

It is well known that history of mankind can be seen, in a very simplified form, as consisting of the following three eras. Observe that their descriptions differ basically only by one word. The three magic words are **food**, **energy** and **information**.

Neolithic era: Progress was made on the basis that man learned how to make use of the potentials provided by the biological world to have **food** available in a sufficient amount and whenever needed.

Industrial era: Progress has been made on the basis that man has learned how to make use of the laws and limitations of the physical world to have **energy** available in a sufficient amount and whenever needed.

Information era: Progress is and will be made on the basis that man learns how to make use of the laws and limitations of the information world to have **information** available in a sufficient amount and whenever needed.

In this context the following question arises: What can we expect to have as “being” in the fourth era to come? Of course, this is hard to predict. *Artificial* (worlds, intelligence, life,...)? That may be the case, but I would like to foresee the fourth coming era as one in which the key concepts are those of security, safety, privacy, anonymity and so on. Modern cryptography, broadly understood, is the key science behind.

CC: How do you see modern cryptography?

JG: The general goal of modern cryptography is the construction of schemes which are robust against malicious attempts to deviate from a prescribed functionality. The fact that *an adversary can devise its attacks after the scheme has been specified* makes the design of such schemes very difficult—schemes should be secure under all possible attacks. It makes very difficult to specify precisely enough when a cryptographic scheme is perfectly secure.

CC: We have several concepts of security.

JG: Correct: informational security—an enemy has not enough information to break the scheme; computational security—an enemy cannot have enough computational power to break the scheme and so called unconditional security—an enemy cannot break the scheme, due to physical laws, no matter how much computational power she has.

CC: How successful is actually our “fight” for security?

JG: In this area we have a constant fight between “good” and “bad”. Both sides are trying to (maximally) use whatever sciences and technologies bring us. Adi Shamir said that concerning security *we are winning battles, but losing wars*. One of the key issues is that society has still problems to realise and accept that security is very costly, requires sophisticated tools, and we have to pay for it with time and freedom.

CC: Why the study of security would lead to deeper issues into information processing and physical worlds?

JG: Look, in all problems concerning security, authentication, but especially concerning more subtle problems of anonymity and privacy, we have as goal to get *perfect* or *unconditional* security, anonymity, privacy, authentication and such a goal is much more demanding than to have more efficient computation or asymptotically best computation.

CC: Already well-known fundamental cryptographic concepts, as one-way function, one-way function with trapdoor, hard predicate, zero-knowledge proof, are fundamental for information processing.

JG: In addition, look how stimulating the study of variations of bit commitment, coin tossing and oblivious transfer protocols, has been for our understanding of the quantum information processing world and its relations to the classical information processing world.

CC: Please highlight a simple result in quantum cryptography.

JG: Unconditionally secure generation of classical keys and the impossibility of having unconditional secure bit commitment are theoretical highlights. However, even more surprising, at least for me, is that using a simple quantum version of a one-time pad cryptosystem one needs only two bits to hide perfectly any qubit even if its specification requires infinitely many classical bits.

CC: How about relations between cryptographic concepts and foundational issues of quantum mechanics?

JG: One of the big things of interest to foundational people is whether we can derive quantum mechanics from some simple axioms that have a natural physical, or information processing based, interpretation. Fuchs and Brassard suggested to consider as axioms (a) the existence of unconditionally secure cryptographic key generation, and (b) the impossibility of secure bit commitment. One such attempt was done, as I have already mentioned last time, by Clifton, Bub and Halvorson with three axioms: No signalling, no broadcasting, and no bit commitment.

CC: At a first glance, it seems odd that quantum mechanics could be derived from the axioms (a) and (b).

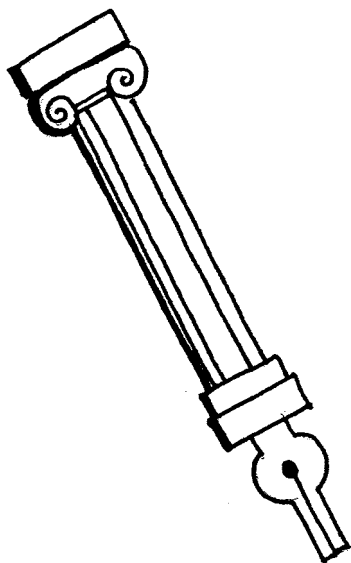
JG: Actually, it is not. Look, unconditional secure key generation is possible only if the no-cloning theorem holds and quantum measurement causes a disturbance of quantum states. Unconditionally secure bit commitment is impossible only in case we have correlations similar to those quantum entanglement provides. And here we are.

CC: We can therefore expect interesting developments at the intersection between informatics and physics.

JG: Of course, and at the end of our discussion I would only like to mention several citations to illustrate how the views of the physics and physical world keep changing. Demokritos is quoted as saying (400 BC) *Nothing exists except atoms and empty space; everything else is opinion*; Ernest Rutherford said in 1912, *All science is either physics or stamp collecting*. My position is that *Physics is not the only science capable of producing a deep understanding of physical world. Informatics can and should help. Or, even, it should take the initiative*.

CC: Many thanks for this interview.

THE EATCS COLUMNS



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>

The initial development of efficient Quantum algorithms for the factorization of integers drew much attention to this area of computation. It was soon observed that the methods could be extended to other symmetry finding scenarios like the hidden subgroup problem for the case of abelian groups. The question of whether quantum methods could also solve the problem for general finite groups is a central one in quantum computing. A positive solution to it would imply efficient quantum algorithms for the Graph Isomorphism problem. Gorjan Alagic and Alexander Russell give in this column an excellent overview of the progress achieved in the last years towards the solution of this exciting question.

QUANTUM COMPUTING AND THE HUNT FOR HIDDEN SYMMETRY

Gorjan Alagic*

Alexander Russell[†]

1 Introduction

In 1994, Peter Shor gave efficient quantum algorithms for factoring integers and extracting discrete logarithms [20]. If we believe that nature will permit us to

*University of Connecticut, alagic@math.uconn.edu

[†]University of Connecticut, acr@cse.uconn.edu Supported by NSF CAREER award CCR-0093065, NSF grants EIA-0523456 and EIA-0523431, and ARO-ARDA grant 47976-PH-QC.

faithfully implement our current model of quantum computation, then these algorithms dramatically contradict the Strong Church-Turing thesis.¹ The effect is heightened by the fact that these algorithms solve computational problems with long histories of attention by the computational and mathematical communities alike.

In this article we discuss the branch of quantum algorithms research arising from attempts to generalize the core quantum algorithmic aspects of Shor’s algorithms. Roughly, this can be viewed as the problem of generalizing algorithms of Simon [21] and Shor [20], which work over abelian groups, to general nonabelian groups.

The article is meant to be self-contained, assuming no knowledge of quantum computing or the representation theory of finite groups. We begin in earnest in Section 2, describing the problem of *symmetry finding*: given a function $f : G \rightarrow S$ on a group G , this is the problem of determining $\{g \in G \mid \forall x, f(x) = f(gx)\}$, the set of *symmetries* of f . We switch gears in Section 3, giving a short introduction to the circuit model of quantum computation. The connection between these two sections is eventually established in Section 4, where we discuss the representation theory of finite groups and the quantum Fourier transform - a unitary transformation specifically tuned to the symmetries of the underlying group. Section 4.2 is devoted to *Fourier sampling*, the basic algorithmic method that connects the symmetry finding problem and the Fourier “symmetry” basis effected by the quantum Fourier transform. Finally, we discuss some algorithmic successes in Section 5.

The reader should be cautioned that in many places we have forsaken precision for the sake of improved readability (such as it is), and have made no attempt to survey the rich and fascinating landscape of quantum algorithms. In fact, our selection of algorithms to highlight in Section 5 is motivated by an attempt to emphasize the relationship between representation theory and quantum algorithms; as such, many of the exciting technical advances in this area are unrepresented. The reader is encouraged to explore these other directions; indeed, one recent and closely related development is the discovery of efficient algorithms for finding hidden nonlinear structures in vector spaces over finite fields [6, 7].

2 Symmetries and Computation

Groups were invented to abstract the concept of symmetry. After all, if A is a geometric object then the identity map on A is surely a symmetry; furthermore, composing two symmetries ought to result in a symmetry, as should “inverting” a sym-

¹By this we mean the statement “Any reasonable model of computation can be efficiently simulated on a probabilistic Turing machine.” [5]

metry. The computational problem we describe below is the problem of *inferring* the family of symmetries of a given object (typically combinatorial rather than geometric, in our story). For example, the group S_{10} acts on the Petersen graph, of Figure 1, by permuting the 10 vertices. The symmetries of the Petersen graph under this action are those permutations that preserve incidence—these symmetries form a subgroup isomorphic to S_5 .²

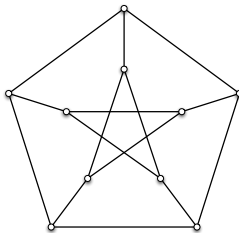


Figure 1: The Petersen graph.

Let G be a finite group that *acts* on a set X . This means that we associate with every group element g a permutation π_g of the set X in such a way that the group operation of G is respected: $\pi_g \circ \pi_h = \pi_{gh}$. Thus the map $g \mapsto \pi_g$ is a *homomorphism* from G into the group of all possible permutations of X . We will write $g \cdot x$ or gx for $\pi_g(x)$ when it won't cause confusion. Examples of groups acting on sets are everywhere: (i.) The group S_n of all permutations of n letters acts on n letters by...permutation! (ii.) Any group G acts on itself by left-multiplication, associating with each element g the permutation $\pi_g : h \mapsto gh$. (iii.) The group A_4 acts as the symmetries of a regular tetrahedron. For more discussion, see Dummit and Foote's text [8]. Observe that if G acts on the set X , then it also acts on the set of all *functions* $\{f : X \rightarrow S\}$ by the rule

$$g \cdot f : x \mapsto f(g^{-1} \cdot x). \quad (1)$$

(Here $g \in G$, $f : X \rightarrow S$ is a function, and the $^{-1}$ is introduced so that this action composes correctly: $g_1(g_2(f)) = (g_1g_2)f$.) For a function f on X , the *symmetries* of f are the group elements g for which $g \cdot f = f$. These elements form a subgroup $S(f) = S_G(f)$ of G . While we will often drop the subscript G in this notation, $S_G(f)$ does of course depend both on the choice of G and the details of the action of G on X .

²One pleasing way to view the S_5 action on the Petersen graph P is to identify the vertices of P with the 10 subsets of $\{1, \dots, 5\}$ of size 2 in such a way that adjacent vertices correspond to pairs of subsets with nontrivial intersection; the obvious action of S_5 yields the entire family of automorphisms of P .

Our signature computational problem shall be the problem of *determining the symmetries $S(f)$ of a function f* . We shall be very generous regarding the “presentation” of the function f (and the group G), merely asking that $f : X \rightarrow S$ be provided as an oracle. We shall assume, among other things, that the elements of X (and G) have a canonical representation as strings of length $O(\log |X|)$ (and $O(\log |G|)$) and that we can perform operations such as g^{-1} , $g_1 g_2$, and $g \cdot x$ in polynomial time in the lengths of these representations.

Before any discussion of quantum computation, we advertise below a number of interesting computational problems that directly reduce to symmetry finding.

2.1 Order finding

Let us consider the possible symmetries of a function $f : \mathbb{Z}_n \rightarrow S$, where $\mathbb{Z}_n = \{0, \dots, n-1\}$ is the cyclic group of size n acting on itself by translation (i. e., addition modulo n). When is f invariant under a translation? This is the problem of *period finding*: we say that a function f on \mathbb{Z}_n is *t-periodic* if $\forall x, f(x+t) = f(x)$. The least such integer t is called the *period* of f . It is easy to show that f is *m*-periodic if and only if m is an integer multiple of the period t . Evidently, the subgroup $S(f)$ of all symmetries of f is the cyclic subgroup of \mathbb{Z}_n generated by t . In this case, the group in question is $G = \mathbb{Z}_n$, and the set on which the group acts is $X = \mathbb{Z}_n$ as well.

A similar problem, that of *order finding*, is a central component of Shor’s celebrated quantum algorithm for factoring [20]. Recall that for a prime p , the group \mathbb{Z}_p^* is the set $\{1, \dots, p-1\}$ under multiplication modulo p . Order finding is the problem of determining the (multiplicative) order of an element x of \mathbb{Z}_p^* , i. e., determining the least integer t such that $x^t \equiv 1 \pmod{p}$. If we define $f : \mathbb{Z}_{p-1} \rightarrow \mathbb{Z}_p^*$ by the rule

$$f : s \mapsto x^s \pmod{p}$$

then the period of the function f is one more than the order of x . Thus order finding reduces to symmetry finding. In order finding, the group \mathbb{Z}_{p-1} acts on itself, so that $G = X = \mathbb{Z}_{p-1}$.

2.2 Hidden shifts of the Legendre symbol

Let p be a prime number. A nonzero element x of \mathbb{Z}_p is called a *quadratic residue* modulo p if there exists an integer n such that $n^2 \equiv x \pmod{p}$. It is not hard to check that setting

$$\chi_2 : x \mapsto \begin{cases} 0 & \text{if } x = 0; \\ 1 & \text{if } x \text{ is a quadratic residue modulo } p; \\ -1 & \text{otherwise.} \end{cases}$$

defines a multiplicative function on \mathbb{Z}_p (so that $\chi_2(yz) = \chi_2(y)\chi_2(z)$). The image of x under χ_2 is known as the *Legendre symbol* of x modulo p . “Shifting” the function χ_2 has been proposed as a pseudorandom generator [22]: specifically, fixing a prime p and a length $\ell > \log p$, translation of χ_2 by a randomly selected element $t \in \mathbb{Z}_p$ defines a sequence

$$\chi_2(t), \chi_2(t+1), \dots, \chi_2(t+\ell).$$

One way to “break” such a generator is to completely recover t from this sequence of values. To place the problem of breaking the pseudorandom generator in our context, we consider the (potentially easier) problem of determining t given oracle access to the entire shifted function

$$f_t : x \mapsto \chi_2(x+t).$$

Though the sequence of bits $\chi_1(0), \dots, \chi_1(\ell)$ has been conjectured to be pseudorandom for appropriate values of ℓ , the function f_t defined above does possess some rich symmetries. To capture these symmetries, we introduce the group of *affine linear transformations* of \mathbb{Z}_p . A function $\alpha : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$ is *affine linear* if it has the familiar form $\alpha(x) = ax + b$, where $a \neq 0$. (Here $a, b \in \mathbb{Z}_p$ and all arithmetic is done modulo p .) These functions form a nonabelian group, denoted \mathbb{A}_p , under composition: observe that if $\alpha(x) = ax + b$ and $\alpha'(x) = a'x + b'$ then $\alpha \circ \alpha'(x) = aa'x + ab' + b$.

Recalling that \mathbb{A}_p acts on the *functions on \mathbb{Z}_p* via equation (1), we investigate the symmetries of f_t under \mathbb{A}_p . A short calculation shows that if $\alpha(x) = ax + t(a-1)$ and, furthermore, $\chi_2(a) = 1$, then

$$\begin{aligned} [\alpha^{-1} \cdot f_t](y) &= f_t(\alpha y + t(a-1)) = \chi_2(\alpha y + t(a-1) + t) \\ &= \chi_2(a(y+t)) = \chi_2(a)\chi_2(y+t) \\ &= \chi_2(y+t) = f_t(y). \end{aligned}$$

It is easy to check that elements of the form $x \mapsto ax + t(a-1)$, where a is a quadratic residue, each represent a symmetry of f ; together these form $S(f_t)$, the symmetry subgroup of f_t (under this \mathbb{A}_p action). Moreover, each f_t induces a *different* subgroup of \mathbb{A}_p ; evidently, solving this “hidden shift” problem can be reduced to symmetry finding under the group \mathbb{A}_p , acting on the set $X = \mathbb{Z}_p$.

2.3 Graph automorphism and isomorphism

Let $\mathcal{G} = (V, E)$ denote a simple undirected graph with vertex set $V = \{1, \dots, n\}$ and edge set E . An *automorphism* of \mathcal{G} is a permutation π of the vertices V that preserves incidence: $(v, w) \in E \Leftrightarrow (\pi(v), \pi(w)) \in E$. These automorphisms,

taken together, form a subgroup $\text{Aut}(\mathcal{G})$ of S_n , the group of all permutations of $V = \{1, \dots, n\}$. The *graph automorphism* problem is the problem of determining if $\text{Aut}(\mathcal{G})$ is nontrivial (that is, if there is a nontrivial automorphism of \mathcal{G}).

Let us express this problem in our framework of symmetry finding. First, we let a permutation $\pi \in S_n$ act on an edge (u, v) by the rule $\pi(u, v) = (\pi(u), \pi(v))$; it acts on the edge set E by $\pi E = \{\pi(u, v) \mid (u, v) \in E\}$. Define the function $s_{\mathcal{G}}$ on S_n by the rule

$$s_{\mathcal{G}}(\pi) = \pi E.$$

Then $\text{Aut}(\mathcal{G})$ is precisely the group $S(s_{\mathcal{G}})$: thus solving the symmetry finding problem in this case determines the entire automorphism group of \mathcal{G} (and, in particular, the answer to the graph automorphism problem for this graph). This is another example of a symmetry finding problem, where the group is $G = S_n$ acting on itself.

We remark that the task of determining if two graphs \mathcal{G}_1 and \mathcal{G}_2 are *isomorphic* reduces to the above situation by considering the automorphisms of the (disjoint) union $\mathcal{G}_1 \cup \mathcal{G}_2$. Indeed, if the graphs are not isomorphic, the only automorphisms of $\mathcal{G}_1 \cup \mathcal{G}_2$ would be of the form $\pi \cup \sigma$, where π is an automorphism of \mathcal{G}_1 and σ is an automorphism of \mathcal{G}_2 . On the other hand, if $\tau : \mathcal{G}_1 \rightarrow \mathcal{G}_2$ is an isomorphism, then there exists an automorphism of $\mathcal{G}_1 \cup \mathcal{G}_2$ which swaps \mathcal{G}_1 and \mathcal{G}_2 , and then applies $\tau^{-1} \cup \tau$.

3 Quantum computing

For the sake of analogy, we begin by retelling the story of classical computation: beginning with an input vector $\mathbf{v} \in \{0, 1\}^n$, we carry out a sequence of *local transformations*, each of which affects but a constant number of coordinates by subjecting them to some function $t_i : \{0, 1\}^c \rightarrow \{0, 1\}^c$. In order for this story to really capture the familiar circuit model, we need to be more generous with “workspace,” initially embedding \mathbf{v} into the first n coordinates of $\{0, 1\}^\omega$. Finally, we honor the first bit of the result by allowing it to determine the *outcome* of this computation which, in this way, determines a Boolean function. The story is most exciting when it realizes some Boolean function we care about as a particularly short (or particularly nicely-structured) sequence of transformations.

The story of quantum computation can be told by “unitarily extending” the tale above: beginning with an input vector $\mathbf{v} \in H^{\otimes n}$, we carry out a sequence of *local unitary transformations*, each of which affects but a constant number of coordinates by subjecting them to an arbitrary unitary (i. e., length-preserving linear) operator $t_i : H^{\otimes c} \rightarrow H^{\otimes c}$. The first unfamiliar character appearing in the quantum tale is H , which we take to be a 2-dimensional complex vector space, on which we have an inner product $\langle \cdot, \cdot \rangle$ and hence also a notion of length: $\|x\|^2 = \langle x, x \rangle$. To

maximize our analogy with the classical story, we select an orthonormal basis for H consisting of two orthonormal vectors named $|0\rangle$ and $|1\rangle$. Leaving the details of what we mean by $H^{\otimes n}$ and a unitary operator “affecting but a constant number of coordinates” to the reader’s imagination for the moment, we continue to say that in this quantum case, as well, we shall embed $H^{\otimes n}$ inside $H^{\otimes \omega}$ to be equally generous regarding workspace and, furthermore, allow the first coordinate of the resulting state to determine the outcome of this computation in an appropriate fashion. For our purposes, an *efficient* quantum computation is one that involves a polynomial number of local transformations.

Despite whatever “compelling analogies” the reader has been cajoled to accept by the story above, this computational model appears complicated and suspicious, especially if you have been brought up on a healthy diet of recursive function theory, Turing machines, and classical circuit complexity. However, this model of “unitary evolution” is one of the essential features of quantum mechanics, a physical model that predicts both qualitative and quantitative features of the small-scale behavior of the universe. To the best of our current knowledge, the many fundamental curiosities and surprises that manifest in quantum mechanics are facts of nature, though they be wildly inconsistent with our physical intuition born of a lifetime of experience with classical macroscopic phenomena. Engineering advances of the last decade demonstrate that these quantum phenomena are available for human tinkering and, hopefully, for the highly-structured sort of tinkering that would comprise “computation.”

Let us return to flush out the remaining details of the model of quantum computation introduced above. The state of the computation, throughout the process above, lies in $H^{\otimes n}$, the n -fold *tensor* power of H . In general, if A and B are vector spaces with orthonormal bases $\{\mathbf{a}_i\}$ and $\{\mathbf{b}_j\}$, a *tensor product* of these spaces is a vector space, denoted $A \otimes B$, along with a map $(\mathbf{a}, \mathbf{b}) \mapsto \mathbf{a} \circ \mathbf{b}$ of $A \times B$ to $A \otimes B$ so that \circ is linear in each coordinate and, furthermore, the set $\{\mathbf{a}_i \circ \mathbf{b}_j\}$ is a basis for $A \otimes B$. As we assume that A and B have an inner product, we can naturally define an inner product on $A \otimes B$ by extending the rule $\langle \mathbf{a} \circ \mathbf{b}, \mathbf{a}' \circ \mathbf{b}' \rangle = \langle \mathbf{a}, \mathbf{a}' \rangle \langle \mathbf{b}, \mathbf{b}' \rangle$; this will, in particular, make the basis $\mathbf{a}_i \circ \mathbf{b}_j$ above orthonormal. Unfolding the definition and trusting that \circ is associative (it is), we find that $H^{\otimes n}$ is a vector space with 2^n orthonormal basis vectors

$$|a_1\rangle \circ \cdots \circ |a_n\rangle, \quad (a_1, \dots, a_n) \in \{0, 1\}^n,$$

one for each binary string in $\{0, 1\}^n$! It is customary to use the symbol \otimes for the map \circ above (even though this overloads the symbol, which we also used to denote the vector space $A \otimes B$); furthermore, it is customary in this case to use the notation $|a_1 \dots a_n\rangle$ for the vector $|a_1\rangle \otimes \cdots \otimes |a_n\rangle$.

This apparent relationship between the basis of $H^{\otimes n}$ and the “intermediate states” that appeared in our notion of *classical* computation is not an accident.

Indeed, our rule for feeding (classical) inputs to a quantum machine will be to identify the classical input $a \in \{0, 1\}^n$ with the unit-length basis vector $|a\rangle \in H^{\otimes n}$.

It is easy to check that if U is a unitary operator on the vector space A , then U naturally defines a unitary operator on $A \otimes B$ by the rule $\mathbf{a} \otimes \mathbf{b} \mapsto (U\mathbf{a}) \otimes \mathbf{b}$. It is precisely this sense in which the local operators discussed in the definition of computation above are applied to a “constant number of indices.” The sequence of unitary (and hence length-preserving) operators associated with a quantum computation thus carries any unit length input vector \mathbf{a} to a unit length result \mathbf{r} . While we naturally have a clear interpretation of the input vector \mathbf{a} as an element of $\{0, 1\}^n$, the result \mathbf{r} is typically a linear combination

$$\mathbf{r} = \sum_{a \in \{0,1\}^n} \alpha_a |a\rangle, \quad \alpha_a \in \mathbb{C}$$

with exponential support. What we *can* be sure of, however, is that \mathbf{r} has unit length: $\sum_a |\alpha_a|^2 = 1$. These amplitudes $|\alpha_a|^2$ evidently give rise to a probability distribution on $\{0, 1\}^n$ - this is how we shall extract a classical (though stochastic) output from a quantum computation; indeed, we shall say that the probability that the computation *accepts* is the probability that the first bit is a 1 according to this probability distribution.

We remark that this curious method for “terminating” our quantum computation is, along with the rule of unitary evolution, directly borrowed from the model of quantum mechanics used to model the universe: this is the process of *measurement*. This is the second time we have justified a potentially surprising aspect of the model by direct appeal to the mathematical foundations of quantum mechanics. Indeed, one of the original motivations for defining (and yearning for) such a model is that it would allow us to “simulate” and study quantum systems of physical interest. Especially after hearing this explanation for the genesis of the model, it is natural to wonder if the new features of the model offer any new traction in the *classical* computational arena. We’ll see below is that the answer is yes: quantum computation can, in some cases, uncover exactly the combinatorial symmetries mentioned in Section 2.

We describe, in the next section, how these symmetries are related to the unitary evolution (and measurement) in the model above. To briefly advertise what comes ahead, however, we remark that with any finite group G one may inexorably associate a finite collection of “harmonic objects,” objects that precisely capture the symmetry structure of G . These objects, taken together, define a unitary basis change in the set of \mathbb{C} -valued functions on G . For many groups of interest, this unitary basis change can be efficiently carried out on a quantum computer, thus exposing the symmetries of objects on which G acts.

4 Group harmonics and applications

4.1 Representations of finite groups

Let G be a group acting on a set X ; as in Section 2, we may “extend” this group action to the set of functions $\mathbb{C}X \triangleq \{f : X \rightarrow \mathbb{C}\}$ by the rule

$$(\pi f)(x) = f(\pi^{-1}x).$$

This case, where we consider \mathbb{C} -valued functions on X (rather than the various “combinatorial” choices taken in Section 2) is especially attractive because we can bring to bear the tools of linear algebra to study these actions and their symmetries. To emphasize this connection, we observe that $\mathbb{C}X$ is a complex vector space with a distinguished basis: the delta functions $f_x : y \mapsto \delta_{xy}$ (where δ_{xy} is equal to one when $x = y$ and is zero otherwise). Notationally, if we let $|x\rangle$ denote the function f_x above, elements of $\mathbb{C}X$ can be written

$$\sum_{x \in X} a_x |x\rangle, \quad a_x \in \mathbb{C},$$

a convention we shall adopt throughout. Now we can view the action of G on $\mathbb{C}X$ as a family of linear operators; in particular, each g is associated with a linear operator $\rho_g : \mathbb{C}X \rightarrow \mathbb{C}X$ in such a way that $\rho_g \rho_h = \rho_{gh}$.

Note that with this linear algebraic view of group actions, $g \in S(f)$ (that is, the group element g is a symmetry of a function f) *exactly when f is an eigenvector with eigenvalue 1 of the map ρ_g* . This suggests a general study of the eigenvalues and invariant spaces of these group actions, a lifestyle known as the *representation theory* of finite groups. The general definition is the following:

Definition 1. *Let G be a finite group. A **representation** of G is a homomorphism $\rho : G \rightarrow \text{GL}(V)$, where $\text{GL}(V)$ is the collection of invertible linear operators on a (finite dimensional) \mathbb{C} -vector space V . The **dimension** of ρ , denoted d_ρ , is the dimension of V .*

Observe that ρ assigns to each group element g a linear operator $\rho(g)$ so that $\rho(g)\rho(h) = \rho(gh)$, just as in our permutation example above. Indeed, any action of G on a set X immediately induces a representation of G (on $\mathbb{C}X$). The representations of a group G offer a principled approach to the problem of understanding symmetries under G and its subgroups. We remark that when G is finite (as always, in this article), we may assume without loss of generality that representations come equipped with an inner product for which each $\rho(g)$ is unitary.

What renders the theory especially attractive is that a given finite group G possesses a finite number of atomic “irreducible” representations, in terms of which

all others can be expressed. To develop this decomposition machinery, we set down a few more definitions. We say that two representations $\rho_1 : G \rightarrow GL(V_1)$ and $\rho_2 : G \rightarrow GL(V_2)$ are *equivalent* when they are related by a (linear) isomorphism $E : V_1 \rightarrow V_2$; specifically, for every g we have $\rho_2(g) = E\rho_1(g)E^{-1}$. (If these ρ_i happened to be defined on the same space, they would be related by a change of basis.) The familiar notion of direct sum can be applied to sew together a pair of representations: if $\rho_1 : G \rightarrow GL(V_1)$ and $\rho_2 : G \rightarrow GL(V_2)$ are two representations of G , we may construct a new representation on $V_1 \oplus V_2$ with the action

$$\mathbf{v} \oplus \mathbf{w} \mapsto \rho_1(g)\mathbf{v} \oplus \rho_2(g)\mathbf{w};$$

we name this representation $\rho_1 \oplus \rho_2 : G \rightarrow GL(V_1 \oplus V_2)$. In matrix form, $\rho_1 \oplus \rho_2(g)$ can be realized as a block diagonal matrix, with each of the $\rho_i(g)$ forming one of two blocks.

Finally returning to the notion of irreducibility promised above, if $\rho : G \rightarrow GL(V)$ is a representation, we say that a subspace W of V is *invariant* when each $\rho(g)$ fixes W as a space. Of course, V and $\{0\}$ are always invariant. When these are the *only* invariant spaces, the representation is said to be *irreducible*. As mentioned above, a finite group G has a finite number of distinct irreducible representations up to equivalence; we let \hat{G} denote this finite set of (equivalence classes of) irreducible representations. Every other representation of G can be expressed as a direct sum of copies of representations in \hat{G} .

Our discussion of group actions in Section 2 leads us to naturally define another representation. Recall that any group G acts on itself by left-multiplication. Again, this allows G to act on the space $\mathbb{C}G = \{f : G \rightarrow \mathbb{C}\} = \text{span}(\{|g\rangle \mid g \in G\})$ via

$$g : |h\rangle \mapsto |gh\rangle.$$

The representation so defined is called the *left regular representation* of G . We remark that this representation is not irreducible (unless $|G| = 1$). In particular, note that the element $\sum_{g \in G} |g\rangle$ is fixed by *any* permutation, and in particular by those that correspond to left multiplication by elements of G . The linear span of this element is clearly a one-dimensional invariant subspace of $\mathbb{C}G$. This same argument applies to any permutation representation: evidently, every representation discussed in the article thus far is reducible! We remark that if the full symmetric group S_n acts on

$$\mathbb{C}\{1, \dots, n\} = \left\{ \sum_{i=1}^n a_i |i\rangle \mid a_i \in \mathbb{C} \right\}$$

by permuting the $|i\rangle$, the vector space perpendicular to the vector $\sum_i |i\rangle$ is an irreducible representation of dimension $n - 1$. (The inner product used to make sense of the word perpendicular is the one consistent with the $|i\rangle$ basis: $\langle i | j \rangle = \delta_{ij}$.)

This example above might suggest that the general problem of decomposing a representation into irreducible pieces (or, indeed, even identifying irreducible representations) is quite difficult and, potentially, delicate. This is true; however, there is a remarkable tool that considerably simplifies these questions. Given an irreducible ρ , its *character* at a group element g is defined to be $\chi_\rho(g) \triangleq \text{tr } \rho(g)$, the trace of $\rho(g)$. The important feature of these characters is that under the pairing

$$(\chi, \psi) = \frac{1}{|G|} \sum_g \chi(g) \psi(g^{-1}),$$

we find that for irreducible representations ρ and σ ,

$$(\chi_\rho, \chi_\sigma) = \begin{cases} 1 & \text{if } \rho \text{ and } \sigma \text{ are equivalent,} \\ 0 & \text{otherwise.} \end{cases}$$

As it is clear that for two representations β and γ , $\chi_{\beta \oplus \gamma}(g) = \chi_\beta(g) + \chi_\gamma(g)$, this pairing offers an immediate method for determining if a given representations is reducible and, moreover, decomposing a representation into irreducibles. Indeed, observe that if β is a representation and σ irreducible then $(\chi_\beta, \chi_\sigma)$ is precisely the number of times σ appears in the decomposition of β .

With these tools it is possible to show that the left regular representation $R : G \rightarrow GL(\mathbb{C}G)$ of a group G has a *generic* decomposition into irreducible representations. In particular, every irreducible representation ρ of G appears in R exactly d_ρ times. We write

$$R = \bigoplus_{\rho \in \hat{G}} \rho^{\oplus d_\rho} = \bigoplus_{\rho \in \hat{G}} \underbrace{\rho \oplus \cdots \oplus \rho}_{d_\rho}. \quad (2)$$

In particular, counting dimensions on both sides of this equation yields the equality $|G| = \sum_\rho d_\rho^2$.

The Fourier transform The definition of $\mathbb{C}G$ immediately distinguishes the *group basis* of $\mathbb{C}G$: the elements $\{|g\rangle\}$. On the other hand, the direct sum decomposition of Equation (2) above provides an alternate (and, in general, transverse) means of describing elements of $\mathbb{C}G$: in terms of their projections into the spaces $\rho^{\oplus d_\rho}$. If $f : G \rightarrow \mathbb{C}$ is a function and $\rho \in \hat{G}$, the *Fourier transform of f at ρ* is the linear operator

$$\hat{f}(\rho) = \sqrt{\frac{d_\rho}{|G|}} \sum_{g \in G} f(g) \rho(g^{-1}).$$

If we select a basis for the space V_ρ on which ρ operates, $\hat{f}(\rho)$ is realized as a $d_\rho \times d_\rho$ matrix of complex numbers. Taken over all $\rho \in \hat{G}$, the linear operators $\hat{f}(\rho)$

determine $\sum_{\rho} d_{\rho}^2 = |G|$ complex numbers—exactly the dimension of $\mathbb{C}G$. With the scaling factor $\sqrt{d_{\rho}/|G|}$ appearing above, this transformation from $f \in \mathbb{C}G$ to these matrices is unitary and actually corresponds to writing f in a basis consistent with the decomposition (2) of $\mathbb{C}G$ above. More prosaically, the Fourier transform is a unitary map from the group basis $\{|g\rangle\}$ to the basis

$$\{|\rho, i, j\rangle : \rho \in \hat{G} \text{ and } 1 \leq i, j \leq d_{\rho}\}, \quad (3)$$

where the i and j entries correspond to the rows and columns of the matrices $\hat{f}(\rho)$ above.

The reason for this long digression, and the critical fact that weds quantum computing and representation theory, is that the Fourier transform, as described above, can be efficiently computed on a quantum computer for many groups of interest [4, 15]. In this context, the unitary Fourier transform is called the *quantum Fourier transform*, and is a key ingredient in the algorithms we discuss in the sequel.

4.2 Looking for hidden symmetries in coset states

We now discuss a special case of the symmetry finding problem presented in Section 2. Suppose we are given oracle access to a function $f : G \rightarrow S$ and a promise that f is a *transversal* of some unknown subgroup H of G in the sense that

$$f(hx) = f(x) \Leftrightarrow h \in H.$$

For concreteness, we will assume that S is the set of integers $\{1, 2, \dots, n\}$ for a suitably large n . As f is constant and distinct on each coset of H , the problem of determining $S(f)$ is precisely the problem of determining the “hidden” subgroup H . This problem is aptly titled the *Hidden Subgroup Problem* (HSP). The standard method of *Fourier sampling* [5] for solving such problems on a quantum computer proceeds as follows.

Our first step is to prepare the “uniform superposition over G ,” i. e., the state

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle,$$

and then “evaluate” the oracle function f on this state.³ The resulting state of the system is

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle |f(g)\rangle. \quad (4)$$

³To make this a unitary operation, we actually need some additional “empty” workspace. The quantum oracle is the map $U : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle$, where \oplus denotes the (clearly unitary) operation of bitwise addition. In our case, $x = \sum_{g \in G} |g\rangle$ and $y = |00 \cdots 0\rangle$, where the register containing y is large enough to accommodate the possible values of f .

Now, we wish to “measure the second register” (that is, the one containing $|f(g)\rangle$), thereby isolating the elements of G where f takes a particular value $k \in S$. In general, a quantum measurement on a vector space V is described by an orthogonal decomposition $V = W_1 \oplus \dots \oplus W_k$; when such a measurement is applied to a vector \mathbf{v} , it results in the measured value i with probability $\|\Pi_i(\mathbf{v})\|^2$, where Π_i projects onto the subspace W_i . Since we are supposing that \mathbf{v} has length 1, this yields a probability distribution on i . The state of our system after the measurement will be the (renormalized) projection

$$\frac{\Pi_i \mathbf{v}}{\|\Pi_i \mathbf{v}\|_2} \in W_i .$$

In our case, the state (4) lives in the space $V = \mathbb{C}G \otimes \mathbb{C}S$. We have specific bases in mind, too: the basis for the space $\mathbb{C}G$ is $\{|g\rangle : g \in G\}$, while the basis for $\mathbb{C}S$ is $\{|1\rangle, |2\rangle, \dots, |n\rangle\}$. Our desired measurement corresponds to the orthogonal decomposition

$$V = [\mathbb{C}G \otimes |1\rangle] \oplus [\mathbb{C}G \otimes |2\rangle] \oplus \dots \oplus [\mathbb{C}G \otimes |n\rangle] .$$

The result of this measurement will be the (renormalized) projection of the state (4) to one of the subspaces $\mathbb{C}G \otimes |k\rangle$; that is, it will be a uniform superposition over all elements of G where f takes the particular value k . The set of such group elements is, by the conditions we imposed on f , a coset of some hidden subgroup H . We are thus left in the “coset state”

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle , \quad (5)$$

where c is an unknown but, fortunately, random element of G . Next, we apply the Quantum Fourier Transform from Section 4.1. Let us first view the above state as the scaled characteristic function

$$\psi(g) = \begin{cases} 1/\sqrt{|H|} & \text{if } g \in cH, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Recall that the Fourier transform of this function, at an irreducible $\rho \in \hat{G}$, is a $d_\rho \times d_\rho$ matrix denoted by $\hat{\psi}(\rho)$. Thus, the Quantum Fourier Transform of the above state will be indexed by $|\hat{G}|$ -many representations ρ , and d_ρ^2 -many entries for each such ρ . Concretely, the new state is given by

$$\sum_{\rho, i, j} \hat{\psi}(\rho)_{ij} |\rho, i, j\rangle . \quad (7)$$

We think of this state as being indexed by three registers: the first register specifies the representation ρ , the second register specifies the “row” i , and the third register specifies the “column” j . Our task now is to perform some kind of measurement on the above state, and attempt to determine the hidden subgroup H based on this measurement. For instance, we may simply measure the first register - resulting in a particular representation σ with probability

$$\sum_{i,j} |\hat{\psi}(\sigma)_{ij}|^2 = \|\hat{\psi}(\sigma)\|_2^2.$$

This method is called *weak Fourier sampling*. Alternatively, we may apply *strong Fourier sampling*, which will completely measure all of the registers. The result of strong sampling is a matrix entry $\hat{\psi}(\tau)_{kl}$, occurring with probability $|\hat{\psi}(\tau)_{kl}|^2$. (Note that this measurement depends on the basis in which σ is expressed by our Fourier transform.)

In the coming sections, we will describe how one can efficiently reconstruct normal subgroups via the weak Fourier sampling method discussed above. As any subgroup of an abelian group is normal, this will allow us to completely solve the HSP on abelian groups. In particular, this will imply that the period finding and order finding applications discussed in Section 2 have efficient solutions on a quantum computer.

Some progress has also been made in the arena of non-normal subgroups. For instance, Hallgren, Ip, and Van Dam [22] constructed an efficient quantum algorithm for the hidden shifts of the quadratic character discussed in Section 2 (their algorithm actually departs from the Fourier sampling framework above, directly using the oracle as “phase information”; the affine group formulation of the problem presented in Section 2 was solved by strong sampling in [16]). Kuperberg [14] devised a “sieve” algorithm to solve the hidden subgroup problem in the dihedral groups D_n with subexponential running time $2^{O(\sqrt{n})}$. (We remark that no classical algorithm can have running time $2^{o(n)}$.) We will briefly discuss these algorithms in Section 5.2.2. Finally, in Section 5.3, we will discuss an approach developed by Bacon, Childs, and van Dam [3] that has given rise to efficient algorithms for various semi-direct product groups, including the Heisenberg groups $\mathbb{Z}_p \ltimes \mathbb{Z}_p^2$.

Sadly, our final example of Graph Automorphism, where we seek to find certain hidden subgroups of the symmetric group S_n , still defies the community’s attempts at devising efficient algorithms. In fact, several negative results have shown that solving the HSP using coset states in the symmetric groups (and some other group families) *requires* so-called *multiregister* Fourier sampling. This entails sampling several coset states $\psi_1, \psi_2, \dots, \psi_k$ and then performing some kind of (non-separable) measurement on the tensor product $\psi_1 \otimes \psi_2 \otimes \dots \otimes \psi_k$ of the k registers. In the case of the symmetric group, we must use at least $\Omega(\log |G|)$ many samples [10, 18]. As we will later discuss, however, all is not lost even in

this area. Indeed, even for some group families to which these negative results apply, there are quantum algorithms which perform better than any possible classical algorithm.

5 Algorithmic progress

5.1 Reconstructing normal subgroups

In this section, we will show how to resolve the HSP efficiently in the case where the hidden subgroup is normal. The results of this section are due to Hallgren, Russell, and Ta-shma [11]. As every subgroup of an abelian group is normal, our discussion will include the HSP on abelian groups as a special case. Recall from Section 4.1 that the Quantum Fourier Transform is the basis change operator that maps the group basis to the Fourier basis; that is, it rewrites a function $\psi \in \mathbb{C}G$ (indexed by group elements) as another function $\hat{\psi}$ (indexed by representations, rows, and columns). As discussed in Section 4.2, performing “weak Fourier sampling” on $\hat{\psi}$ will then measure a particular representation $\rho \in \hat{G}$, while ignoring the rows and the columns. The probability of observing ρ is equal to the norm $\|\hat{\psi}(\rho)\|^2$ of the Fourier transform of ψ at ρ .

We first let ψ be the “coset state” (5) from Section 4.2; it’s easy to show that our analysis will not depend on the value of c , and so we assume that $c = 1$. Then ψ takes the value $1/\sqrt{|H|}$ on the hidden subgroup H of G , and is zero elsewhere. The Fourier transform of ψ at an irreducible ρ is then given by

$$\hat{\psi}(\rho) = \sqrt{\frac{d_\rho}{|G|}} \sum_{h \in G} \psi(h) \rho(h) = \sqrt{\frac{d_\rho |H|}{|G|}} \Pi_H^\rho$$

where $\Pi_H^\rho \triangleq |H|^{-1} \sum_{h \in H} \rho(h)$. It’s easy to check that $(\Pi_H^\rho)^2 = \Pi_H^\rho$, i.e., Π_H^ρ is a projection operator. The probability of measuring a particular ρ using weak sampling is then

$$P_H(\rho) \triangleq \|\hat{\psi}(\rho)\|^2 = \frac{d_\rho |H|}{|G|} \text{rk } \Pi_H^\rho. \quad (8)$$

This distribution takes a particularly nice form when H is a normal subgroup. Let H^\perp denote the set of representations from \hat{G} whose kernel contains H . The representations in H^\perp (that is, representations which are trivial on H) are precisely the same as the representations of the quotient group G/H .

Lemma 1. *Let H be a normal subgroup of G . If ρ is an element of H^\perp , then the probability of observing ρ is equal to $d_\rho^2 |H|/|G|$; otherwise, the probability of observing ρ is zero.*

Proof. If $\rho \in H^\perp$, then for every $h \in H$, $\rho(h)$ is the identity operator. Since Π_H^ρ is simply the average of these, it is also equal to the identity operator, and thus has full rank. Hence

$$P_H(\rho) = \frac{d_\rho \mathbf{rk} \Pi_H^\rho |H|}{|G|} = \frac{d_\rho^2 |H|}{|G|}.$$

Now, if we add up the contributions to P_H from the representations in H^\perp , we have

$$\sum_{\rho \in H^\perp} P_H(\rho) = \sum_{\rho \in H^\perp} \frac{d_\rho^2 |H|}{|G|} = \frac{|H|}{|G|} \cdot \sum_{\rho \in G/H} d_\rho^2 = \frac{|H|}{|G|} \cdot |G/H| = 1.$$

Hence the representations outside H^\perp must contribute zero probability. \square

We now show that one can reconstruct a normal subgroup H of an arbitrary finite group G in polynomial time, simply by sampling from the distribution P_H . Once enough samples have been produced, we then reconstruct the subgroup by intersecting the kernels of our sampled representations.

Theorem 1. *Let H be a normal subgroup of G . Let $\sigma_1, \dots, \sigma_s$ be independent random variables sampled from the distribution P_H , with $s = c \log |G|$. Then*

$$\Pr \left[H \neq \bigcap_i \ker \sigma_i \right] \leq e^{-\frac{(c-2)^2}{2c} \log |G|}.$$

Proof. Let $N_0 = G$, and let $N_i = \bigcap_{j=1}^i \ker \sigma_j$ be the intersection of the kernels sampled thus far. As they are intersections of normal subgroups, each N_i is normal in G . By Lemma 1, we cannot observe any representations except those whose kernel contains H , and thus $H \subseteq \ker \sigma_i$ for every i . Hence

$$H \subseteq N_s \subseteq N_{s-1} \subseteq \dots \subseteq N_0 = G.$$

Our theorem rests on the fact that, with each new sample, we will make progress along the above chain with probability at least $1/2$; reaching H will thus take roughly $\log |G|$ many steps. We thus claim that if $N_i \neq H$, then $\Pr_{\sigma_{i+1} \in P_H} [N_{i+1} = N_i] \leq 1/2$. Indeed, by making use of Lemma 1 again, we see that

$$\begin{aligned} \Pr[N_{i+1} = N_i] &= \Pr[N_i \subseteq \ker \sigma] = \sum_{\rho \in N_i^\perp} d_\rho^2 \frac{|H|}{|G|} \\ &= |G/N_i| \cdot \frac{|H|}{|G|} = \frac{|H|}{|N_i|} \leq 1/2. \end{aligned}$$

To complete the proof, we will need to apply a Chernoff bound. Let X_i be indicator random variables defined by $X_i = 1$ if $N_i = H$ or $N_i \neq N_{i-1}$ and zero otherwise.

While the X_i are not necessarily independent, our claim above showed that $\Pr[X_i = 0 | \sigma_1, \dots, \sigma_{i-1}] \leq 1/2$. We can thus define new independent random variables Y_i satisfying $\Pr[Y_i = 0] = 1/2$ and $\sum Y_i \leq \sum X_i$. By the Chernoff bound given in [11], $\Pr[\sum_i Y_i \leq (s - a)/2] < e^{-a^2/2s}$. This implies that

$$\Pr\left[\sum_i X_i \leq c \log |G|\right] < e^{-(c-2)^2 \log |G|/2c}.$$

Thus $\sum_i X_i \geq \log |G|$ with overwhelming probability; but in this case, $N_i \subsetneq N_{i-1}$ for every i , and hence $N_s = H$. \square

We remark that the problem of reconstructing a normal subgroup by computing an intersection of representation kernels may, for general groups, be a very difficult problem. However, with the aid of a classical machine we can easily perform this task on abelian groups. By the structure theorem for finite abelian groups, we need only discuss the cyclic groups $G = \mathbb{Z}_n$. Now, suppose $\alpha \in \mathbb{Z}_n$ is a generator for the hidden subgroup in question, and that $\chi_{g_1}, \chi_{g_2}, \dots, \chi_{g_s}$ are the sampled representations (that is, characters) of \mathbb{Z}_n . By Theorem 1 above, we know that

$$\bigcap_i \ker \chi_{g_i} = \langle \alpha \rangle.$$

We claim that the left hand side is simply $\ker \chi_g$ where $g = \gcd(g_1, g_2, \dots, g_s)$. If $x \in \bigcap_i \ker \chi_{g_i}$, then

$$\chi_{g_i}(x) = e^{\frac{2\pi i g_i x}{n}} = 1,$$

and hence $g_i x \equiv 0 \pmod n$, for every i . By taking linear combinations, we have $gx \equiv 0 \pmod n$, and thus $x \in \ker \chi_g$. On the other hand, if x satisfies $gx \equiv 0 \pmod n$, then certainly $g_i x \equiv 0 \pmod n$ for every i , since the g_i are integer multiples of g . We have thus shown that $\langle \alpha \rangle = \ker \chi_g$. Along with Theorem 1, this proves that we can reconstruct hidden subgroups of cyclic groups efficiently by weak Fourier sampling a sufficient number of characters, and then computing their gcd using a classical machine.

5.2 Sieve Algorithms

5.2.1 Weak Sampling Fails

As we have shown, weak Fourier sampling (that is, measuring the representation name only) allows for the reconstruction of normal subgroups and, in particular, arbitrary subgroups of abelian groups. For some groups, however, this method cannot solve the HSP efficiently [11]. The following proposition, due to Alagic,

Moore and Russell [1] shows that certain subgroups of *product groups* are indistinguishable using weak Fourier sampling alone. This is an example of a family of groups where *strong Fourier sampling* (that is, measuring the rows and columns in addition to the representation name) is necessary for resolving the HSP.

Proposition 1. *Let G be a group with an involution $\mu \notin Z(G)$, and let $H = \{1, m\} \leq G^n$ where m is chosen uniformly at random from the conjugacy class $[(\mu, \dots, \mu)]$. Then the total variation distance between the weak Fourier sampling distributions (8) for the subgroups H and $\{1\}$ is at most $2^{-n/2}$.*

Proof. We upper bound the total variation distance between the distributions in question:

$$\begin{aligned} \|P_{\{1\}} - P_H\|_1 &= \sum_{\rho \in \widehat{G}^n} \left| \frac{d_\rho}{|G|^n} \mathbf{rk} \Pi_{\{1\}}^\rho - \frac{2d_\rho}{|G|^n} \mathbf{rk} \Pi_H^\rho \right| \\ &= \sum_{\rho \in \widehat{G}^n} \left| \frac{d_\rho}{|G|^n} \mathbf{rk} \Pi_{\{1\}}^\rho - \frac{2d_\rho}{|G|^n} \mathbf{tr} \left[\frac{\mathbb{1}_\rho + \rho(m)}{2} \right] \right| \\ &= \frac{1}{|G|^n} \sum_{\rho \in \widehat{G}^n} \left| d_\rho^2 - d_\rho^2 \left(1 + \frac{\chi_\rho(m)}{d_\rho} \right) \right| \\ &= \frac{1}{|G|^n} \sum_{\rho \in \widehat{G}^n} |d_\rho \cdot \chi_\rho(m)|. \end{aligned}$$

Viewing the last line as an inner product, we apply Cauchy-Schwarz to get

$$\begin{aligned} \|P_{\{1\}} - P_H\|_1 &\leq \frac{1}{|G|^n} \left(\sum_\rho d_\rho^2 \right)^{1/2} \left(\sum_\rho \chi_\rho(m) \chi_\rho^*(m) \right)^{1/2} \\ &= \frac{1}{|G|^{n/2}} \left(\sum_{\rho \in \widehat{G}^n} \chi_\rho(m) \chi_\rho^*(m) \right)^{1/2} \\ &= \frac{1}{|G|^{n/2}} \left(\sum_{\rho \in \widehat{G}} \chi_\rho(\mu) \chi_\rho^*(\mu) \right)^{n/2}. \end{aligned}$$

Here we have used the fact that the character of an irreducible G^n -representation is an n -fold product of characters of irreducible G -representations. The term $\sum_\rho \chi_\rho(\mu) \chi_\rho^*(\mu)$ is in fact the character of the so-called *conjugation representation* of G ; this is the representation defined on $\mathbb{C}G$ by linearly extending the rule $g \cdot x \mapsto gxg^{-1}$. The character of this representation, evaluated at μ , is exactly the number of fixed points of the conjugation action of μ on G , i.e., the size of

the centralizer C_μ . As μ is not in the center, C_μ is a proper subgroup, and hence $\chi_C(\mu) \leq |G|/2$, which completes the proof. \square

5.2.2 Sieve Algorithm Sketch

Recent negative results have shown that even strong Fourier sampling is insufficient to efficiently resolve the HSP on certain highly nonabelian groups. For instance, Hallgren, Moore, Rötteler, Russell and Sen [10] showed that multiregister Fourier sampling over $\Omega(\log |G|)$ registers is required to efficiently distinguish subgroups of certain families of groups; these families include the symmetric groups (presumably critical for the Graph Isomorphism application discussed in Section 2.3), and the nonabelian direct product groups. Despite this, we can still do provably better than classically for certain groups to which these negative results apply. Indeed, using a *representation sieve* idea pioneered by Kuperberg [14] for the HSP on dihedral groups, a subexponential-time sieve algorithm of Alagic, Moore and Russell [2] resolves the HSP on direct product groups. These are groups of the form $G = K^n$ where K is nonabelian and of constant size. We now discuss, in brief, the central idea behind algorithms based on Kuperberg’s representation sieve.

Recall that the result of weak Fourier sampling is essentially a “state vector” lying in an irreducible representation of our group G . Now, if we perform weak sampling twice, we will have two state vectors lying in irreducibles which we will label ρ and σ . The representations ρ and σ naturally define another representation $\rho \otimes \sigma$ on the tensor product $V_\rho \otimes V_\sigma$ of their respective spaces. This is done by the “diagonal action”:

$$[\rho \otimes \sigma](g) \triangleq \rho(g) \otimes \sigma(g).$$

In general, this new representation is not irreducible; it does, however, have an irreducible decomposition

$$\rho \otimes \sigma \cong \bigoplus_j \tau_j.$$

The essential ingredient of the aforementioned sieve algorithms is this: given state vectors in ρ and σ , we can use a modified form of weak sampling to produce a state vector in one of the constituents τ_j of $\rho \otimes \sigma$. Moreover, our knowledge of the representation theory of the underlying group gives us a nice picture of which τ_j our new state vector may lie in; for instance, if the hidden subgroup is actually trivial, then the resultant state will lie in τ_j with probability $d_{\tau_j}/d_{\rho \otimes \sigma} = d_{\tau_j}/(d_\rho d_\sigma)$. A sieve algorithm may thus proceed as follows:

1. use weak Fourier sampling to generate a large (but still subexponential) collection of states, each lying in some irreducible representation;

2. cleverly pair up the states generated above, and sample new states from the tensor products of these pairs, now lying in representations of *smaller dimension* (on average);
3. repeat (2) until you generate states lying in one-dimensional representations.

Once we have generated sufficiently many states in one-dimensional representations, we can then reconstruct the hidden subgroup; this is roughly analogous to the abelian case, where all irreducibles are one-dimensional. We remark that the repeated applications of step (2) constitute a (rather complicated) multiregister measurement over all of the registers containing the coset states initially sampled in step (1).

5.3 The pretty-good measurement

Recall from the discussion preceding Equation (5) that typical approaches to the hidden subgroup problem involve production of *coset states* of the form

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle ,$$

where c is an independently chosen element in G . This “state” is really a classical probability distribution over the sorts of quantum states we have described in the article thus far. Such objects arise anytime a measurement takes place, and are called *mixed states*. (The quantum states we have discussed previously, unit length vectors, are the special case with a trivial probability distribution and are called *pure states* when it is useful to emphasize the distinction.)

It turns out that there are group/subgroup combinations where this state contains very little information about H [18]. On the other hand, it is known that polynomially many (in $\log |G|$) of these states do, at least in principle, completely determine the subgroup H [9]. The problem, from a quantum computational perspective, is to discover an efficient means of extracting this information from the mixed state.

The problem of “identifying” a given state from a known list of possibilities has a long history in quantum mechanics, though most previous work was less concerned with computational efficiency than it was with the information-theoretic aspects of the problem: that is, whether or not measurements even existed to tease apart various families of states.

A remarkable discovery of Bacon, Childs, and van Dam [3] is that a *generic* measurement, the “pretty-good measurement,” that one can define for any fixed family of mixed states can actually be efficiently implemented in some cases of

interest for the hidden subgroup problem. Furthermore, they show that these measurements are powerful enough to distinguish the subgroups, giving rise to efficient solutions to the hidden subgroup problem for specially structured groups. (It was later shown that with sufficiently many copies of the mixed state above, the pretty good measurement is always a rich enough measurement to distinguish hidden subgroups [12].)

6 Conclusion

The hidden subgroup problem (and the symmetry finding problem, in general) unite quantum computation, a handful of classical computational problems, and aspects of the representation theory of finite groups. We indicated, in Section 5, the principal algorithmic techniques that have been developed for attacking this problem in general: Fourier sampling, sieve methods, and implementation of the pretty good measurement. In all of these cases, the essential insight on which the algorithm depends illuminates the connection between coset states and the algebraic structure of the group's representation theory. Despite this progress, the guiding problem in the area of nonabelian hidden subgroup problems—Graph Isomorphism—appears to be quite immune to known techniques [17, 19]. Happily, the area has seen rapid development in the last 5 years on the algorithmic, information-theoretic, and representation-theoretic fronts; we can be sure that the area and its hallmark problem have more secrets to uncover.

References

- [1] Gorjan Alagic, Cristopher Moore, and Alexander Russell. Strong Fourier sampling fails over G^n . Technical Report quant-ph/0511054, arXiv.org e-Print archive, 2005.
- [2] Gorjan Alagic, Cristopher Moore, and Alexander Russell. Quantum algorithms for Simon's problem over general groups. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proc. of SODA 2007*, pages 1217–1224. SIAM, 2007.
- [3] Dave Bacon, Andrew M. Childs, and Wim van Dam. From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups. In *Proc. of FOCS 2005* [13], pages 469–478.
- [4] Robert Beals. Quantum computation of fourier transforms over symmetric groups. In *Proc. of STOC '97*, pages 48–53, New York, NY, USA, 1997. ACM Press.
- [5] Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.

- [6] Andrew Childs, Leonard Schulman, and Umesh Vazirani. Quantum algorithms for hidden nonlinear structures. Technical Report quant-ph/0705.2784, arXiv.org e-Print archive, 2007.
- [7] Thomas Decker, Jan Draisma, and Pawel Wocjan. Quantum algorithm for identifying hidden polynomial function graphs. Technical Report quant-ph/0706.1219, arXiv.org e-Print archive, 2007.
- [8] David Dummit and Richard Foote. *Abstract Algebra*. John Wiley & Sons, Hoboken, NJ, third edition, 2004.
- [9] Mark Ettinger, Peter Høyer, and Emanuel Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Inf. Process. Lett.*, 91(1):43–48, 2004.
- [10] Sean Hallgren, Cristopher Moore, Martin Rötteler, Alexander Russell, and Pranab Sen. Limitations of quantum coset states for graph isomorphism. In Jon M. Kleinberg, editor, *Proc. of STOC 2006*, pages 604–617. ACM, 2006.
- [11] Sean Hallgren, Alexander Russell, and Amnon Ta-Shma. Normal subgroup reconstruction and quantum computation using group representations. In *Proc. of STOC 2000*, pages 627–635. ACM, 2000.
- [12] Masahito Hayashi, Akinori Kawachi, and Hirotada Kobayashi. Quantum measurements for hidden subgroup problems with optimal sample complexity. Technical Report quant-ph/0604174, arXiv.org e-Print archive, 2006.
- [13] IEEE. *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*. IEEE, 2005.
- [14] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005.
- [15] Cristopher Moore, Daniel Rockmore, and Alexander Russell. Generic quantum fourier transforms. In *Proc. of SODA '04*, pages 778–787, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
- [16] Cristopher Moore, Daniel N. Rockmore, Alexander Russell, and Leonard J. Schulman. The power of basis selection in fourier sampling: Hidden subgroup problems in affine groups. In J. Ian Munro, editor, *Proc. of SODA 2004*, pages 1113–1122. SIAM, 2004.
- [17] Cristopher Moore, Alexander Russell, and Leonard Schulman. Tight results on multi-register fourier sampling: Quantum measurements for graph isomorphism require entanglement. Technical Report quant-ph/0511149, arXiv.org e-Print archive, 2006.
- [18] Cristopher Moore, Alexander Russell, and Leonard J. Schulman. The symmetric group defies strong Fourier sampling. In *Proc. of FOCS 2005* [13], pages 479–490.
- [19] Cristopher Moore, Alexander Russell, and Piotr Sniady. On the impossibility of a quantum sieve algorithm for graph isomorphism. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 536–545, New York, NY, USA, 2007. ACM Press.

- [20] Peter W. Shor. Polynomial time algorithms for discrete logarithms and factoring on a quantum computer. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *Proc. of ANTS 1994*, volume 877 of *Lecture Notes in Computer Science*, page 289. Springer, 1994.
- [21] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, 1997.
- [22] Wim van Dam, Sean Hallgren, and Lawrence Ip. Quantum algorithms for some hidden shift problems. In *Proc. of SODA 2003*, pages 489–498. ACM, 2003.

THE CONCURRENCY COLUMN

BY

LUCA ACETO

BRICS, Department of Computer Science
Aalborg University, 9220 Aalborg Ø, Denmark
luca@cs.auc.dk, <http://www.cs.auc.dk/~luca/BEATCS>

Concurrency-theoretic concepts and techniques have recently been used in the study of dynamic web data and web services. This installment of the concurrency column provides a survey of work in this area, with emphasis on the approach taken by the author, Sergio Maffei, and Philippa Gardner. I trust that readers will find it interesting and inspiring.

The CONCUR conference series has now reached the “age of reason”. The 18th conference in this series, CONCUR 2007, was beautifully organized in Lisbon by Luis Caires and Vasco Vasconcelos. I enjoyed it immensely. The organizers and their support team deserve a word of thanks from the whole community for holding such a successful event, both scientifically and socially. The next CONCUR will be held in Toronto, Canada.

DYNAMIC WEB DATA AND PROCESS CALCULI*

Sergio Maffei[†]
Imperial College London,
University of California at Santa Cruz.

Abstract

*This paper is an abridged and revised version of [18].

[†]Partially supported by EPSRC grant EP/E044956/1.

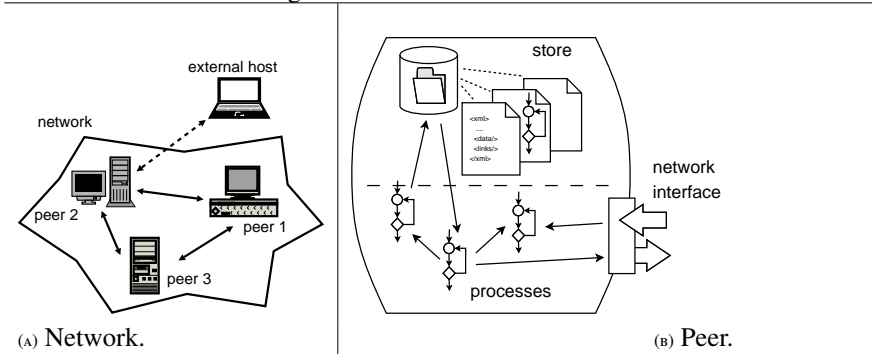
Peer-to-peer systems exchanging dynamic documents through web services are a simple and effective platform for data integration on the Web. Dynamic documents contain both data and references to external sources in the form of links, calls to web services, or coordination scripts. XML standards and industrial platforms for Web services provide a technological basis for building such systems. We use the Core $Xd\pi$ calculus to argue that process calculi provide a useful framework for studying and understanding their formal properties. In particular, we consider a natural notion of semantic equivalence for Core $Xd\pi$, and we use it to compare several distributed query patterns.

1 Introduction

The World Wide Web is a global network, used in daily activities to find information, communicate ideas, conduct business and carry out distributed computations. In order to fully exploit the potential of this massive network, there is a need for scalable mechanisms to organize and manipulate the available information. Peer-to-peer architectures help us to deal with the issue of scalability, and technologies such as XML and Web services facilitate the development of distributed applications. XML [32] is a standardized data model, used to represent uniformly documents containing tagged information not adhering to a fixed structure. Web services [34] are Web sites which are designed to be used by applications rather than humans. Web service inter-operability is facilitated by the use of XML for data representation and of related standards for service invocation, description and discovery (SOAP, WSDL, UDDI [35, 33, 31]).

Data integration on the Web constitutes a challenging application for these technologies, because of the extreme heterogeneity of data sources involved and the complexity of communication patterns which can arise. For example, translating a declarative request for networked data into a low-level execution plan may involve recursively invoking other declarative requests on different Web sites. Inspired by this problem, the $Xd\pi$ calculus [11] models peer-to-peer architectures for exchanging Web data, schematically represented in Figure 1. Each *network* is composed by a variable number of interconnected peers, all sharing a similar internal structure, and each one identified by a unique name (Figure 1_(A)). Peers share a common messaging protocol where the name of a peer is assumed to coincide with its network address: at this level of abstraction there are no restrictions to connectivity due to network domains or firewalls. Networks are *open* in the sense that it is always possible to add new peers or learn dynamically about their existence, and external hosts may participate in the data exchange although typically playing a limited role. Each peer (schematized in Figure 1_(B)) acts both as

Figure 1: Reference architecture



a provider and consumer of information. It contains a data repository, an internal working space where processes carry out local computations, and a network interface providing remote communication and services to other peers. Processes can communicate locally with each other, query and update the local repository and, when the architecture supports mobility, migrate to other peers to continue execution. Repositories present a semi-structured view of their data to the processes. Typically, the data contains enough meta-information about its structure to make it possible to write expressive queries.

Data is not completely static. It often contains references to other data and services, in the form of URLs, queries, or *scripts*. A script is some code describing a process which can be interpreted by the working space to add dynamic content to documents. We refer to such data as *dynamic Web data*. The World Wide Web itself is a very general example of architecture for dynamic Web data. Servers use the HTTP protocol to interact with each other, either requesting or providing information. HTML pages may contain hyperlinks, forms and client-side scripts, which provide dynamic behaviour. Web clients running a browser can be considered as “external hosts” which participate to a smaller degree in the exchange of information, by mostly consuming rather than providing data.

A more specific example comes from the database world. The Active XML [30, 3] system for data integration (AXML for short) consists of a network of peers, each containing a repository of documents and a set of service definitions. AXML service definitions typically consist of queries and updates on the local repository, but in general can consist of arbitrary Web services providing an interface to hosts external to the AXML system. AXML documents are XML documents which can include special tags representing calls to services on other peers. The parameters to these service calls can be local queries (path expressions) or AXML data, hence service calls can be nested.

Besides Web browsers and AXML, a large class of other Web applications (such as file-sharing programs, personal Web portals, online bibliographic databases, etc.) can be seen as instances of the reference architecture given above, each with its own particular features and restrictions. The problems that these architectures have to address, in order to be practically useful, are varied. Firstly, it is well-known that interaction between concurrent processes is difficult to regulate. In the case of Web services, this problem is complicated by the difficulty in maintaining state across different Web service invocations, and requires the study of choreography techniques. Secondly, a major concern for systems dealing with dynamic Web data is security. Depending on the application domain, it may be crucial to have control over for example data integrity, confidentiality, or access control. The formal study of security properties needs to be grounded on a rigorous model of these architectures, and process algebraic techniques have been shown to be particularly suited for the task [1, 13, 10].

1.1 The process algebraic approach

We want to show that process algebraic techniques are well-suited to represent and study formal properties of dynamic web data. In order to do so, we present Core $Xd\pi$, an explicitly located dialect of $Xd\pi$. In Core $Xd\pi$, the agents running in the working space of each peer are modelled by a parallel composition of located π -like processes, and the XML data repositories of the peers are modelled by a store, mapping each peer name (a *location*) with an ordered, edge-labelled tree. We regard processes as agents with a simple set of functionalities: they communicate with each other, query and update the local repository, and migrate to other peers to continue execution. Process descriptions, in the form of scripts, may be included in documents and executed by other processes. The definition of Core $Xd\pi$ is parametric with respect to the choice of a specific query and update language.

The combination of Web services and scripted processes provides many alternative patterns for exchanging information, and a theory of semantic equivalence for dynamic web data can be useful to show, for example, that some complex data-exchange protocol corresponds to some intuitive behaviour. Motivated by this consideration, we present network equivalence for Core $Xd\pi$, which states when two group of peers can be considered indistinguishable with respect to their attempts to interact with their local stores. Our objective is to define an equivalence relation on processes such that, when we place equivalent processes in the same context, we obtain equivalent networks.

As an application, we study communication patterns used by servers in distributed query systems to answer queries from clients. We show that some existing patterns [26] can be combined together obtaining a flexible infrastruc-

Figure 2: Syntax

$T, S ::= E \mid T \mid \emptyset \mid x$	$E, F ::= a[V] \mid x$	
$U, V ::= T \mid p @ l \mid \langle A \rangle$	$l ::= l \mid x$	$p ::= p \mid x$
$A ::= A \mid x$		
$a, b, c \in \mathcal{E}$ (EDGE LABELS)	$l, m \in \mathcal{L}$ (countably infinite)	(LOCATIONS NAMES)
$p, q \in \mathcal{Q}$ (QUERIES)	$x, y, z \in \mathcal{V}$ (VARIABLES)	$E, F \in \mathcal{B} \stackrel{\text{def}}{=} \{E : fv(E) = \emptyset\}$ (BRANCHES)
$U, V \in \mathcal{D} \stackrel{\text{def}}{=} \{V : fv(V) = \emptyset\}$ (DATA)	$T, S \in \mathcal{T} \stackrel{\text{def}}{=} \{T : fv(T) = \emptyset\}$ (TREES)	
$D, B \in \mathcal{S} \stackrel{\text{def}}{=} \mathcal{L} \rightarrow \mathcal{T}$		(STORES)
$N, M \in \mathcal{N} \stackrel{\text{def}}{=} \{(D, P) : D \in \mathcal{S}, P \in \mathcal{P}, dom(P) \subseteq dom(D)\}$		(NETWORKS)
$P, Q, R ::= \mathbf{0} \mid P \mid P \mid (\nu c)P \mid \overline{l \cdot c}(\tilde{v}) \mid l \cdot c(\tilde{\pi}).P \mid !l \cdot c(\tilde{\pi}).P \mid l \cdot \text{go } m.P \mid A \circ \langle l, \tilde{v} \rangle$		
$\mid l \cdot \text{req}_p(c)$		
$a, b, c ::= c \mid c \mid x$	$\nu ::= c \mid l \mid p \mid A \mid E \mid T$	
$a, b, c \in C_p$ (countably infinite)		(PRIVATE CHANNELS)
$a, b, c \in C_s$		(SERVICE CHANNELS)
$v, u \in \mathcal{U} \stackrel{\text{def}}{=} \{v : fv(v) = \emptyset\}$		(VALUES)
$\pi \in \mathcal{K} \stackrel{\text{def}}{=} \mathcal{V} \cup \{V : distinct(V)\}$		(PATTERNS)
$P, Q, R \in \mathcal{P} \stackrel{\text{def}}{=} \{P : fv(P) = \emptyset \text{ and } wf(P)\}$		(PROCESSES)
$A \in \mathcal{A} \stackrel{\text{def}}{=} \{(x, \tilde{\pi})P : \begin{array}{l} fv(P) = \emptyset, fv(P) \subseteq fv(x, \tilde{\pi}), \\ distinct(x, \tilde{\pi}), dom(P) = \{x\} \end{array}\}$		(SCRIPTS)

ture which is provably equivalent to an intuitive specification of the intended behaviour. By exploiting process migration, we also propose a new communication pattern, and show that it is behaviourally equivalent to a naive, less efficient one.

2 A process calculus for dynamic Web data

In this Section we present the syntax and semantics for Core $Xd\pi$, omitting non-essential definitions (see [17, 18] for a complete exposition).

Data and queries. We represent semi-structured data using ordered labelled trees. The formal definition is given in Figure 2 (we use italic bold letters for terms T which can contain variables, and plain italic letters for closed terms T). We represent a tree as a \emptyset -terminated list of branches $E_1 \mid \dots \mid E_n \mid \emptyset$ (abbreviated with $E_1 \mid \dots \mid E_n$) which start from the root. Each branch E_i has the form $a[V]$ and denotes an edge labelled a leading to a node containing the data V . A data item can be a subtree T , a pointer $p @ l$ referencing the data selected at location

l by query p (described below), or a script $\langle A \rangle$ (described in Section 2) which can be executed to collect data or perform coordination tasks. For example, $a[b[c[\langle A \rangle]d[p@l]]e[]]$ is a tree containing a script and a pointer (we abbreviate $e[\emptyset]$ with $e[]$). We assume a standard function fv returning the free variables of a term, and we use the same identifiers x, y, z, \dots to range over all variables. When necessary, the kind of each variable can be understood by the place where the variable occurs.

From now on, use the word “queries” to mean expressions for querying *or* updating a tree. Core $Xd\pi$ is parametric on the choice of a particular query language chosen, as long as it is a language of expressions which can be evaluated against a tree to obtain a new tree (the result of updating the tree) and some data (the list of trees resulting from querying the tree). The only conditions that we need to impose on such a language are that the application of a substitution to a query must be well-defined and yield a query. The reasons why we need to define substitutions on queries will be clear after describing the semantics of processes in Section 2.

Definition 2.1 (Query Language). *A query language consists of a triple (Q, fv, \mathfrak{E}) where Q is a set of queries ranged over by p, q, \dots , together with a function $fv : Q \rightarrow \wp(\mathcal{V})$ giving the free variables of each query, and an evaluation function $\mathfrak{E} : (Q \times \mathcal{T}) \rightarrow \mathcal{T} \times \text{lists}(\mathcal{D})$, which, given a query and a tree, returns an updated tree and a finite list of results. Additionally, Q must be closed under substitution.*

Note that query evaluation is a partial function. This generality accounts for both the cases of ill-formed queries, which may not have a precise semantics, and Turing-equivalent query languages, which may not terminate. In [17, 18] we define a concrete query language as an example.

Networks. A Core $Xd\pi$ network represents a peer-to-peer system, where each location corresponds to a peer. Each peer can communicate with any other peer, and has a unique name. A network is represented by a pair (D, P) where the first component (the *store*) is a finite partial function from location names to trees, and the second component is a process. The formal definition is given in Figure 2. For example, in the network $(\{l \mapsto T\}, P)$, the term $\{l \mapsto T\}$ says that the store of the peer at location l is the tree T , and the term P represents the processes running on the peer, which contain explicit location information. Interaction between processes and data is always local. We assume a function dom giving the domain of both networks, stores and processes. By definition, the domain of a network is the domain of the store, and a network is well-formed if the domain of the process is contained in the domain of the store.

Processes and scripts. The formal definition for Core $Xd\pi$ processes is given in Figure 2. Parallel composition, restriction and the $\mathbf{0}$ process are standard. The

output process $\overline{l \cdot c}(\tilde{v})$ denotes a vector of values \tilde{v} waiting to be sent via channel c at location l , the *input* process $l \cdot c(\tilde{\pi}).P$ waits to receive values matching the patterns $\tilde{\pi}$ from an output process via channel c at l , and the *replicated input* is standard. The communication constructs, which use polyadic synchronization, are inspired by the $\epsilon\pi$ -calculus [7]. We assume a well-formedness predicate $wf(P)$ requiring that the continuation of an input (or replicated input) process must be located at the same location where the input is defined. Channel names are partitioned into *private* and *service* channel names. The private channels denote “usual” π -calculus channels, which are typically used for coordination, and which can be kept secret in order to protect a protocol from external interferences. The service channels denote those channels which are used to implement the services which a peer offers to other peers, and which therefore are not meant to be restricted and can be referenced inside scripts. In order to parse trees and pointers, we have added to π -calculus communication a very simple form of pattern matching. Patterns π_1, \dots, π_n are terms containing distinct variables (we assume a predicate *distinct* to enforce that) which are instantiated, if pattern matching succeeds, with the values found in the corresponding position in the term to be matched. Our patterns do not include regular or recursive expressions, and we will avoid algorithmic issues by simply requiring the guessing of an appropriate substitution in order for pattern matching to take place. Our processes use patterns to parse data, and queries to query trees. This conceptual separation does not exclude the possibility for the query language to be based on pattern matching itself.

A script $(x, \tilde{\pi})P$ represents the code of P parameterized on x , a placeholder for the location where the script is going to be run, and $\tilde{\pi}$, other optional parameters of P . By the side condition on the free variables of scripts, we impose that scripts remain statically defined until they are deployed dynamically by instantiation of their parameters. The application construct $A \circ \langle l, \tilde{v} \rangle$ passes the parameters \tilde{v} to the script A and runs it in the working space of l . Note that application is defined only when the first parameter passed to the script is a location. Communication in Core $Xd\pi$ is higher-order, in the sense that processes may send scripts over channels, possibly as leaves inside trees.

Process migration, which we represent explicitly, models communication across locations: the process $m\text{-go } l.P$ represents a (higher-order) message from m addressed to l containing a request to run the (closed) code P . When P is an output process, we use the abbreviation $l \cdot \overline{m \cdot c}(\tilde{v})$ for $l\text{-go } m.\overline{m \cdot c}(\tilde{v})$. The well-formedness condition $wf(P)$ requires that the continuation process must be correctly located at the destination location. Due to the peer-to-peer nature of our domain, each location is ready to receive and run any incoming code, so we do not need to provide an explicit operation to run a received process. Processes access the local tree by using a request operation $l \cdot \text{req}_p \langle c \rangle$ parametric in a query-update expression p and a channel c . The effect of evaluating expression p is to modify the local tree and

Figure 3: Semantics: reduction and contexts

(CR <small>ED</small> ST <small>AY</small>)	$(\{l \mapsto T\}, l.\text{go } l.P \mid Q) \longrightarrow (\{l \mapsto T\}, P \mid Q)$	
(CR <small>ED</small> G <small>o</small>)	$(\{l \mapsto T\} \uplus \{m \mapsto S\}, l.\text{go } m.P \mid Q) \longrightarrow (\{l \mapsto T\} \uplus \{m \mapsto S\}, P \mid Q)$	
(CR <small>ED</small> C <small>OM</small>)	$(\{l \mapsto T\}, \overline{l.c}(\widetilde{\pi}\sigma) \mid l.c(\widetilde{\pi}).P \mid Q) \longrightarrow (\{l \mapsto T\}, P\sigma \mid Q) \quad c \in C_p \cup C_s$	
(CR <small>ED</small> C <small>OM</small> !)	$(\{l \mapsto T\}, \overline{l.c}(\widetilde{\pi}\sigma) \mid !l.c(\widetilde{\pi}).P \mid Q) \longrightarrow (\{l \mapsto T\}, !l.c(\widetilde{\pi}).P \mid P\sigma \mid Q) \quad c \in C_p \cup C_s$	
(CR <small>ED</small> R <small>UN</small>)	$(\{l \mapsto T\}, (x, \widetilde{\pi})P \circ \langle l, \widetilde{\pi}\sigma \rangle \mid Q) \longrightarrow (\{l \mapsto T\}, P\{l/x\}\sigma \mid Q)$	
(CR <small>ED</small> R <small>EQ</small> U <small>EST</small>)	$\mathfrak{E}(p, T) = (T', U_1 \mid \dots \mid U_n \mid \emptyset)$	
	$(\{l \mapsto T\}, l.\text{req}_p\langle c \rangle \mid Q) \longrightarrow (\{l \mapsto T'\}, \overline{l.c}(\text{x}[U_1] \mid \dots \mid \text{x}[U_n] \mid \emptyset) \mid Q)$	
$\mathcal{K}_P \stackrel{\text{def}}{=} C_P[-] ::= - \mid P \mid C_P[-] \mid C_P[-] \mid P \mid (\nu c)C_P[-]$		(P <small>ROCESS</small> C <small>ONTEXTS</small>)
$\mathcal{K}_S \stackrel{\text{def}}{=} C_S[-] ::= - \mid C_S[-] \uplus D$		(S <small>TORE</small> C <small>ONTEXTS</small>)
$\mathcal{K}_N \stackrel{\text{def}}{=} C_N[-, -] ::= (C_S[-], C_P[-]), \text{dom}(C_P) \subseteq \text{dom}(C_S)$		(N <small>ETWORKS</small> C <small>ONTEXTS</small>)
$(C_S[-], C_P[-])(D, P) \stackrel{\text{def}}{=} (C_S[D], C_P[P])$		(C <small>ONTEXT</small> A <small>PPLICATION</small>)
Notation: $\{l \mapsto T\}(l) \stackrel{\text{def}}{=} T; \quad (D \uplus B)(l) \stackrel{\text{def}}{=} \begin{cases} D(l) & \text{if } l \in \text{dom}(D) \\ B(l) & \text{if } l \in \text{dom}(B) \end{cases}$		

to return a list of query results on the c . Our request operation is defined for any query which given a tree returns an updated tree and a list of results.

2.1 Reduction semantics

The reduction relation \longrightarrow for Core $Xd\pi$ describes process interaction, the interaction between processes and data, and the movement of processes across locations. The definition is given in Figure 3. We assume a standard notion of structural congruence for processes and networks and standard rules which allow reduction under parallel composition, restriction and structural congruence.

There are two rules for process movement between locations: rule (CRED STAY) describes the case where the process is already at the target location, and rule (CRED Go) allows a process $l.\text{go } m.P$ to move from l to m . Rule (CRED COM) states that if an output $\overline{l.a}(\widetilde{\nu})$ and an input $l.a(\widetilde{\pi}).P$ on the same channel a are in the same

location l (part of the store), and the values \tilde{v} match the input patterns $\tilde{\pi}$ (there is a substitution σ such that $\tilde{v} = \tilde{\pi}\sigma$), then communication takes place and execution proceeds with $P\sigma$. Rule (CRED COM!) is similar, but leaves the replicated input process $!l \cdot a(\tilde{\pi}).P$ in place for further use. We show an example of the communication of a private channel over a service channel below:

$$\begin{aligned} & (\{l \mapsto T\}, (\nu c)(\overline{l \cdot a}(c, b[y]) \mid l \cdot a(x, b[y]).(\overline{l \cdot x}(y) \mid P)) \\ & \longrightarrow (\{l \mapsto T\}, (\nu c)(\overline{l \cdot c}(\emptyset) \mid P\{c/x, \emptyset/y\})) \end{aligned}$$

Rule (CRED RUN) runs a script, passing as the first parameter the name of the location where it is going to run. Rule (CRED REQUEST) applies the query denoted by p on the local tree T , obtaining an updated tree T' which replaces T , and a list of results $U_1 \dots U_n \emptyset$ which is turned into a tree of results labelled with r (a reserved label used to denote results) sent on channel c at l . For example, supposing that p is a query which extracts from a tree the data found by following the path a/b :

$$\begin{aligned} & (\{l \mapsto a[b[V_1]]b[V_2]a[U]\}, l \cdot \text{req}_p(c) \mid P) \\ & \longrightarrow (\{l \mapsto a[b[]]b[]a[U]\}, \overline{l \cdot c}(r[V_1]r[V_2]) \mid P) \end{aligned}$$

Note that the subtrees V_i removed from the store are returned as results on c .

2.2 Behavioural Equivalences

Network equivalence for Core $Xd\pi$ is a standard reduction-closed, contextual equivalence [14] which preserves request observations (the effect processes have on data). In [17] we consider different choices of observation predicates (in particular the shape of the data tree of a location and the presence of output actions in a process), resulting in congruences which coincide or are implied by the one below.

Definition 2.2 (Request observation predicate). *We define the request observation predicate $\downarrow_{l \cdot p}$ as $(D, P) \downarrow_{l \cdot p} \iff \exists C, c, Q. P \equiv C[l \cdot \text{req}_p(c) \mid Q]$ and the weak observation predicate $\Downarrow_{l \cdot p}$ as $N \Downarrow_{l \cdot p} \iff \exists N'. N \xrightarrow{*} N' \text{ and } N' \downarrow_{l \cdot p}$.*

Definition 2.3 (Reduction Congruence). *Reduction congruence \simeq on Core $Xd\pi$ networks is the largest symmetric relation \simeq which is*

- *observation preserving:* $N \simeq M \implies \forall l, p. N \Downarrow_{l \cdot p} \implies M \Downarrow_{l \cdot p}$;
- *reduction closed:* $N \simeq M \implies \forall N'. N \xrightarrow{*} N' \implies \exists M'. M \xrightarrow{*} M', N' \simeq M'$;
- *contextual:* $N \simeq M \implies \forall C. C[N] \simeq C[M]$.

For example, we have $(\{l \mapsto T\}, \overline{l \cdot a \langle c \rangle}) \neq (\{l \mapsto T\}, \overline{l \cdot a \langle b \rangle})$, because the context $K = (-, - \mid l \cdot a(x). \overline{l \cdot x} \mid l \cdot c. l \cdot \text{req}_p \langle a \rangle)$ can tell a difference between the two networks.

Process equivalence. We now define process equivalence for Core $Xd\pi$. This equivalence depends on the locations present in the network. Consider replacing the definition of a service at location l , which uses only local data, with one located at m (where there is a cached copy of the same data) and providing an equivalent functionality. If location m is connected, then the behaviour of the services is the same. On the other hand, if location m is not connected, the behaviour of the services is different. With network equivalence, the connected locations are those in the domain of the store. With process equivalence, we must state explicitly the locations which we assume to be part of the network. As a consequence, process equivalence is indexed by a *domain* (a set of locations) Λ .

A Core $Xd\pi$ process can be seen as a partial specification of a network, describing only some of the processes running in some of the locations. This point of view is useful for reasoning about replacing components which are part of some distributed data-exchange protocol. Accordingly, we say that two processes P and Q are equivalent with respect to a domain Λ if all the networks containing *at least* the locations in Λ and either P or Q , are equivalent.

Besides comparing partial network specifications, process equivalences can be useful for example to replace optimized pieces of code inside a specific process.

Definition 2.4 (Extended Contexts). Extended process contexts \mathcal{K}_e are the terms generated by $C ::= - \mid C \mid P \mid P \mid C \mid (\nu c)C \mid l \cdot a(\tilde{\pi}).C \mid !l \cdot a(\tilde{\pi}).C \mid l \cdot \text{go } m.C$.

Definition 2.5 (Domain Congruence). Given a set of location names Λ , we define the induced domain congruence \sim^Λ on processes by

$$\sim^\Lambda = \{(P, Q) : \forall D, C[-]. \Lambda \subseteq \text{dom}(D) \implies (D, C[P]) \simeq (D, C[Q])\}.$$

Domain congruence is monotonic (if $\Lambda \subseteq \Lambda'$ then $\sim^\Lambda \subseteq \sim^{\Lambda'}$): the larger the set of locations which we assume to be part of the network, the larger the number of processes which we can equate.

Asynchronous Laws. Core $Xd\pi$ is an extension of the asynchronous π -calculus, so we consider some equational laws inspired by the latter. The *asynchrony law*, stating that the presence of a communication buffer cannot be observed, holds also in Core $Xd\pi$:

$$!l \cdot a(\tilde{\pi}). \overline{l \cdot a}(\tilde{\pi}) \sim_r^\Lambda 0.$$

The law stating that two channels a and b cannot be distinguished if they are part of the same equator does not hold. For example,

$$!l \cdot a(\tilde{\pi}). \overline{l \cdot b}(\tilde{\pi}) \mid !l \cdot b(\tilde{\pi}). \overline{l \cdot a}(\tilde{\pi}) \mid \overline{l \cdot c} \langle a \rangle \not\sim_r^\Lambda !l \cdot a(\tilde{\pi}). \overline{l \cdot b}(\tilde{\pi}) \mid !l \cdot b(\tilde{\pi}). \overline{l \cdot a}(\tilde{\pi}) \mid \overline{l \cdot c} \langle b \rangle$$

because a context could intercept the channel name a and use it in some fresh location m where a and b are not equated. We have instead a new law about equating located channels across different locations:

$$\begin{aligned} & !l \cdot a(\pi).l \cdot \overline{m \cdot b}(\pi) \mid !m \cdot b(\pi).m \cdot \overline{l \cdot a}(\pi) \mid \overline{l \cdot a}(\pi\sigma) \\ & \sim_r^{\{l,m\}} !l \cdot a(\pi).l \cdot \overline{m \cdot b}(\pi) \mid !m \cdot b(\pi).m \cdot \overline{l \cdot a}(\pi) \mid \overline{m \cdot b}(\pi\sigma) \end{aligned}$$

This law could be useful to show that we can replicate Web services (improving efficiency) without the clients needing to be aware of the change.

3 Distributed Query Patterns

We now study some communication patterns used by servers in distributed query systems to answer queries from clients. In *Core Xdπ*, distributed queries take the form of processes which retrieve and combine data from different locations by using remote communication and local requests. We show that some existing patterns [26] can be combined together obtaining a flexible infrastructure which corresponds to an intuitive specification of the intended behaviour. By exploiting process migration, we also propose a new communication pattern, and we show that it is behaviourally equivalent to a naive, less efficient one. In [17, 18] we argue about the equivalences claimed below.

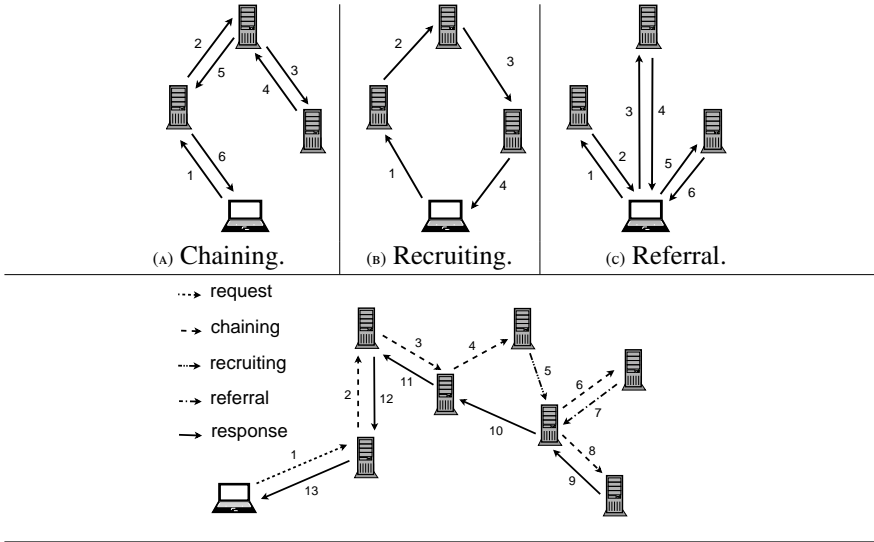
3.1 Chaining, recruiting and referral

We now consider *chaining*, *recruiting* and *referral*, three distributed query patterns studied by Sahuguet *et al.* in [26, 24] and described below. These patterns are interesting because, as will soon be apparent, they are simple yet can express ways of answering requests which are non-trivial, and display different levels of cooperation between the parties involved.

The usage of these patterns presupposes an architecture of servers sharing a common communication protocol for answering cooperatively the queries issued by clients. The protocol consists of alternative actions which depend on the contents of a query and on the local data, and is implemented by dedicated services running on each peer. The distributed querying infrastructure obtained by combining the three query patterns is very flexible and can provide location independence to the clients. In fact, a client simply needs to invoke a service on a peer acting as the “entry point” to the network in order to get access to data which may reside on some other server unknown to the client itself.

We now describe the three patterns. In each case, a server receiving a query will try to execute it locally, and if that is not possible, will take alternative action.

Figure 4: Chaining, recruiting and referral



Chaining (Figure 4.(A)): if a server cannot deal directly with the call, it re-issues it to an alternative server, waits for the answer, and then returns the answer to the client. Recruiting (Figure 4.(B)): if a server cannot deal directly with the call, it forwards it to another server (without noticing the client), so that the result will eventually return to the client without further intervention of the first server. To implement this pattern the address for returning the result must be a parameter of the call, and the client must be willing to accept asynchronous connections. Referral (Figure 4.(C)): if a server cannot deal directly with the call, it suggests to the client an alternative server which might be able to. This strategy requires active collaboration from the client, which must be ready to contact the alternative server. When each server involved in answering a request is able to use any of the patterns above, the flow of the data from the initial service call to the final answer can become complex and involve arbitrary combinations of the patterns, as in the example shown in Figure 4.

Implementing the patterns. We now describe, step by step, some Core Xd π code which implements a system where a client request can be answered by servers using an arbitrary combination of chaining, recruiting and referral. The code is based on services which retrieve and combine data from different locations by exploiting remote communication and local requests. A simple way to

represent a web service call in Core $\lambda d\pi$ is

$$(\nu c)(l.\overline{m}a(\widetilde{v}, l, c) \mid l.c(\widetilde{\pi}).P)$$

where a is the name of the service to be invoked at location m with parameters \widetilde{v} yielding a result to be passed on the channel c local to l , and P is the code for handling the results, which are expected to match $\widetilde{\pi}$. In this section, a service call will carry four parameters: a tree T used to represent a condition, checked using pattern matching, that a server must satisfy in order to provide the right service (for example specifying the kind of result expected), a query p which is meant to be run on the store of the service matching tag T , and the return parameters m and c stating the location and the channel where the result should be returned.

A client must be able to deal with the referral query pattern, therefore its code consists essentially of a loop. The loop consists of calling a first server (which could in principle provide the final result, terminating the loop), and then repeating the same call at the alternative addresses received in unsuccessful replies, until a reply containing the final result is received. The context defined below implements the loop at location m :

$$m.\text{Ref}_{(n,l,s,T,p,z)}[-] \stackrel{\text{def}}{=} (\nu c) \left(\frac{m.c(\text{OK}[], z). - \mid}{\overline{m}.c(\text{REF}[], l) \mid !m.c(\text{REF}[], x).m.\overline{x}.s(T, p, n, c)} \right)$$

It is parametric in the location n where the result must be returned, the location l of the first server to be interrogated, and the parameters of the call: the service name s and condition T , the actual query p and the variable z for binding the result in the continuation. The context uses a private channel c to implement the referral loop and uses the tags $\text{OK}[]$ and $\text{REF}[]$ as guards to exit or continue the loop. Any process built using this context always starts the referral loop by invoking s at l and then waiting for two possible answers: either a referral message with the tag $\text{REF}[]$ and the name of an alternative location (bound to x), which starts another iteration of the loop against the corresponding server, or a result message with the tag $\text{OK}[]$ and the result of the service call (bound to z), which terminates the loop and passes the result on to the process which replaces the context hole “-”.

The server filters calls based on the parameter T and decides whether they can be served locally or not. Its code consists of the parallel composition of the two

processes below, given along with the syntactic sugar or products and summations:

$$\begin{aligned}
 l\text{-Local}_{(s,T)} &\stackrel{\text{def}}{=} !l.s(T, x, y, z).(\nu c)(l.\text{req}_x\langle c \rangle \mid l.c(w).l.\overline{y}.z\langle \text{OK}[\] , w \rangle) \\
 l\text{-Remote}_{(s,\Delta)} &\stackrel{\text{def}}{=} \prod_{(m,S_m) \in \Delta} !l.s(S_m, x, y, z). \left(\begin{array}{l} l.\overline{m}.s\langle S_m, x, y, z \rangle \\ \oplus l.\overline{y}.z\langle \text{REF}[\] , m \rangle \\ \oplus l.\text{Ref}_{(l,m,s,S_m,x,w)}[l.\overline{y}.z\langle \text{OK}[\] , w \rangle]. \end{array} \right) \\
 \bigoplus_{i \in 1..n} lP_i &\stackrel{\text{def}}{=} (\nu c)(\overline{l}.c \mid \prod_{i \in 1..n} l.c.P_i) \quad c \notin \text{fn}(P_i) \\
 \prod_{i \in 1..n} P_i &\stackrel{\text{def}}{=} P_1 \mid \dots \mid P_n \quad P_1 \oplus \dots \oplus P_n \stackrel{\text{def}}{=} \bigoplus_{i \in 1..n} lP_i
 \end{aligned}$$

If the first parameter matches T , the server runs the query (bound to x) on the local data and sends the result back to the client on channel z at y . If the first parameter does not match T , the server selects another server more appropriate for that request out of the set Δ relating servers to tags specifying their services (the outermost parallel composition of the remote process). It then invokes s on the chosen server using either chaining (third branch of the choice), or recruiting (first branch), or referral (second branch). In the case of chaining, the server runs the same code as the client with different parameters. Notice that the code handling the result forwards the result to the client instead of using it locally.

Installation. In order to use these patterns, the code implementing the services must be installed somehow on each participating server. We can assume that it is pre-installed on each peer, or we can install it “on demand” using either process migration or a specialized service which runs scripts. For example, consider the code of a service P parametric in the location x where the service is run and some other initialization pattern π . If we assume that an arbitrary location l exists then, given an arbitrary initialization parameter $v = \pi\sigma$, it is easy to see that running the code at m is equivalent to installing the service code from l :

$$P\{m/x\}\sigma \sim^{[l]} l.\text{go } m.(x, \pi)P \circ \langle m, v \rangle.$$

Alternatively, one could use a dedicated installation service $Inst$ at location m which receives an abstraction and some parameters, and runs the abstraction locally:

$$P\{m/x\}\sigma \sim^{[l]} (\nu Inst)(\overline{l.m}.Inst\langle (x, \pi)P, v \rangle \mid m.Inst(y, z).y \circ \langle m, z \rangle).$$

Relating the patterns to a specification. We use a simple system with a client and two servers as an example of how to compare the patterns to a simple specification. The reasoning is analogous in the case of multiple servers. The client is on peer m , and runs the code

$$m\text{-Client}_{(l,s,a[\])} \stackrel{\text{def}}{=} m.\text{Ref}_{(m,l,s,a[\],p,z)}[m.P]$$

where the service is requested to match the tag $a[]$, and the continuation process P is an arbitrary process located at m which does not contain free occurrences of channel c mentioned in the definition of $\text{Ref}_{(-)}$. A server is composed by the parallel composition of the branches dealing with local and remote processing, as described in Section 3.1:

$$l_1 \cdot \text{Server}_{(s,T,l_2,S)} \stackrel{\text{def}}{=} l_1 \cdot \text{Remote}_{(s,\{(l_2,S)\})} \mid l_1 \cdot \text{Local}_{(s,T)}.$$

We consider two processes P_1 and P_2 , where a client requests from the server at l_1 the data specified by $a[]$ (served locally) or $b[]$ (served remotely at l_2).

$$\text{Servers} \stackrel{\text{def}}{=} l_1 \cdot \text{Server}_{(s,a[],l_2,b[])} \mid l_2 \cdot \text{Server}_{(s,b[],l_1,a[])}$$

$$P_1 \stackrel{\text{def}}{=} m \cdot \text{Client}_{(l_1,s,a[])} \mid \text{Servers} \quad P_2 \stackrel{\text{def}}{=} m \cdot \text{Client}_{(l_1,s,b[])} \mid \text{Servers}$$

We compare P_1 and P_2 with Q_1 and Q_2 defined below, which provide a specification of the expected behaviour respectively of P_1 and P_2 . Each process goes directly from m to the relevant location, fetches the data returned by query p , and goes back to paste it as the new data tree of m :

$$m \cdot \text{Spec}_{(l)} \stackrel{\text{def}}{=} m \cdot \text{go } l.(v c)(l \cdot \text{req}_p \langle c \rangle \mid l \cdot c(z).l \cdot \text{go } m.m \cdot P)$$

$$Q_1 \stackrel{\text{def}}{=} m \cdot \text{Spec}_{(l_1)} \mid \text{Servers} \quad Q_2 \stackrel{\text{def}}{=} m \cdot \text{Spec}_{(l_2)} \mid \text{Servers}.$$

An important difference between the client and the specification is that the client sends an output message to a service which can in principle be intercepted by some process external to the protocol which performs an input on the same service channel. To rule out this undesired interference, we restrict the name of the service s both in each P_i and Q_i , with the side effect of preventing also the unharmed case in which several clients use the services at the same time. We can show, in a domain containing both l_1 and l_2 , that both

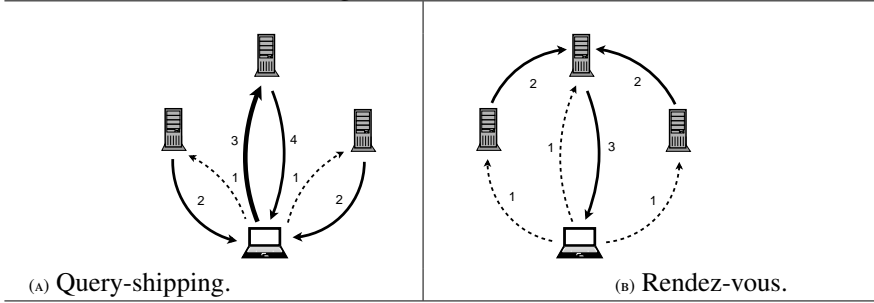
$$(v s)P_1 \sim^{[l_1,l_2]} (v s)Q_1 \text{ and } (v s)P_2 \sim^{[l_1,l_2]} (v s)Q_2.$$

Hence, by definition, we can replace $(v s)P_i$ by $(v s)Q_i$ in any network, and preserve network equivalence.

3.2 Rendez-vous and shipping

In the previous example, the infrastructure of servers implementing the distributed query patterns was fixed in advance, while the actual interactions between them were determined at run-time. The messages exchanged between different locations were always service calls or their results. Now, we consider a more flexible scenario which exploits code mobility.

Figure 5: Rendez-vous



Data-shipping and *query-shipping* are two traditional database techniques for distributed query evaluation: the first consists of evaluating locally a query on remote data by asking for the relevant data to be sent from the remote sources; the second consists of delegating the evaluation of a query to one of the remote sources in order to reduce the bandwidth used by data transfers. Below, we propose a distributed query pattern, called *rendez-vous*, which combines data and query shipping by using code mobility and private channels. The idea is to give a client the ability to ship result-handling code to another location, and to redirect the results of arbitrary service calls towards the location containing the result-handler. Within an infrastructure of services such as the one used above for chaining, recruiting and referral, this pattern can help to save bandwidth by eliminating unnecessary data transfers.

Comparing query-shipping and rendez-vous. We now compare the query-shipping and rendez-vous patterns by giving a concrete example where a client calls a remote service using as parameters two large sets of data obtained by other remote service calls. Suppose that on location l there is a specialized service $l\text{-Join}(x_1, x_2, y, z)$ which returns on channel z at location y the result of joining the data bound to x_1 with the data bound to x_2 . Suppose moreover that a client running on location m wants to join some data obtained by query p at location l_1 with other data obtained by query q at location l_2 . We assume that l_1 and l_2 run the services described in Section 3.1, that l_1 (respectively l_2) serves locally the requests tagged by $a[]$ (respectively $b[]$).

Query shipping. The client can use query shipping: it first invokes the query services at locations l_1 and l_2 , then passes on the results as inputs to the join service on location l (see Figure 5(A)). Below we give the code of a client implementing

this approach:

$$m\text{-ClientQ} \stackrel{\text{def}}{=} (\nu c, c_1, c_2) \left(\begin{array}{l} m\cdot\overline{l_1}\cdot\mathbf{S}\langle a[], p, m, c_1 \rangle \\ | m\cdot\overline{l_2}\cdot\mathbf{S}\langle b[], q, m, c_2 \rangle \\ | m\cdot c_1(\text{OK}[], x_1).m\cdot c_2(\text{OK}[], x_2).m\cdot\overline{l}\cdot\mathbf{Join}\langle x_1, x_2, m, c \rangle \\ | m\cdot c(z).m\cdot\mathbf{P} \end{array} \right)$$

It starts sending off the two service calls to l_1 and l_2 and then waits for the results respectively on c_1 and c_2 to bind them to x_1 and x_2 . The remaining code is a standard service call for the join service at l with parameters x_1 and x_2 , binding the final result to z in the continuation $m\cdot\mathbf{P}$, which can be an arbitrary process.

Rendez-vous. In order to save bandwidth, a better strategy is to request the query services at l_1 and l_2 to forward their results to location l , and to install at l a process which collects the two results and invokes the join service locally, asking for the final result to be returned at location m (see Figure 5_(B)). Below we give a context implementing the general pattern, with two holes for inserting the code to handle the intermediate results at l and the final result at m . The code is parametric in the tags T_i and the queries p_i used to determine the partial results, the variables x_i for binding them in the intermediate code at “ $-_1$ ”, and the variable z for binding the final result in the continuation code at “ $-_2$ ”:

$$m\text{-RzV}_{(T_1, p_1, x_1, T_2, p_2, x_2, z)}[-]_1[-]_2 \stackrel{\text{def}}{=} (\nu c, c_1, c_2) \left(\begin{array}{l} m\cdot\overline{l_1}\cdot\mathbf{S}\langle T_1, p_1, l, c_1 \rangle \\ | m\cdot\overline{l_2}\cdot\mathbf{S}\langle T_2, p_2, l, c_2 \rangle \\ | m\cdot\mathbf{go} \ l.l\cdot c_1(\text{OK}[], x_1).l\cdot c_2(\text{OK}[], x_2).-_1 \\ | m\cdot c(z).-_2 \end{array} \right)$$

The code given above can be easily parameterized also on the number, the names and the locations of the services involved, and can be adapted to return the final results at an arbitrary location on an arbitrary channel.

We give below the code for a client, equivalent to ClientQ , which uses the rendez-vous strategy:

$$m\text{-ClientR} \stackrel{\text{def}}{=} m\text{-RzV}_{(a[], p, x_1, b[], q, x_2, z)}[\overline{l}\cdot\mathbf{Join}\langle x_1, x_2, m, c \rangle][m\cdot\mathbf{P}]$$

The code for handling the intermediate results consists in a local call to the join service, whereas the continuation is the same generic process used for ClientQ .

Equivalence of the patterns. Consider the process **Servers** defined in Section 3.1, consisting of the parallel compositions of the servers for implementing chaining, recruiting and referral at locations l_1 and l_2 . The clients given above,

each in parallel with Servers, are equivalent in any network regardless of what locations are present:

$$(\nu s)(m \cdot \text{ClientQ} \mid \text{Servers}) \sim^0 (\nu s)(m \cdot \text{ClientR} \mid \text{Servers}).$$

4 Related work

Core $Xd\pi$ was developed independently from the AXML system of Abiteboul *et al.* [30]. The key difference is that Core $Xd\pi$ consists of processes in the working space, and embedded processes (scripts) in the data, whereas AXML focusses on web services in the working space and service calls in the data. In contrast, the ubQL distributed query language of Sahuguet and Tannen [24] was a direct source of inspiration for the design of Core $Xd\pi$. The ubQL language incorporates process manipulation primitives to any “host” query language. These primitives, inspired by the π -calculus [25], are used in a *deployment phase* to set up a network of processes which, in a successive *execution phase*, will query local repositories and forward their results to other sites, thus implementing a global query execution plan. ubQL processes are able to deal with streaming data, but there is no support for concurrent execution of query-processes on the same site (so in principle the system may not be able to execute more than one global query at a time). The main influences of ubQL on the design of Core $Xd\pi$ were on the choices of separating the queries from the process primitives, and maintaining independence from a specific query language. Also our examples on distributed query patterns of Section 3 are inspired by ubQL. Overall, Core $Xd\pi$ and ubQL have a significantly different focus and are studied using different methodologies. For example, an important part of the work on ubQL is the study of algorithms for query installation based on cost estimates, which we do not address, whereas behavioural equivalences are not studied in ubQL. Both AXML and ubQL are studied from a data-management viewpoint, which our process algebraic techniques complement nicely. There are many specific issues which are important in databases, such as the use of meta-data to guide the optimization of queries, which we do not study. Instead we give a formal semantics to the distributed interaction between queries and processes, arguing about their equivalence and providing a framework on which to base a formal study of security properties.

We now consider work related to our process-algebraic approach. The work on Core $Xd\pi$, and its implicitly located predecessor $Xd\pi$ [11, 17] constitute one of the first attempts to integrate the study of mobile processes and semi-structured data, and are characterized by their emphasis on dynamic web data. To the best of our knowledge, the only work relating the π -calculus with XML which pre-dates ours is the Iota concurrent XML scripting language of Bierman and Sewell [4], used to

program Home Area Networks. Iota is a strongly typed functional language with concurrency primitives inspired by the π -calculus. Although the language has a formal semantics, its behavioural theory has not been studied. The programs for Home Area devices written in Iota are designed to run on the same Home Area server, and the communication with physical devices is modelled through input and output on special channels: distribution is not represented explicitly. Moreover, as opposed to $\text{Core } Xd\pi$, the application domain of Home Area Network programming is more control-oriented than data-oriented. In particular, there is no explicit representation of stores, which are central to our approach. Brown, Laneve and Meredith [6] have recently defined an (untyped) extension of the π -calculus with native XML datatypes called πDuce . They compare its expressivity to that of the functional language XDuce [15], and also consider a higher-order extension which enables dynamic content in documents. An interesting idea underlying the design of πDuce is that processes and data share a similar tree-like structure, and can inhabit the same semantic universe. The authors show a very simple encoding of an evaluator for the subset of the language without new name generation into the language itself: the execution of processes represented as nested document elements can be simulated in the language. A similar approach could be taken in $\text{Core } Xd\pi$ to represent scripts as semi-structured data. In the case of dynamic Web data though, it is better to hide the internal structure of processes from the queries, so that one can replace a process by an equivalent one whilst preserving the observable behaviour of the system as a whole. Castagna, De Nicola and Varacca [8] propose $\mathbb{C}\pi$, a π -calculus extended with pattern matching and tuples of values (XML values can be represented through an encoding). The language comes with a very expressive type system featuring intersection and input-output types. The language itself is not distributed and does not include a concept of store. Acciai and Boreale [2] have recently proposed XPi , an extension of the asynchronous π -calculus with code mobility and ML-like pattern matching of structured values. A combination of static and dynamic typing ensures that each channel always exchanges values of the same types, which describe the partial structure of documents. Pattern matching plays a lesser role in $\text{Core } Xd\pi$, although it could be easily extended to the more expressive form adopted in XPi . Query expressions instead, which are separate entities from processes, are the primary means to extract information from XML trees.

5 Conclusions

The process equivalence defined in Section 2.2 is hard to use directly because it requires a costly property of closure under all contexts. In [17, 18], we define *domain bisimilarity*, a coinductive equivalence relation defined using a labelled

transition system which does not quantify over contexts and which entails process equivalence. The definition of domain bisimilarity is non-standard. It requires an adaptation of a technique used for higher-order π -processes [27, 16] to our processes, since in our setting data containing processes may be passed as values. It is also sensitive to the set of locations in axs network, requiring a generalisation of bisimulation to families of relations indexed by sets of locations

An important design choice, enabling us to study how properties of data can be affected by process interaction, was to model data and processes at the same level of abstraction, rather than encoding data into processes, as customary in the π -calculus [19, 20, 21, 29]. Whilst such an encoding makes sense when using the π -calculus as a low-level concurrency language, it becomes a burden when reasoning about the coordination of higher-level processes. Our choice also kept our language modular with respect to the choice of a query language, which can be easily adapted from the existing literature on XML [5].

Migration has been included in Core $Xd\pi$ to maintain a closer correspondence with $Xd\pi$. It is not strictly necessary though, since each located action already contains information about where it is to be executed (confirming the thesis of [7] that locations can be encoded in the π -calculus with polyadic synchronization). For example, in the process below we can imagine that each input on a_i at l_i is followed by an implicit migration step $l_i \cdot \text{go } l_{i+1}$ to the next location: $l_1 \cdot a_1(x).l_2 \cdot a_2(y).l_3 \cdot a_3(x, y)$ corresponds to $l_1 \cdot a_1(x).l_1 \cdot \text{go } l_2.l_2 \cdot a_2(y).l_2 \cdot \text{go } l_3.l_3 \cdot a_3(x, y)$. Overall, if explicit migration were to be discarded, the presentation of our model would be slightly simpler.

Studying types for dynamic web data is on-going work. A type system would be useful to guarantee the absence of run-time errors, refine the behavioural equivalences, guarantee the conformance of data trees to schemas, and study security properties. Dezani *et. al.* [9] propose a first type system for $Xd\pi$, which guarantees that the security level of values exchanged during communication is compatible (lower or equal) with the level of the channel, and constraints the ability of processes to migrate and execute scripts, depending on their security level.

Given the use of mobile code in our systems, in the absence of trust, we face the problem of protecting a host from a potentially malicious agent. This problem could be tackled by dynamically type-checking each agent entering a location [12] (possibly relying on the ability of a location to infer the type of the agent), or by using the Proof Carrying Code [23] approach (to send a migrating process along with its type), or by a combination of both techniques.

References

- [1] Martín Abadi and Andrew D. Gordon. A Calculus for Cryptographic Protocols: The spi Calculus. *Information and Computation*, 148(1):1–70, 1999.
- [2] Lucia Acciai and Michele Boreale. XPi: A typed process calculus for XML messaging. In *Proc. of FMOODS'05*, 2005.
- [3] Omar Benjelloun. Active XML: A data-centric perspective on Web services. Ph.D. Thesis, University of Paris XI, 2002.
- [4] Gavin Bierman and Peter Sewell. Iota: a concurrent XML scripting language with application to Home Area Networks. University of Cambridge Technical Report 557, 2003.
- [5] Angela Bonifati and Stefano Ceri. Comparative analysis of five XML query languages. *SIGMOD Record*, 29(1):68–91, 2000.
- [6] Allen Brown, Cosimo Laneve, and Greg Meredith. PiDuce: a process calculus with native XML datatypes. In *Proc. of WSFM'05*. LNCS, 2005.
- [7] Marco Carbone and Sergio Maffeis. On the expressive power of polyadic synchronisation in π -calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.
- [8] Giuseppe Castagna, Rocco De Nicola, and Daniele Varacca. Semantic subtyping for the pi-calculus. In *Proc. of LICS'05*, pages 92–101. IEEE Computer Society Press, 2005.
- [9] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan and Jovanka Pantovic. Security types for dynamic Web data. In *Proc. of TGC'06*, 2006.
- [10] Cédric Fournet, Andrew D. Gordon, and Sergio Maffeis. A Type Discipline for Authorization Policies. *ACM Trans. Program. Lang. Syst.*, 29.5, 2007.
- [11] Philippa Gardner and Sergio Maffeis. Modelling dynamic Web data. *Theoretical Computer Science*, 342:104–131, 2005.
- [12] Matthew Hennessy, Julian Rathke, and Nobuko Yoshida. safeDpi: A language for controlling mobile code. In *Proc. of FoSSaCS 2004*, volume 2987 of LNCS, pages 241–256. Springer Verlag, 2004.
- [13] Matthew Hennessy and James Riely. Resource access control in systems of mobile agents. In *HLCL '98*, volume 16.3 of *ENTCS*, pages 3–17. Elsevier Science Publishers, 1998.
- [14] Kohei Honda and Nobuko Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151(2):437–486, 1995.
- [15] Haruo Hosoya and Benjamin C. Pierce. Xduce: A statically typed xml processing language. *ACM Transactions on Internet Technology*, 3(2):117–148, 2003.
- [16] Alan Jeffrey and Julian Rathke. Contextual equivalence for higher-order pi-calculus revisited. *Logical Methods in Computer Science*, 1(1:4), 2005.

- [17] Sergio Maffei. *Dynamic Web Data: A Process Algebraic Approach*. PhD thesis, Imperial College London, September 2005.
- [18] Sergio Maffei and Philippa Gardner. Behavioural equivalences for dynamic Web data. *Journal of Logic and Algebraic Programming*, to appear, 2007.
- [19] Robin Milner. *Communication and concurrency*. Prentice-Hall, Inc., 1989.
- [20] Robin Milner. *Communicating and Mobile Systems: the π -Calculus*. Cambridge University Press, May 1999.
- [21] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40,41–77, September 1992.
- [22] Robin Milner and Davide Sangiorgi. Techniques of weak bisimulation up-to. In *Proc. of CONCUR '92*, volume 2987 of LNCS, pages 41–77. Springer-Verlag, 1992.
- [23] George Necula and Peter Lee. Safe, untrusted agents using proof-carrying code. In *Mobile Agents and Security*, pages 61–91. Springer Verlag, 1998.
- [24] Arnaud Sahuguet. *ubQL: A Distributed Query Language to Program Distributed Query Systems*. PhD thesis, University of Pennsylvania, 2002.
- [25] Arnaud Sahuguet, Benjamin Pierce, and Val Tannen. Distributed Query Optimization: Can Mobile Agents Help? Unpublished draft, 2000.
- [26] Arnaud Sahuguet and Val Tannen. ubql, a language for programming distributed query systems. In *Proc. of webDB'01*, pages 37–42, 2001.
- [27] Davide Sangiorgi. Expressing mobility in process algebras: First-order and higher-order paradigms. PhD thesis, University of Edinburgh, 1992.
- [28] Davide Sangiorgi. A theory of bisimulation for the pi-calculus. In *Proc. of CONCUR '93*, pages 127–142, Springer Verlag, 1993.
- [29] Davide Sangiorgi and David Walker. *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
- [30] Serge Abiteboul, et al. Active XML primer. INRIA Futurs, GEMO Report number 275, 2003.
- [31] UDDI. Universal Description, Discovery, and Integration of Business for the Web (UDDI) 3.0. <http://www.uddi.org>, 2005.
- [32] W3C. Extensible Markup Language (XML) 1.0 (2nd edition). <http://www.w3.org/TR/REC-xml.html>, 2000.
- [33] W3C. Web Services Description Language (WSDL) 1.1. <http://w3.org/TR/wsdl>, 2001.
- [34] W3C. Web Services Activity. <http://www.w3.org/2002/ws>, 2002.
- [35] W3C. Simple Object Access Protocol (SOAP) Version 1.2. <http://w3.org/TR/SOAP>, 2003.

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

AN INTRODUCTION TO POPULATION PROTOCOLS

James Aspnes*
Yale University

Eric Ruppert†
York University

Abstract

The population protocol model describes a collection of tiny mobile agents that interact with one another to carry out a computation. The agents are identically programmed finite state machines. Interactions between pairs of agents cause the two agents to update their states. These interactions are scheduled by an adversary, subject to a fairness constraint. Input values are initially distributed to the agents, and the agents must eventually converge to the correct output value. This framework can be used to model mobile *ad hoc* networks of tiny devices or collections of molecules undergoing chemical reactions. We survey results that describe what can be computed in various versions of the population protocol model.

1 Introduction

The population protocol model [3] was designed to represent sensor networks consisting of very limited mobile agents with no control over their own movement.

*Supported in part by NSF grant CNS-0435201.

†Supported in part by the Natural Sciences and Engineering Research Council of Canada.

It also bears a strong resemblance to models of interacting molecules in theoretical chemistry [16, 17]. The defining features of the basic model are:

1. Anonymous, finite-state agents. The system consists of a large population of indistinguishable finite-state agents.
2. Computation by direct interaction. In the original model, agents do not send messages or share memory; instead, an **interaction** between two agents updates both of their states according to a joint transition table. The actual mechanism of such interactions is abstracted away.
3. Unpredictable interaction patterns. The choice of which agents interact is made by an adversary. Agents have little control over which other agents they interact with, although the adversary may be limited to pairing only agents that are adjacent in an **interaction graph**, typically representing distance constraints. A strong global **fairness condition** is imposed on the adversary to ensure the protocol makes progress.
4. Distributed inputs and outputs. The input to a population protocol is distributed across the initial state of the entire population. Similarly, the output is distributed to all agents.
5. Convergence rather than termination. Population protocols generally cannot detect when they have finished; instead, the agents' outputs are required to converge after some finite time to a common, correct value.

A formal definition is given in Section 2.

The population protocol model was inspired in part by work by Diamadi and Fischer [13] on trust propagation in a social network. The **urn automata** of [2] can be seen as a first draft of the model that retained in vestigial form several features of classical automata: instead of interacting with each other, agents could interact only with a finite-state controller, complete with input tape. The motivation given for the current model in [3] was the study of sensor networks in which passive agents were carried along by other entities; the canonical example was sensors attached to a flock of birds. The name of the model was chosen by analogy to **population processes** [22] in probability theory.

A population protocol often looks like an amorphous soup of lost, nearly mindless, anonymous agents blown here and there at the whim of the adversary. But though individual agents lack much intelligence or control over their own destinies, the population as a whole is nonetheless capable of performing significant computations—under some conditions, it has the same power as a traditional computer with the same total storage capacity. Some examples of simple population protocols are given in Section 2.

Much of the work so far on population protocols has concentrated on characterizing what predicates on the input configuration can be computed in different variants of the model and under various assumptions, such as a bounded-degree interaction graph or random scheduling.

If the interaction graph is unrestricted, the worst case for computation turns out to be a complete interaction graph, since any other interaction graph can simulate a complete interaction graph by shuffling agents between the nodes [3]. In a complete interaction graph, all agents with the same state are indistinguishable, and only the counts of agents in each state affect the outcome of the protocol. The set of computable predicates in most variants of the basic model for such a graph is now known to be either exactly equal to or closely related to the set of **semi-linear** predicates, those definable in **first-order Presburger arithmetic** [18, 25]. These results, which originally appeared in [1, 3, 5, 7, 8, 9, 12], are summarized in Sections 3, 4, 5, 7 and 9. Sometimes the structure of incomplete interaction graphs can be exploited to simulate a Turing machine, which implies that a restricted interaction graph can make the system stronger than the complete graph. didn't

More recent work has concentrated on performance. Because the weak scheduling assumptions in the basic model allow the adversary to draw out a computation indefinitely, the worst-case adversary scheduler is replaced by a random scheduling assumption, where the pair of agents that interacts at each step is drawn uniformly from the population as a whole. This gives a natural notion of **time** equal to the total number of steps to convergence and **parallel time** equal to the average number of steps initiated by any one agent (essentially the total number of steps divided by the number of agents).

As with adversarial scheduling, for random scheduling the best-understood case is that of a complete interaction graph. In this case, it is possible to construct a register machine, where subpopulations of the agents hold tokens representing the various register values in unary. It is not hard to implement register operations like addition, subtraction, and comparison by local operations between pairs of agents; with the election of a leader, we can further construct a finite-state control. The main obstacle to implementing a complete register machine is to ensure that every agent completes any needed tasks for each instruction cycle before the next cycle starts. In [3], this was handled by having the leader wait a polynomial number of steps on average before starting the next cycle, a process which gives an easy proof of polynomially-bounded error but which also gives an impractically large slowdown. Subsequent work has reduced the slowdown to polylogarithmic by using epidemics both to propagate information quickly through the population and to provide timing [4, 6]. These results are described in more detail in Section 6.

In addition to work on the basic population protocol model, there have been several extensions of the model to more accurately reflect the requirements of

practical systems. The basic model requires coordinated two-way communication between interacting agents; this assumption is relaxed in Section 4. Work on incorporating agent failures into the model are discussed in Sections 7 and 9. Versions of the model that give agents slightly increased memory capacity are discussed in Section 8.

2 The basic model

In the basic population protocol model, a collection of agents are each given an input value, and agents have pairwise interactions in an order determined by a scheduler, subject to some fairness guarantee. Each agent is a kind of finite state machine and the “program” for the system describes how the states of two agents can be updated by an interaction. The agents’ output values change over time and must eventually converge to the correct output value for the inputs that were initially distributed to the agents.

A protocol is formally specified by

- Q , a finite set of possible states for an agent,
- Σ , a finite input alphabet,
- ι , an input map from Σ to Q , where $\iota(\sigma)$ represents the initial state of an agent whose input is σ ,
- ω , an output map from Q to the output range Y , where $\omega(q)$ represents the output value of an agent in state q , and
- $\delta \subseteq Q^4$, a transition relation that describes how pairs of agents can interact.

We now describe how a computation proceeds according to such a protocol. The computation takes place among n **agents**, where $n \geq 2$. Each agent is given an input value from Σ . Each agent’s initial state is determined by applying ι to its input value. This determines an initial configuration for an execution. A **configuration** of the system can be described by a vector of all the agents’ states. Because agents with the same state are indistinguishable, each configuration can be summarized as an unordered multiset of states.

An execution of a protocol proceeds from the initial configuration by interactions between pairs of agents. Suppose two agents in states q_1 and q_2 meet and have an interaction. They can change into states q'_1 and q'_2 as a result of the interaction if (q_1, q_2, q'_1, q'_2) is in the transition relation δ . We sometimes describe δ by listing all possible interactions using the notation $(q_1, q_2) \rightarrow (q'_1, q'_2)$. (We assume a null transition $(q_1, q_2) \rightarrow (q_1, q_2)$ if no others are specified with (q_1, q_2) on the left hand side.) If there is only one possible transition $(q_1, q_2) \rightarrow (q'_1, q'_2)$ for each pair (q_1, q_2) , then the protocol is **deterministic**. If C and C' are configurations, we

write $C \rightarrow C'$ if C' can be obtained from C by a single interaction of two agents. This means that C contains two states q_1 and q_2 and C' is obtained from C by replacing q_1 and q_2 by q'_1 and q'_2 , where (q_1, q_2, q'_1, q'_2) is in δ . An **execution** of the protocol is an infinite sequence of configurations C_0, C_1, C_2, \dots , where C_0 is an initial configuration and $C_i \rightarrow C_{i+1}$ for all $i \geq 0$. Thus, an execution is a sequence of snapshots of the system after each interaction occurs. In a real distributed execution, interactions could take place simultaneously, but when writing down an execution we can order those simultaneous interactions arbitrarily.

The order in which pairs of agents interact is unpredictable: we think of the schedule of interactions as being chosen by an adversary, so that protocols must work correctly under any schedule the adversary may choose. In order for meaningful computations to take place, we must put some restrictions on the adversarial scheduler; otherwise it could divide the agents into isolated groups and schedule interactions only between agents that belong to the same group.

The **fairness** condition that we impose on the scheduler is quite simple to state, but is somewhat subtle. Essentially, we do not allow the scheduler to avoid a possible step forever. More formally, if C is a configuration that appears infinitely often in an execution, and $C \rightarrow C'$, then C' must also appear infinitely often in the execution. Another way to think of this is that anything that is always possible eventually happens: it is equivalent to require that any configuration that is always reachable is eventually reached.

At any point during an execution of a population protocol, each agent's state determines its output at that time. If the agent is in state q , its output value is $\omega(q)$. Thus, an agent's output may change over the course of an execution. The fairness constraint allows the scheduler to behave arbitrarily for an arbitrarily long period of time, but does require that it behave nicely eventually. It is therefore natural to phrase correctness as a property to be satisfied eventually too. For example, the scheduler could schedule only interactions between agents 1 and 2, leaving the other $n - 2$ agents isolated, for millions of years, and it would be unreasonable to expect any sensible output during the period when only two agents have undergone state changes. Thus, for correctness, we require that all agents produce the correct output (for the input values that were initially distributed to the agents) at some time in the execution and continue to do so forever after that time.

To summarize, we say that a protocol computes a function f that maps multisets of elements of Σ to Y if, for every such multiset I and every fair execution that starts from the initial configuration corresponding to I , the output value of every agent eventually stabilizes to $f(I)$.

Example 1. Suppose each agent is given an input bit, and we want all agents to output the 'or' of those bits. There is a very simple protocol to accomplish this: each agent with input 0 simply outputs 1 as soon as it discovers that another agent

had input 1. Formally, we have $\Sigma = Y = Q = \{0, 1\}$ and the input and output maps are the identity functions. The only interaction in δ is $(0, 1) \rightarrow (1, 1)$. If all agents have input 0, no agent will ever be in state 1. If some agent has input 1 the number of agents with state 1 cannot decrease and fairness ensures that it will eventually increase to n . In both cases, all agents stabilize to the correct output value.

Example 2. Suppose the agents represent dancers. Each dancer is (exclusively) a leader or a follower. We wish to determine whether there are more leaders than followers. We use $Y = \{0, 1\}$, with 1 indicating that there are more leaders than followers. A centralized solution would count the leaders and the followers and compare the totals. A more distributed solution is to ask everyone to start dancing with a partner (who must dance the opposite role) and then see if any dancers are left without a partner. We formalize this cancellation procedure as a population protocol with $\Sigma = \{L, F\}$ and $Q = \{L, F, 0, 1\}$. The input map ι is the identity, and the output map ω maps L and 1 to 1 and maps F and 0 to 0. The transitions of δ are $(L, F) \rightarrow (0, 0)$, $(L, 0) \rightarrow (L, 1)$, $(F, 1) \rightarrow (F, 0)$ and $(0, 1) \rightarrow (0, 0)$. The first rule ensures that, eventually, either no L 's or no F 's will remain. At that point, if there are L 's remaining, the second rule ensures that all agents will eventually produce output 1. Similarly, the third rule takes care of the case where F 's remain. In the case of a tie, the third rule ensures that the output stabilizes to 0.

It may not be obvious why the protocol of in Example ?? must converge. Consider, for example, the transitions

$$\{L, L, \underline{F}\} \rightarrow \{0, \underline{L}, 0\} \rightarrow \{1, L, \underline{0}\} \rightarrow \{0, \underline{L}, 0\} \rightarrow \{0, L, \underline{1}\} \rightarrow \{0, L, 0\},$$

where in each configuration, the agents that are about to interact are underlined. By repeating this loop, we obtain a non-converging execution in which every pair of agents interacts infinitely often. However, this execution is not fair: the configuration $\{0, L, 1\}$ appears infinitely often and $\{0, L, 1\} \rightarrow \{1, L, 1\}$, but $\{1, L, 1\}$ never appears. This is because the first two agents only interact at “inconvenient” times, *i.e.*, when third agent is in state 0. The definition of fairness rules this out. Thus, in some ways, the definition of fairness is stronger than saying that each pair of agents must interact infinitely often. (In fact, the two conditions are incomparable, since there can be fair executions in which two agents never meet. For example, an execution where every configuration is $\{L, L, L\}$ and all interactions take place between the first two agents is fair.)

Exercise 3. Show the protocol of Example 2 converges in every fair execution.

The definition of fairness was chosen to be quite weak (although it is still strong enough to allow useful computations). Many models of mobile systems assume that the mobility patterns of the agents follow some particular probability distribution. The goal of the population protocol model is to be more general.

If there is an (unknown) underlying probability distribution on the interactions, which might even vary with time, and that distribution satisfies certain independence properties and ensures that every interaction's probability is bounded away from 0, then an execution will be fair with probability 1. Thus, any protocol will converge to the correct output with probability 1. So the model captures computations that are correct with probability 1 for a wide range of probability distributions, even though the model definition does not explicitly incorporate probabilities.

Other predicates can be computed using an approach similar to Example 2.

Exercise 4. Design a population protocol to determine whether more than 40% of the dancers are leaders.

Some predicates, however, require a different approach.

Example 5. Suppose each agent is given an input from $\Sigma = \{0, 1, 2, 3\}$ and we wish to find the sum of the inputs, modulo 4. The protocol can gather the sum (modulo 4) into a single agent. Once an agent has given its value to another agent, its value becomes null, and it obtains its output values from the eventually unique agent with a non-null value. Formally, we have $Q = \{0, 1, 2, 3, \perp_0, \perp_1, \perp_2, \perp_3\}$, where \perp_v represents a null value with output v . Let $\iota(v) = v$ and $\omega(v) = \omega(\perp_v) = v$ for $v = 0, 1, 2, 3$. The transition rules of δ are $(v_1, v_2) \rightarrow (v_1 + v_2, \perp_{v_1+v_2})$ and $(v_1, \perp_{v_2}) \rightarrow (v_1, \perp_{v_1})$, where v_1 and v_2 are 0, 1, 2 or 3. (The addition is modulo 4.)

In some cases, agents may know when they have converged to the correct output, but in general they cannot. While computing the 'or' of input bits (Example 1), any agent in state 1 knows that its state will never change again: it has converged to its final output value. However, no agent in the protocol of Example 5 can ever be certain it has converged, since some additional agents with input 1 could always join the computation and change the required output value.

Two noteworthy properties of the population protocol model are its uniformity and anonymity. A protocol is **uniform** because its specification has no dependence on the number of agents that take part. In other words, no knowledge about the number of agents is required by the protocol. The system is **anonymous** because the agents are not equipped with unique identifiers and all agents are treated in the same way by the transition relation. Indeed, because the state set is finite and does not depend on the number of agents in the system, there is not even room in the state of an agent to store a unique identifier.

3 Computability

In studying what functions can be computed in the population protocol model, there is no loss of generality in restricting attention to predicates, *i.e.*, functions

with range $Y = \{0, 1\}$. For any function f with range Y , let $P_{f,y}$ be a predicate defined by $P_{f,y}(x) = 1$ if and only if $f(x) = y$. Then, f is computable if and only if $P_{f,y}$ is computable for each $y \in Y$. The “only if” part of this statement is trivial. For the converse, a protocol can compute all the predicates $P_{f,y}$ in parallel, using a separate component of each agent’s state for each y . Note that Y is finite because each distinct output value corresponds to at least one state in the original protocol.

For the basic population protocol model, there is an exact characterization of the computable predicates: they are precisely the **semilinear predicates**, which we now define. A multiset over the input alphabet Σ can also be thought of as a vector with $d = |\Sigma|$ components, where each component is a natural number representing the multiplicity of one input character. For example, the input multiset $\{a, a, a, b, b\}$ over the input alphabet $\Sigma = \{a, b, c\}$ can be represented by the vector $(3, 2, 0) \in \mathbb{N}^3$. A **semilinear set** is a subset of \mathbb{N}^d that is a finite union of **linear sets** of the form $\{\vec{b} + k_1\vec{d}_1 + k_2\vec{d}_2 + \cdots + k_m\vec{d}_m\}$, where \vec{b} is a d -dimensional base vector, \vec{d}_1 through \vec{d}_m are basis vectors, and k_1 through k_m are non-negative coefficients. A **semilinear predicate** on inputs is one that is true precisely on a semilinear set.

An alternative characterization of semilinear predicates is that they can be described by first-order logical formulas in Presburger arithmetic, which is arithmetic on the natural numbers with addition but not multiplication [25]. Presburger arithmetic allows for **quantifier elimination**, replacing universal and existential quantifiers with formulas involving addition, $<$, the mod- k congruence relation \equiv_k for each constant k , and the usual logical connectives \wedge , \vee , and \neg .

Angluin *et al.* [3] gave protocols to compute any threshold predicate or remainder predicate; the protocols are similar to those in Examples 2 and 5. This gives $<$ and \equiv_k . They further observed that addition is trivially obtained by renaming states: to compute $A + B$ from A and B we pretend that any A or B token is really an $A + B$ token. Finally, Boolean combinations (\wedge , \vee) of these predicates can be computed by running the protocols for each of the basic predicates in parallel, using separate components of the agents’ states, and negation (\neg) simply involves relabeling the output values. It follows that population protocols can compute all predicates definable with Presburger formulas, *i.e.*, all semilinear predicates.

Amazingly, the converse also holds: only semilinear predicates can be computed by population protocols. For the basic model this result was shown by Angluin, Aspnes, and Eisenstat [5] by applying results from partial order theory. The proof is quite involved, but the essential idea is that, like finite-state automata, population protocols can be “pumped” by adding extra input tokens that turn out not to affect the final output. By carefully considering exactly when this is possible, it can be shown that the positive inputs to a population protocol (considered as sets of vectors) can be separated into a collection of cones over some finite set of minimal positive inputs, and that each of these cones can be further expressed

using only a finite set of basis vectors. This is sufficient to show that the predicate corresponds to a semilinear set as described above. We thus have:

Theorem 6 ([3,5,7]). *A predicate is computable in the basic population protocol model if and only if it is semilinear.*

Similar results with weaker classes of predicates hold for restricted models with various forms of one-way communication [8]; we describe these results in more detail in Section 4. Indeed, these results were a precursor to the semilinearity theorem of [5]. The journal paper [7] combines and extends these results.

A useful property of Theorem 6 is that it continues to hold unmodified in many simple variants of the basic model. The reason is that any change that weakens the agents can only decrease the set of computable predicates, while any model that is still strong enough to compute congruence modulo k and comparison can still compute all the semilinear predicates. So the semilinear predicates continue to be those that are computable when the inputs are not given immediately but stabilize after some finite time [1] or when one agent in an interaction can see the other's state but not vice versa [7], as in each case it is still possible to compute congruence and threshold in the limit. A similar result holds when a small number of agents can fail [12]; here a slight modification must be made to allow for partial predicates that can tolerate the loss of part of the input. We describe all of these results in later sections.

4 One-way communication

In the basic population protocol model, it is assumed that two interacting agents can simultaneously learn each other's state before updating their own states as a result of the interaction. This requires two-way communication between the two agents. Angluin *et al.* [7] studied several weaker interaction models where, in an interaction, information flows in one direction only. A **receiver** agent learns the state of a **sender** agent, but the sender learns nothing about the state of the receiver. The power of a system with such one-way communication depends on the nature of the communication mechanism.

The model is called a **transmission** model if the sender is aware that an interaction has happened (and can update its own state, although the update cannot depend on the state of the receiver). In an **observation** model, on the other hand, the sender's state is passively observed by the receiver. Another independent attribute is whether an interaction happens instantaneously (**immediate transmission** and **immediate observation** models) or requires some interval of time (**delayed transmission** and **delayed observation** models). The **queued transmission** model is similar to the delayed transmission model, except that receivers

can temporarily refuse incoming messages so that they are not overwhelmed with more incoming information than they can handle. The queued transmission model is the closest to traditional message-passing models of distributed computing.

The weakest of these one-way models is the delayed observation model: protocols can detect whether any particular input symbol is present or absent (and compute any predicate that depends only on this information) simply by observing other agents' input values. Nothing else can be computed: there is no way to distinguish between a sequence of observations of several agents with the same input and a sequence of observations of a single agent.

The immediate observation model is slightly stronger: it can count the number of agents with a particular input, up to some constant threshold. For example, a protocol can determine whether the number of copies of input symbol a is 0, 1, 2, 3 or more than 3. Consequently, any predicate that depends only on this kind of information can be computed. A kind of pumping lemma can be used to show that no other predicates are computable.

Angluin *et al.* also showed that the immediate and delayed transmission models are equivalent in power. They gave a characterization of the computable predicates that shows the power of these models is intermediate between the immediate observation model and the standard two-way model.

Finally, the queued transmission model is equivalent in power to the standard two-way model: any protocol designed for the two-way model can be simulated using queued transmission and vice versa. This holds even though the set of configurations in queued transmission is in principle unbounded; the ability to generate large numbers of buffered messages does not help the protocol, largely because there is no guarantee of where or when they will be delivered.

5 Restricted interaction graphs

In some cases the mobility of agents will have physical limitations, and this will limit the possible interactions that can occur. We can represent this information in an **interaction graph**, where nodes represent agents and edges represent possible interactions. The basic model corresponds to the case where the graph is complete. In this model, a configuration is always represented as a vector of states. (The agents are no longer indistinguishable, so we cannot use a multiset.) If C and C' are configurations, we write $C \rightarrow C'$ if C' can be obtained from C through a single interaction of *adjacent* agents, and the definitions of executions and fairness are as before, using this modified notion of a step.

Having a non-complete (but connected) interaction graph does not make the model any weaker, since adjacent agents can swap states to simulate free movement [3]. For some interaction graphs, the model becomes strictly more power-

ful. For example, consider a straight-line graph. It is not difficult to simulate a linear-space Turing machine by using each agent to represent one square of the Turing machine tape. This allows computation of any function or predicate in LINSPEACE, many of which are not semilinear and thus not computable in the complete interaction graph of the basic model.

In addition to computing predicates on the inputs to agents, it also makes sense in this model to ask whether properties of the interaction graph itself can be computed by the agents in the system. Such problems, which were studied by Angluin *et al.* [1], could have useful applications in determining the network topology induced by an *ad hoc* deployment of mobile agents.

As a simple example, one might want to determine whether the interaction graph has maximum degree k or more, for some fixed k . This can be done by electing a single moving leader token. Initially, all agents hold a leader token. When two leader tokens interact, the tokens coalesce, and when a leader agent interacts with a non-leader agent the leader token may change places. To test the maximum degree, the leader may instead choose to mark up to k distinct neighbors of its current node. By counting how many nodes it successfully marks, the leader can get a lower bound on the degree of the node.

A complication is that the leader has no way to detect when it has interacted with all neighbors of the current node. The best it can do is nondeterministically wait for some arbitrary but finite time before gathering in its marks and trying again. In doing so it relies on the fairness condition to eventually drive it to a state where it has correctly computed the maximum degree. Because the original population model used deterministic transitions, some additional machinery is needed to simulate a nondeterministic transition function by exploiting the nondeterminism of the interaction schedule. Note that the unmarking step does not require nondeterminism: since the leader keeps track of how many marks it has placed, it can simply wait until it has encountered each marked neighbor again.

During the initial leader election phase, two leaders deploying marks could interfere with each other. To handle this, the survivor of any interaction between two leaders collects all outstanding marks from both and resets its degree estimate.

A similar mechanism can be used to assign unique colors to all neighbors of each node in a bounded-degree graph: a wandering **colorizer** token deploys pairs of marks to its neighbors and recolors any it finds with the same color. Once this process converges, the resulting **distance-2 coloring** (so called because all nodes at distance 2 have distinct colors) effectively provides local identifiers for the neighbors of each node. These can be used to carry out arbitrary distributed computations using standard techniques (subject to the $O(1)$ space limit at each node). An example given in [1] is the construction of a rooted spanning tree, which can be used to simulate a Turing machine tape (as in the case of a line graph) by threading the Turing machine tape along a traversal of the tree. It follows

that arbitrary LINSPEACE-computable properties of bounded-degree graphs can be computed by population protocols.

6 Random interactions

An alternative assumption that also greatly increases the power of the model is to replace the adversarial (but fair) scheduler of the basic model with a more constrained interaction pattern. The simplest such variant assumes **random interactions**: each pair of agents is equally likely to interact at each step.

Protocols for random scheduling were given in the initial population protocol paper of Angluin *et al.* [3], based in part on similar protocols for the related model of urn automata [2]. The central observation was that the main limitation observed in trying to build more powerful protocols in the basic model was the inability to detect the absence of agents with a particular state. However, if a single leader agent were willing to wait long enough, it could be assured (with reasonably high probability) that it would meet every other agent in the population, and thus be able to verify the presence or absence of particular token values stored on the other agents by direct inspection. The method used was to have the leader issue a single special marked token to some agent; when the leader encountered this special agent k times in a row it could be reasonably confident that the number of intervening interactions was close to $\Theta(n^{k+1})$. This is sufficient to build unary counters supporting the usual increment, decrement, and zero test operations (the last probabilistic). With counters, a register machine with an $O(\log n)$ bit random-access memory can be simulated using a classic technique of Minsky [23].

The cost of this simulation is a polynomial blowup for the zero test and a further polynomial blowup in the simulation of the register machine. A faster simulation was given by Angluin, Aspnes, and Eisenstat [4], based on epidemics to propagate information quickly through the population. This simulation assumes a single designated leader agent in the initial configuration, which acts as the finite-state controller for the register machine. Register values are again stored in unary as tokens scattered across the remaining agents.

To execute an operation, the leader initiates an epidemic containing an operation code. This opcode is copied through the rest of the population in $\Theta(n \log n)$ interactions on average and with high probability; the latter result is shown to follow by a reduction to a concentration bound for coupon collector due to Kamath *et al.* [21]. Arithmetic operations such as addition, comparison, subtraction, and multiplication and division by constants can be carried out by the non-leader agents in $O(n \log^c n)$ interactions (or $O(\log^c n)$ parallel time units) each. Some of these algorithms are quite simple (adding A to B requires only adding a new B token to each agent that already holds an A token, possibly with an additional

step of unloading extra B tokens onto empty agents to maintain $O(1)$ space per agent), while others are more involved (comparing two values in [4] involves up to $O(\log n)$ alternating rounds of doubling and cancellation, because simply having A and B tokens cancel each other as in Example 2 might require as many as $\Theta(n^2)$ expected interactions for the last few survivors to meet). The most expensive operation is division, at $O(n \log^4 n)$ interactions (or $O(\log^4 n)$ parallel time units).

Being able to carry out individual arithmetic operations is of little use if one cannot carry out more than one. This requires that the leader be able to detect when an operation has finished, which ultimately reduces down to being able to detect when $\Theta(n \log n)$ interactions have occurred. Here the trick of issuing a single special mark is not enough, as the wait needed to ensure a low probability of premature termination is too long.

Instead, a **phase clock** based on successive waves of epidemics is used. The leader starts by initiating a phase 0 epidemic which propagates through the population in parallel to any other activity. When the leader meets an agent that is already infected with phase 0, it initiates a phase 1 epidemic that overwrites the phase 0 epidemic, and similarly with phase 2, 3, and so on, up to some fixed maximum phase $m - 1$ that is in turn overwritten by phase 0 again. Angluin *et al.* show that, while the leader might get lucky and encounter one of a small number of newly-infected agents in a single phase, the more typical case is that a phase takes $\Theta(n \log n)$ interactions before the next is triggered, and over m phases the probability that all are too short is polynomially small. It follows that for a suitable choice of m , the phase clock gives a high-probability $\Theta(n \log n)$ -interaction clock, which is enough to time the other parts of the register machine simulation.

A curious result in [4] is that even though the register machine simulation has a small probability of error, the same techniques can compute semilinear predicates in polylogarithmic expected parallel time with no error in the limit. The trick is to run a fast error-prone computation to get the answer quickly most of the time, and then switch to the result of a slower, error-free computation using the mechanisms of [3] after some polynomially long interval. The high time to converge for the second algorithm is apparent only when the first fails to produce the correct answer; but as this occurs only with polynomially small probability, it disappears in the expectation.

This simulation leaves room for further improvement. An immediate task is to reduce the overhead of the arithmetic operations. In [6], the same authors show how to drop the cost of the worst-case arithmetic operation to $O(n \log^2 n)$ interactions by combining a more clever register encoding with a fast **approximate majority** primitive based on dueling epidemics. This protocol has only three states: the decision values x and y , and b (for “blank”). When an x token meets a y token or vice versa, the second token turns blank. When an x or y token meets a blank

agent, it converts the blank token to its own value. Much of the technical content of [6] involves showing that this process indeed converges to the majority value in $O(n \log n)$ interactions with high probability, which is done using a probabilistic potential function argument separated into several interleaved cases. The authors suggest that simplifying this argument would be a very useful target for future research. It is also possible that further improvements could reduce the overhead for arithmetic operations down to the $O(n \log n)$ interactions needed simply for all tokens to participate.

A second question is whether the distinguished leader in the initial configuration could be replaced. The coalescing leader election algorithm of [3] takes $\Theta(n^2)$ interactions to converge, which may dwarf the time for simple computations. A heuristic leader-election method is proposed in [6] that appears to converge much faster, but more analysis is needed. The authors also describe a more robust version of the phase clock of [4] that, by incorporating elements of the three-state majority protocol, appears to self-stabilize in $O(n \log n)$ interactions once the number of leaders converges to a polynomial fraction, but to date no proof of correctness for this protocol is known.

7 Self-stabilization and related problems

A series of papers [9, 10, 15] have examined the question of when population protocols can be made self-stabilizing [14], or at least can be made to tolerate input values that fluctuate over some initial part of the computation. Either condition is a stronger property than the mere convergence of the basic model, as both require that the population eventually converge to a good configuration despite an unpredictable initial configuration. Many of the algorithms designed to start in a known initial configuration (even if it is an inconvenient one, with, say, all agents in the same state) will not work if started in a particularly bad one. An example is leader election by coalescence: this algorithm can reduce a population of many would-be leaders down to a single unique leader, but it cannot create a new leader if the initial population contains none.

Angluin *et al.* [9] gave the first self-stabilizing protocols for the population protocol model, showing how to carry out various tasks from previous papers without assuming a known initial configuration. These include a distance-2 coloring protocol for bounded-degree graphs based on local handshaking instead of a wandering colorizer token (which is vulnerable to being lost). Their solution has each node track whether it has interacted with a neighbor of each particular color an odd or even number of times; if a node has two neighbors of the same color, eventually its count will go out of sync with that of one or the other, causing both the node and its neighbor to choose new colors. This protocol is applied

in a framework that allows self-stabilizing protocols to be composed, to give additional protocols such as rooted spanning tree construction for networks with a single special node. This last protocol is noteworthy in part because it requires $O(\log D)$ bits of storage per node, where D is the diameter of the network; it is thus one of the earliest examples of pressure to escape the restrictive $O(1)$ -space assumption of the original population protocol model. Other results in this paper include a partial characterization of which network topologies do or do not support self-stabilizing leader election.

This work was continued by Angluin, Fischer, and Jiang [10], who considered the issue of solving the classic **consensus problem** [24] in an environment characterized by unpredictable communication, with the goal of converging to a common consensus value at all nodes eventually (as in a population protocol) rather than terminating with one. The paper gives protocols for solving consensus in this stabilizing sense with both crash and Byzantine failures. The model used deviates from the basic population protocol model in several strong respects: agents have identities (and the $O(\log n)$ -bit memories needed to store them), and though the destinations to which messages are delivered are unpredictable, communication itself is synchronous.

Fischer and Jiang [15] return to the anonymous, asynchronous, and finite-state world of standard population protocols to consider the specific problem of leader election. As observed above, a difficulty with the simple coalescence algorithm for leader election is that it fails if there is no leader to begin with. Fischer and Jiang propose adding to the model a new **eventual leader detector**, called $\Omega?$, which acts as an oracle that eventually correctly informs the agents if there is no leader. Self-stabilizing leader election algorithms based on $\Omega?$ are given for complete interaction graphs and rings. Curiously, the two cases distinguish between the standard global fairness condition assumed in most population protocol work and a local fairness condition that requires only that each action occurs infinitely often (but not necessarily in every configuration in which it is enabled). The latter condition is sufficient to allow self-stabilizing leader election in a complete graph but is provably insufficient in a ring. Many of these results are further elaborated in Hong Jiang's Ph.D. dissertation [20].

8 Larger states

The assumption that each agent can only store $O(1)$ bits of information is rather restrictive. One direction of research is to slowly relax this constraint to obtain other models that are closer to real mobile systems while still keeping the model simple enough to allow for a complete analysis.

Unique identifiers As noted in Section 2, the requirements that population protocols be independent of n and use $O(1)$ space per agent imply that agents cannot have unique identifiers. This contrasts with the vast majority of models of distributed computing, in which processes do have unique identifiers that are often a crucial component of algorithms. Guerraoui and Ruppert investigated a model, called **community protocols**, that preserve the tiny nature of agents in population protocols, but allow agents to be initially assigned unique identifiers drawn from a large set [19]. Each agent is equipped with $O(1)$ memory locations that can each store an identifier. It is assumed that transition rules cannot be dependent on the values of the identifiers: the identifiers are atomic objects that can only be tested for equality with one another. (For example, bitwise operations on identifiers are not permitted.) This preserves the property that protocols are independent of n . They gave the following precise characterization of what can be computed in this model.

Theorem 7 ([19]). *A predicate is computable in the community protocol model if and only if it is in $NSPACE(n \log n)$ and permuting the input characters does not affect the output value.*

The necessity of the second condition (symmetry) follows immediately from the fact that the identifiers cannot be used to order the input symbols. The proof that any computable predicate is in $NSPACE(n \log n)$ uses a non-deterministic search of the graph whose nodes are configurations of the community protocol and whose edges represent transitions between configurations.

Conversely, the proof that any symmetric predicate in $NSPACE(n \log n)$ can be computed by a community protocol uses Schönhage's pointer machines [26] as a bridge. A pointer machine is a sequential machine model that runs a program using only a directed graph structure as its memory. A community protocol can emulate a pointer machine by having each agent represent a node in the graph data structure. Some care must be taken to organize the agents to work together to simulate the sequential machine. It was known that a pointer machine that uses $O(n)$ nodes can simulate a Turing machine that uses $O(n \log n)$ space [27].

It follows that the restriction that agents can use their additional memory space only for storing $O(1)$ identifiers can essentially be overcome: the agents can do just as much as they could if they each had $O(\log n)$ bits of storage that could be used arbitrarily.

Heterogeneous systems One interesting direction for future research is allowing some heterogeneity in the model, so that some agents have more computational power than others. As an extreme example, consider a network of weak sensors that interact with one another, but also with a base station that has unlimited capacity.

Beauquier *et al.* [11] studied a scenario like this, focusing on the problem of having the base station compute n , the number of mobile agents. They replaced the fairness condition of the population protocol model by a requirement that all pairs of agents interact infinitely often. They considered a self-stabilizing version of the model, where the mobile agents are initialized arbitrarily. (Otherwise the problem can be trivially solved by having the base station mark each mobile agent as it is counted.) The problem cannot be solved if each agent's memory is constant size: they proved a tight lower bound of n on the number of possible states the mobile agents must be able to store.

9 Failures

The work described so far assumes that the system experiences no failures. This assumption is somewhat unrealistic in the context of mobile systems of tiny agents and was made to obtain a clean model as a starting point. Some work has studied fault-tolerant population protocols, although this topic is still largely unexplored.

Crash failures. Crash failures are a relatively benign type of failure: faulty agents simply cease having any interactions at some time during the execution. Delporte-Gallet *et al.* [12] examined how crash failures affect the computational power of population protocols. They showed how to transform any protocol that computes a function in the failure-free model into a protocol that can tolerate $O(1)$ crash failures. However, this requires some inevitable weakening of the problem specification.

To understand how the problem specification must change when crash failures are introduced, consider the majority problem described in Example 2. We saw that this problem is solvable if there are no failures. Now consider a version of the majority problem where up to 5 agents may crash. Consider an execution with m followers and $m + 5$ leaders. According to the original problem specification, the output of any such execution must be 1. Suppose, however, that the agents associated with 5 of the $m + 5$ leaders crash before having any interactions. There is no way that the non-faulty agents can distinguish such an execution from a failure-free execution involving m followers and m leaders. In the latter execution, the output must be 0. So, the majority problem, in its original form, cannot be solved when crash failures occur. Nevertheless, we can solve a closely related problem. Suppose that we impose preconditions on the problem, requiring that the margin of the majority is at least 5. More precisely, we impose the requirement on the inputs that either the number of leaders exceeds the number of followers by more than 5 or the number of followers exceeds the number of leaders by at least 5. Under this precondition, it can be shown that the majority problem becomes

solvable even when up to 5 agents may crash.

The above example can be generalized in a natural way: If we wish to solve a problem in a way that tolerates up to f crash failures, where f is a constant, we must impose a precondition that says the removal of f of the input values cannot change the output value. To prove that this is sufficient to make the predicate computable in a fault-tolerant way (assuming that the original predicate is computable in the failure-free model), Delporte-Gallet *et al.* [12] designed an automatic transformation that converts a protocol P for the failure-free model into a protocol P' that will tolerate up to f failures.

The transformation uses replication. In P' , agents are divided (in a fault-tolerant way) into $\Theta(f)$ groups, each of size $\Theta(n/f)$. Each group simulates an execution of P on the entire set of inputs. Each agent of P' can store, in its own memory, the simulated states of $O(f)$ agents of P , since f is a constant, so each group of $\Theta(n/f)$ agents has sufficient memory space to collectively simulate all agents of P . To get a group's simulation started, agents within the group gather the initial states (in P) of *all* agents. Up to f agents may crash before giving their initial states to anyone within that group, but the precondition ensures that this will not affect the output of the simulated run. Thus, any group whose members do not experience any crashes will eventually produce the correct output. The number of groups is chosen to ensure that a majority of the groups run correct simulations, and this allows each agent to determine the correct output.

A variant of the simulation handles a combination of a finite number of transient failures (where an agent spontaneously changes state) and crash failures [12]. It can also be used in the community protocol model described in Section 8 [19].

Byzantine failures. An agent that has a Byzantine failure may behave arbitrarily: it can interact with all other agents, pretending to be in any state for each interaction. This behavior can cause havoc in a population protocol since none of the usual techniques used in distributed computing to identify and contain the effects of Byzantine agents can be used. Indeed, it is known that no non-trivial predicate can be computed by a population protocol in a way that tolerates even one Byzantine agent [19]. Two ways of circumventing this fact have been studied.

In the community protocol model of Section 8, if we assume that agent identifiers cannot be tampered with, then some failure detection is possible. Guerraoui and Ruppert give a protocol that solves the majority problem, tolerating a constant number of Byzantine failures, if the margin of the majority is sufficiently wide [19].

Byzantine agents also appear in the random-scheduling work of [6], where it is shown that the approximate majority protocol quickly converges to a configuration in which nearly all non-faulty agents possess the correct decision value despite the actions of a small ($o(\sqrt{n})$) minority of Byzantine agents. Here there is

no extension of the basic population protocol model to include identifiers, but the convergence condition is weak, and the Byzantine agents can eventually—after exponential time—drive the protocol to any configuration, including stable configurations in which no agent holds a decision value. Determining the full power of random scheduling in the presence of Byzantine agents remains open.

References

- [1] D. Angluin, J. Aspnes, M. Chan, M. J. Fischer, H. Jiang, and R. Peralta. Stably computable properties of network graphs. In *Proc. Distributed Computing in Sensor Systems: 1st IEEE International Conference*, pages 63–74, 2005.
- [2] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Urn automata. Technical Report YALEU/DCS/TR-1280, Yale University Department of Computer Science, Nov. 2003.
- [3] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, Mar. 2006.
- [4] D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. In *Proc. Distributed Computing, 20th International Symposium*, pages 61–75, Sept. 2006.
- [5] D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing*, pages 292–299, 2006.
- [6] D. Angluin, J. Aspnes, and D. Eisenstat. A simple protocol for fast robust approximate majority. In *Proc. Distributed Computing, 21st International Symposium*, pages 20–32, 2007.
- [7] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*. To appear; published online in 2007.
- [8] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. On the power of anonymous one-way communication. In *Proc. Principles of Distributed Systems, 9th International Conference*, pages 396–411, 2005.
- [9] D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang. Self-stabilizing population protocols. In *Proc. Principles of Distributed Systems, 9th International Conference*, pages 103–117, 2005.
- [10] D. Angluin, M. J. Fischer, and H. Jiang. Stabilizing consensus in mobile networks. In *Proc. Distributed Computing in Sensor Systems, 2nd IEEE International Conference*, pages 37–50, 2006.
- [11] J. Beauquier, J. Clement, S. Messika, L. Rosaz, and B. Rozoy. Self-stabilizing counting in mobile sensor networks. Technical Report 1470, LRI, Université Paris-Sud 11, 2007.

- [12] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *Proc. 2nd IEEE International Conference on Distributed Computing in Sensor Systems*, pages 51–66, 2006.
- [13] Z. Diamadi and M. J. Fischer. A simple game for the study of trust in distributed systems. *Wuhan University Journal of Natural Sciences*, 6(1–2):72–82, Mar. 2001. Also appears as Yale Technical Report TR–1207, Jan. 2001.
- [14] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.
- [15] M. J. Fischer and H. Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *Proc. Principles of Distributed Systems, 10th International Conference*, pages 395–409, 2006.
- [16] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
- [17] D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188:404–425, 1992.
- [18] S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16:285–296, 1966.
- [19] R. Guerraoui and E. Ruppert. Even small birds are unique: Population protocols with identifiers. Technical Report CSE-2007-04, Department of Computer Science and Engineering, York University, 2007.
- [20] H. Jiang. *Distributed Systems of Simple Interacting Agents*. PhD thesis, Yale University, 2007.
- [21] A. P. Kamath, R. Motwani, K. Palem, and P. Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. *Random Structures and Algorithms*, 7:59–80, 1995.
- [22] T. G. Kurtz. *Approximation of Population Processes*. Number 36 in CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 1981.
- [23] M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., 1967.
- [24] M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, Apr. 1980.
- [25] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes-Rendus du I Congrès de Mathématiciens des Pays Slaves*, pages 92–101, Warszawa, 1929.
- [26] A. Schönhage. Storage modification machines. *SIAM J. Comput.*, 9(3):490–508, Aug. 1980.
- [27] P. van Emde Boas. Space measures for storage modification machines. *Inf. Process. Lett.*, 30(2):103–110, Jan. 1989.

THE FORMAL LANGUAGE THEORY COLUMN

BY

ARTO SALOMAA

Turku Centre for Computer Science and,
Department of Mathematics, University of Turku
FIN-20014 Turku, Finland
asalomaa@utu.fi

DECISION ALGORITHMS FOR SUBFAMILIES OF REGULAR LANGUAGES USING STATE-PAIR GRAPHS

Yo-Sub Han*

Intelligence and Interaction Research Center
Korea Institute of Science and Technology
P.O.BOX 131, Cheongryang, Seoul, Korea
emmous@kist.re.kr

Abstract

We survey recent results on decision algorithms for subfamilies of regular languages. In particular, we look at the decision algorithms using state-pair graphs constructed from finite-state automata. The algorithms rely on the structural property of a finite-state automaton that is preserved in its state-pair graph. We also review applications of state-pair graphs in different subfamilies of regular languages.

*Han was supported by the KIST Tangible Space Initiative Grants 2E20050 and 2Z03050.

1 Introduction

There are many subfamilies of formal languages. Example are recursively enumerable languages, context-sensitive languages, context-free languages and regular languages. A subfamily sometimes includes another subfamily. For instance, among the four example subfamilies, each subfamily includes the following subfamilies in order and this gives rise to the Chomsky hierarchy [6]. Furthermore, a subfamily has many (often infinite) subfamilies depending on how to define subfamilies. For context-free languages, for example, there is an infinite hierarchy for $LL(k)$ languages and all $LL(k)$ languages are a proper subfamily of context-free languages [1]. Given a family \mathcal{L} of languages and a language L , the *decision problem* of L with respect to \mathcal{L} is to decide whether or not L belongs to \mathcal{L} .

In this column, we consider the decision problem of subfamilies of regular languages. Regular languages have many different subfamilies; for example, finite languages, one-unambiguous regular languages [4], block-deterministic regular languages [11] and so on. We investigate subfamilies of regular languages that are defined by code properties such as prefix-freeness, suffix-freeness or infix-freeness. Note that codes have been used in many different areas; for example, information processing, data compression, cryptography and information transmission [24]. Codes are categorized with respect to different conditions according to the applications. For instance, prefix-freeness establishes prefix-free codes¹. In regular languages, prefix-freeness defines a subfamily $\mathcal{L}_{(p,r)}$, prefix-free regular languages, where all languages are prefix-free sets and regular. Namely,

$$\mathcal{L}_{(p,r)} = \{L \mid L \text{ is regular and prefix-free.}\}$$

Similarly, we can define suffix-free, bifix-free, infix-free and outfix-free regular languages. Most of the decision problems related to code properties are decidable for regular languages whereas they often become undecidable for context-free languages [24]. We study decision algorithms for these subfamilies of regular languages. We examine algorithms that use a particular graph, *state-pair graph*.

A state-pair graph is a directed graph computed from a finite-state automaton A using pairs of states and pairs of transitions in A . We review the basic concepts of state-pair graphs and decision algorithms for various subfamilies of regular languages that are defined by code properties. We emphasize the link between the structural property of state-pair graphs and these subfamilies of regular languages.

We define some basic notions in Section 2 and recall the formal definition of a state-pair graph in Section 3. We review decision algorithms for prefix-free, suffix-free and infix-free regular languages in Section 4. Then, we look at k -intercode regular languages in two different cases in Section 5; 1) k is fixed and

¹In the literature, prefix-free codes are often called prefix codes.

2) k is unknown, where k is an index. Lastly, we turn to two finite languages, hypercodes and outfix-free regular languages in Section 6 and conclude the column in Section 7.

2 Preliminaries

Let Σ be a finite alphabet of characters and Σ^* be the set of all strings over Σ . The number of characters in Σ is denoted by $|\Sigma|$. A language over Σ is any subset of Σ^* . The symbol \emptyset denotes the empty language and the symbol λ denotes the null string. Given a string x from a set X , let x^R be the reversal of x , in which case $X^R = \{x^R \mid x \in X\}$.

A finite-state automaton (FA) A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. Let $|Q|$ be the number of states in Q and $|\delta|$ be the number of transitions in δ . Then, the size $|A|$ of A is $|Q| + |\delta|$. Given a transition $\delta(p, a) = q$, we say that p has an *out-transition* and q has an *in-transition*. Furthermore, p is a *source state* of q and q is a *target state* of p . We say that A is *non-returning* if the start state of A does not have any in-transitions and A is *non-exiting* if all final states of A do not have any out-transitions. In the following, we always assume that A has only *useful* states; that is, each state of A appears in some path from the start state to some final state.

Given an FA $A = (Q, \Sigma, \delta, s, F)$ and a state $q \in Q$, we define the *right FA* $A_{\vec{q}}$ to be $(Q, \Sigma, \delta, q, F)$; namely, we make q to be the start state. Then, the *right language* $L_{\vec{q}}$ of q is the set of strings accepted by $A_{\vec{q}}$.

Definition 1 is a list of codes that we use to define subfamilies of regular languages in the following sections.

Definition 1. A language L is

- prefix-free if, for all distinct strings $x, y \in \Sigma^*$, $x \in L$ and $y \in L$ imply that x and y are not prefixes of each other.
- suffix-free if, for all distinct strings $x, y \in \Sigma^*$, $x \in L$ and $y \in L$ imply that x and y are not suffixes of each other.
- bifix-free if L is prefix-free and suffix-free.
- infix-free if, for all distinct strings $x, y \in \Sigma^*$, $x \in L$ and $y \in L$ imply that x and y are not substrings of each other.
- outfix-free if, for all distinct strings $x, y, z \in \Sigma^*$, $xz \in L$ and $xyz \in L$ imply $y = \lambda$.

- a intercode of index k (or a k -intercode) if $L^{k+1} \cap \Sigma^+ L^k \Sigma^+ = \emptyset$.
- a hypercode if, for all distinct strings $x, y \in \Sigma^*$, $x \in L$ and $y \in L$ imply that x and y are not subsequences of each other.

A regular language L is prefix-free if L is a prefix-free set. We say that an FA A is prefix-free if $L(A)$ is prefix-free. We can establish similar notions for the other code sets in Definition 1.

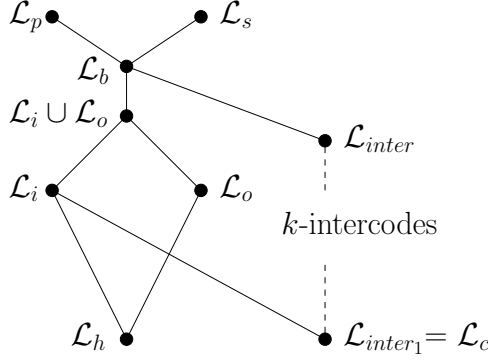


Figure 1: The families of languages defined by code properties in Definition 1. Solid lines indicate proper inclusions and a dotted line denotes a proper hierarchy. The diagram does not, in general, indicate intersections or unions. The full diagram with more code families can be found in Jürgensen and Konstantinidis [24].

For all unexplained notions related to formal languages, refer to the textbooks [21, 30]. For more details on coding theory, refer to Berstel and Perrin [3] or Jürgensen and Konstantinidis [24].

3 State-pair graphs

FAs are the basic model used to represent regular languages in many applications. FAs are essentially labeled directed graphs and each path from a start state to a final state spells out an accepted string. There are two well-known families of FAs in the literature: the Thompson automata [29] and the position automata [13, 26]. One advantage of using such families of FAs is that these automata preserve the structural properties of corresponding regular expressions. Caron and Ziadi [5] studied the structural properties of the position automata and Giammarresi et al. [12] examined the structural properties of the Thompson automata.

On the other hand, if we manipulate FAs, then these FAs easily lose certain structural properties; for example, if we concatenate a position automaton and a Thompson automaton, then the resulting automaton does not preserve either the position automaton properties or the Thompson automaton properties. Nevertheless, one property remains unchanged in FAs: a path from a start state to a final state spells out an accepted string. The use of state-pair graphs relies on this fact. Applications of state-pair graphs have been already investigated earlier by Berstel and Perrin [3], where this notion is called the square of an automaton.

We first recall the definition of a state-pair graph and its complexity from Han et al. [17]. Given an FA $A = (Q, \Sigma, \delta, s, F)$, we assign a unique number for each state from 1 to m , where 1 denotes the start state and $m = |Q|$. If A has a single final state, then we assume that m denotes the final state.

Definition 2. Given an FA $A = (Q, \Sigma, \delta, s, F)$, we define the state-pair graph $G_A = (V_G, E_G)$ of A , where V_G is a set of nodes and E_G is a set of labeled edges, as follows:

$$V_G = \{(i, j) \mid i, j \in Q\} \text{ and}$$

$$E_G = \{((i, j), a, (x, y)) \mid \delta(i, a) = x, \delta(j, a) = y \text{ and } a \in \Sigma\}.$$

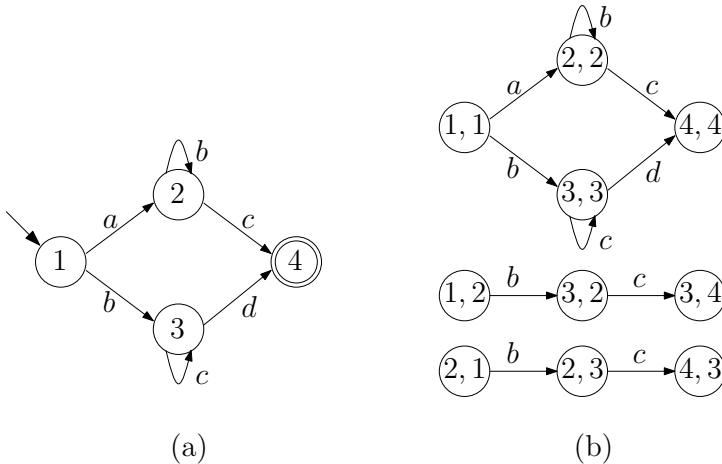


Figure 2: (a) is an FA A and (b) is the corresponding state-pair graph G_A . We omit all nodes without transitions in G_A .

For FAs with λ -transitions, we use $a \in \Sigma \cup \{\lambda\}$ for computing E_G of G_A . The crucial property of G_A is that if there is a string w spelled out by two distinct paths

in A , for example, one path is from i to p and the other path is from j to q , then, there is a path from (i, j) to (p, q) in G_A that also spells out the same string w . The complexity of G_A is as follows:

Since we compute all pairs of states from A ,

$$|V_G| = |Q|^2. \quad (1)$$

Let $|\delta_i|$ be the number of out-transitions from state i in A . Then, $|\delta| = \sum_{i=1}^m |\delta_i|$, where $m = |Q|$. Since a node (i, j) in G_A can have at most $|\delta_i| \times |\delta_j|$ out-transitions,

$$|E_G| = \sum_{i,j=1}^m |\delta_i| \times |\delta_j| \leq |\delta|^2. \quad (2)$$

Therefore, by (1) and (2), G_A has at most $|Q|^2$ nodes and $|\delta|^2$ edges.

Proposition 3. *Given an FA $A = (Q, \Sigma, \delta, s, F)$ and its state-pair graph G_A , $|G_A| \leq |Q|^2 + |\delta|^2$. Namely, $|G_A| = O(|A|^2)$.*

Note that the construction of G_A does not require an input FA A to be deterministic. In the following sections, we present recent results using state-pair graphs for determining subfamilies of regular languages.

4 Prefix-free, suffix-free and infix-free regular languages

We first examine three well-known codes. Prefix-freeness and suffix-freeness are symmetric. A language L is prefix-free if and only if L^R is suffix-free. Infix-freeness is stronger than both prefix-freeness and suffix-freeness as shown in Fig. 1; if L is infix-free, then L is always prefix-free and suffix-free.

4.1 Prefix-free and suffix-free regular languages

Prefix-freeness has already been used in the literature. Prefix-freeness defines Huffman codes [22] and *determinism* for generalized automata [10] and for expression automata [19]. Recently, Han et al. [16] considered prefix-free regular expressions as patterns in text searching and designed an efficient algorithm for the prefix-free regular-expression matching problem based on prefix-freeness.

If a given FA is deterministic, then it is easy to verify the prefix-freeness of $L(A)$; $L(A)$ is prefix-free if and only if A is non-exiting [3, 19]. On the other hand, if A is nondeterministic, then the condition that A is non-exiting is only necessary but not sufficient. Thus, we have to check whether or not $L(A) \cap L(A)\Sigma^+ = \emptyset$. Let us

examine what $L(A) \cap L(A)\Sigma^+ = \emptyset$ implies in state-pair graphs. If $L(A) \cap L(A)\Sigma^+ \neq \emptyset$, then there are two distinct strings $w_1 = xy$ and $w_2 = x$ for some strings x and y , where $x, y \neq \lambda$. Since w_1 and w_2 have a common prefix x , there is a path from $(1, 1)$ to (m, j) such that $j \neq m$. Based on this observation, Han et al. [16] established the following result.

Theorem 4. *Given an FA $A = (Q, \Sigma, \delta, s, f)$, $L(A)$ is prefix-free if and only if there is no path from $(1, 1)$ to (m, j) , for any $j \neq m$, in G_A , where 1 denotes the start state and $m = |Q|$ denotes the final state.*

In Theorem 4, we implicitly assume that A has a single final state. This assumption is valid since a prefix-free FA must be non-exiting and, thus, all final state are equivalent and mergible into a single state. Using Theorem 4, they proposed a prefix-freeness checking algorithm for an FA.

```

Prefix-Freeness( $A = (Q, \Sigma, \delta, s, f)$ )

  if  $A$  is not non-exiting
    then return no
  Construct  $G_A = (V_G, E_G)$  from  $A$ 

  DFS((1, 1)) in  $G_A$ 
  if we meet a node  $(m, j)$  for some  $j, j \neq m$ 
    then return no

  return yes

```

Figure 3: A prefix-freeness checking algorithm.

The sub-function DFS((1, 1)) in Prefix-Freeness (PF) in Fig. 3 is a depth-first search that starts at node $(1, 1)$ in G_A . The construction $G_A = (V, E)$ from A takes $O(|Q|^2 + |\delta|^2)$ time and DFS takes $O(|V| + |E|)$ time. Therefore, the total running time for PF is $O(|Q|^2 + |\delta|^2)$. For details on DFS, refer to the textbook [8]. Note that PF takes an input as FAs. Thus, if a regular language is given by a regular expression E , then we can construct the Thompson automaton A_E for E since $|A_E| = O(|E|)$ [21, 29]. Now PF guarantees the following result.

Theorem 5. *Given an FA A , we can determine the prefix-freeness of $L(A)$ in $O(|A|^2)$ worst-case time.*

Next, we consider suffix-freeness. Since L is prefix-free if and only if L^R is suffix-free by definition, we can establish the following statement from Theorems 4 and 5.

Theorem 6. *Given an FA A , $L(A)$ is suffix-free if and only if there is no path from $(1, i)$ to (m, m) , for any $i \neq 1$, in G_A . Moreover, we can determine the suffix-freeness of $L(A)$ in $O(|A|^2)$ worst-case time.*

To be precise for Theorem 6, we have to transform an FA with multiple final states into an FA with a single final state before computing its state-pair graph: We introduce a new final state f' and make f' to be a target state of all final states by a λ -transition. Then, we change all final states except for f' to non-final states.

A language L is bifix-free if and only if L is prefix-free and suffix-free. Since we can verify prefix-freeness and suffix-freeness in quadratic time, we can also decide bifix-freeness in the same runtime using state-pair graphs.

Proposition 7. *Given an FA A , we can determine the bifix-freeness of $L(A)$ in $O(|A|^2)$ worst-case time using its state-pair graph.*

4.2 Infix-free regular languages

We turn to infix-freeness. Infix-free languages have been used in text searching [7, 16] and computing forbidden words [2, 9]. Ito et al. [23] showed that it is decidable whether or not a given regular language is infix-free and recently, Béal et al. [2] proposed a polynomial-time algorithm that determines infix-freeness for DFAs. We review the infix-freeness decision algorithm for general FAs based on state-pair graphs designed by Han et al. [17].

Given an FA A , if $L(A)$ is not infix-free, then there are two distinct strings w_1 and w_2 accepted by A and w_2 is an infix of w_1 . This implies that there are two distinct paths in A that spell out w_1 and w_2 , respectively, and the path for w_1 has a subpath that spells out w_2 .

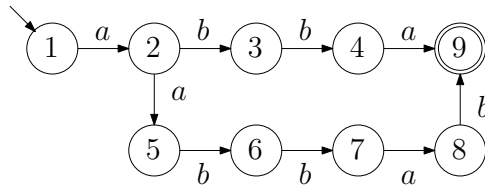


Figure 4: Two strings $abba$ and $aabbab$ are spelled out by two distinct paths.

In Fig. 4, for example, the FA accepts $w_1 = aabbab$ and $w_2 = abba$ and the subpath $q_2 \rightarrow q_5 \rightarrow q_6 \rightarrow q_7 \rightarrow q_8$ of the path for w_1 also spells out w_2 . We can identify such strings in $L(A)$ from its state-pair graph since both strings have a common substring.

Theorem 8. *Given an FA $A = (Q, \Sigma, \delta, s, f)$, $L(A)$ is infix-free if and only if there is no path from $(1, i)$ to (m, j) apart from $(1, 1)$ to (m, m) , where $1 \leq i \leq m$, $1 \leq j \leq m$, 1 denotes the start state and $m = |Q|$ denotes the final state. Moreover, we can determine the infix-freeness of $L(A)$ in $O(|A|^2)$ worst-case time.*

The proof for Theorem 8 can be found in Han et al. [17].

5 Interodes

While comma-free languages have not been studied to the extent of prefix-free languages in the literature, the comma-free property was already introduced in 1958 [14]. Furthermore, Shyr and Yu [27] introduced *interodes*, as a generalization of comma-free codes, see also Yu [31]. Comma-free codes are the interodes of index one. Jürgensen et al. [25] have studied the decidability of the interode property.

Note that if an index k is given, then we can fairly easily check whether or not L is an interode of index k . However, if no index is given, then the problem is not as straightforward. Jürgensen et al. [25] established that it is decidable whether or not a given regular language is an interode (of any index). There the complexity of the decision algorithm is not discussed explicitly, but it is easy to verify that an algorithm derived from the construction of the decidability proof is not a polynomial-time algorithm in the general case where the input language is specified by an NFA.

Recently, Han et al. [15] designed an algorithm that determines whether or not a given regular language L is an interode (of any index) using state-pair graphs. The algorithm runs in polynomial time for both DFAs and NFAs. Besides having better time complexity, the algorithm is conceptually easier to understand and implement compared with the algorithm derived from Jürgensen et al. [25].

Note that interode (regular) languages are a proper subfamily of bifix-free (regular) languages as shown in Fig. 1. Thus, if a given FA A is not bifix-free, then we immediately know that $L(A)$ is not an interode. Therefore, we can assume that a given FA A is bifix-free and this guarantees that

1. A is non-returning and non-exiting.
2. A has a single final state.

From such an FA $A = (Q, \Sigma, \delta, s, f)$, we can construct an FA A^2 for the language $L(A)L(A)$ by merging f of the first copy of A and s of the second copy of A . The FA A^2 has $2|Q| - 1$ states and $2|\delta|$ transitions; namely, $|A^2| < 2|A|$. We can repeat this procedure to construct an FA for the catenation of several A 's. We use

A^k to denote the FA for the catenation of k copies of A and A_i to denote the i th component A of A^k , for $1 \leq i \leq k$. We use (i, j) in A^k to denote the state i in A_j of A^k .



Figure 5: An example of an FA for the catenations of $k+1$ As.

The following results have been proved in Han et al. [15].

Lemma 9. *Given an FA A and an index k , we can determine whether or not $L(A)$ is a k -intercode in $k^2 \cdot O(|A|^2)$ worst-case time.*

Corollary 10. *Given an FA A , we can determine the comma-freeness of $L(A)$ in $O(|A|^2)$ worst-case time since a k -intercode for $k = 1$ is a comma-free code.*

Han et al. [15] also tackled the case when k is unknown. Instead of trying all possible indices, which is very inefficient, they discovered that if $L(A)$ is not a k -intercode for a certain constant k , then $L(A)$ is not an intercode for any index.

Lemma 11. *Given an FA A , $L(A)$ is not an intercode for any index k if $L(A)$ is not a $(m+1)$ -intercode, where m is the number of states in A .*

Using Lemmas 9 and 11, the decision problem for intercode regular languages can be solved as follows:

Theorem 12. *Given an FA A , we can determine whether or not $L(A)$ is an intercode of index k , for some k , in $O(|A|^4)$ worst-case time.*

The family of intercode (regular) languages has a proper hierarchy as illustrated in Fig. 1. Thus, if a regular language L is identified as a k -intercode, L may be a $(k-1)$ -intercode as well. This leads to a new problem that computes the smallest k such that $L(A)$ is a k -intercode but not a $(k-1)$ -intercode in polynomial time. Based on Theorem 12, Han et al. [15] suggested a polynomial-time algorithm that relies on the binary search approach.

Theorem 13. *Given an FA A , in $O(\log |Q| \cdot |A|^4)$ worst-case time, we can determine whether or not $L(A)$ is an intercode for some index $k > 0$, and if the answer is positive we can find the smallest index l such that $L(A)$ is an l -intercode but not an $(l-1)$ -intercode.*

6 Hypercodes and outfix-free regular languages

So far, all subfamilies of regular languages in the preceding sections can be infinite. We now consider two subfamilies of regular languages that are always finite; hypercodes and outfix-free regular languages.

6.1 Hypercodes

A set X of strings is a hypercode if a string in X is not a subsequence of any other string in X . Based on hypercodes, Head and Thierrin [20] derived properties of OL languages. Hypercodes are a proper subfamily of outfix-free languages. Moreover, hypercodes are always finite [28]. Since hypercodes are finite, we can decide whether or not a given finite set of strings is a hypercode by comparing all pairs of strings in the set, although it is certainly undesirable to do so. We look at an efficient algorithm for the decision problem. Since an FA A for a hypercode must be non-exiting and $L(A)$ must be finite, we assume that A has a single final state and has no back transitions that make cycles.

Given an FA $A = (Q, \Sigma, \delta, s, f)$, we assign a unique number for each state in A from 1 to m , where $m = |Q|$. We construct a new FA A' from A by duplicating A and adding a self-loop with Σ to all states. Namely, $A' = (Q, \Sigma, \delta', s, f)$, where

$$\delta' = \delta \cup \{(q, \Sigma, q) \mid q \in Q\}.$$

We introduce a new state-pair graphs from A and A' as follows:

Definition 14. *Given an FA $A = (Q, \Sigma, \delta, s, f)$ and an FA $A' = (Q, \Sigma, \delta', s, f)$, we define the state-pair graph $G_A = (V_G, E_G)$, where V_G is a set of nodes and E_G is a set of edges, as follows:*

$$V_G = \{(i, j) \mid i \in Q_A \text{ and } j \in Q_{A'}\} \text{ and}$$

$$E_G = \{((i, j), a, (x, y)) \mid \delta(i, a) = x \text{ and } \delta'(j, a) = y \text{ and } a \in \Sigma\},$$

where Q_A denotes Q of A and $Q_{A'}$ denotes Q of A' .

Note that the only difference between the new state-pair graph and the previous state-pair graph in Section 3 is that the new graph is constructed from two similar yet different FAs whereas the previous graph is constructed from a single FA. The complexity of the new graph is same as before; $|G_A| = O(|A|^2)$.

Fig. 6 illustrates a state-pair graph constructed as in Definition 14. The language $L(A) = \{abc, ac\}$ is not a hypercode since ac is a subsequence of abc whose path is $(1, 1) \rightarrow (2, 4) \rightarrow (3, 4) \rightarrow (5, 5)$ in G_A . Then, state-pair graphs ensure the following result:

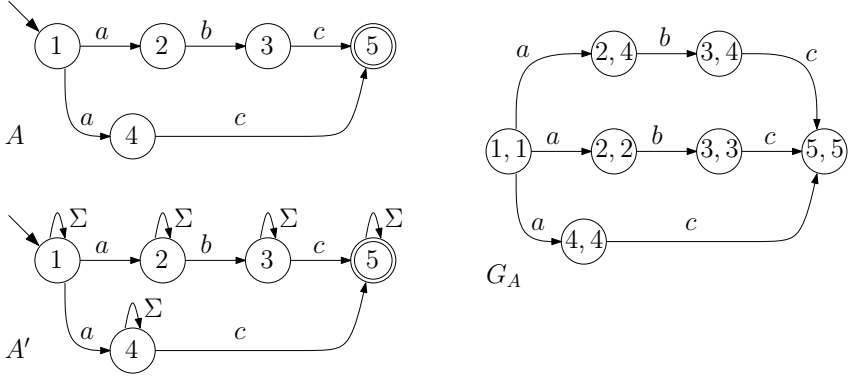


Figure 6: Given an FA A , we construct a new FA A' for deciding whether or not $L(A)$ is a hypercode. G_A is the corresponding state-pair graph. Note that $L(A) = \{abc, ac\}$ is not a hypercode since ac is a subsequence of abc . We omit all nodes that do not appear in any path from $(1, 1)$ to $(5, 5)$ in G_A .

Theorem 15. *Given an FA $A = (Q, \Sigma, \delta, s, f)$, $L(A)$ is a hypercode, if and only if the state-pair graph G_A for A has no path $(i_1, j_1) \rightarrow (i_2, j_2) \rightarrow \dots \rightarrow (i_k, j_k)$ that satisfies the following conditions:*

1. $(i_1, j_1) = (1, 1)$ and $(i_k, j_k) = (m, m)$.
2. *there exists at least one pair of two adjacent nodes $(i_u, j_u) \rightarrow (i_{u+1}, j_{u+1})$ such that $j_u = j_{u+1}$ for $1 \leq u < k$.*

Theorem 15 shows that given an FA A , we can check whether or not $L(A)$ is a hypercode in $O(|A|^2)$ worst-case time using its state-pair graph and DFS.

6.2 Outfix-free regular languages

We turn to outfix-freeness. Assume that we have two distinct strings w_1 and w_2 and w_2 is an outfix of w_1 . This implies that $w_1 = xyz$ for some strings x, y and z such that $w_2 = xz$ and $y \neq \lambda$. Moreover, w_1 and w_2 have a common prefix x and a common suffix z . Fig. 7 illustrates such w_1 and w_2 .

Based on this property, Han and Wood [18] investigated the case when a finite language L is given by a finite set of strings; $L = \{w_1, w_2, \dots, w_n\}$. Note that a finite set of strings is often stored in a trie, which is an ordered tree data structure that is used to store a set of strings and each edge in the tree has a single character label.

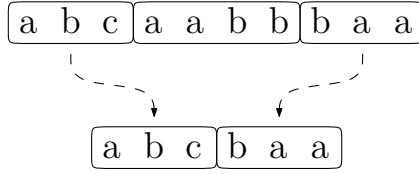


Figure 7: A graphical illustration of an outfix string; $abcbaa$ is an outfix of $abcaabbbaa$.

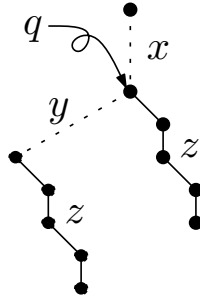


Figure 8: An example of a trie for strings $w_1 = xyz$ and $w_2 = xz$. Note that both paths end with the same subpath sequence in the trie because of a common suffix z .

Once we construct a trie for L , then two strings w_1 and w_2 share a common path from the root if they have a common prefix. See Fig. 8 for an example. Therefore, we only need to check whether or not a sub-trie of T is suffix-free. If a sub-trie is not suffix-free, then L is not outfix-free. By carefully analyzing this procedure, Han and Wood [18] designed an efficient decision algorithm and proved its correctness.

Theorem 16. *Given a finite set $L = \{w_1, w_2, \dots, w_n\}$ of strings, we can determine whether or not L is outfix-free in $O(\sum_i^n |w_i|^2)$ time using $O(\sum_i^n |w_i|)$ space in the worst-case.*

Now consider two strings $w_1 = xyz$ and $w_2 = xz$ in DFAs. If a DFA $A = (Q, \Sigma, \delta, s, f)$ accepts both w_1 and w_2 , then there is a unique path from s to a state q that spells out x , which is a common prefix of w_1 and w_2 . Then, $A_{\neg q}$ accepts yz and z . This implies that $L_{\neg q}$ is not suffix-free.

Proposition 17. *Given a DFA $A = (Q, \Sigma, \delta, s, f)$, $L(A)$ is outfix-free if and only if $L_{\vec{q}}$ is suffix-free for all $q \in Q$.*

However, if a given FA is nondeterministic, then Proposition 17 does not hold anymore. Nevertheless, if an NFA A is not outfix-free, then its state-pair graph has a similar property to the property used in Proposition 17. To make use of this property, Han and Wood [18] introduced a *state-pair DFA* for A from its state-pair graph G_A as follows:

Definition 18. *Given an FA $A = (Q, \Sigma, \delta, s, f)$, we define the state-pair graph $G_A = (V_G, E_G)$, where V_G is a set of nodes and E_G is a set of edges, as follows:*

$$V_G = \{(i, j) \mid i \text{ and } j \in Q\} \text{ and}$$

$$E_G = \{((i, j), a, (x, y)) \mid \delta(i, a) = x \text{ and } \delta(j, a) = y \text{ and } a \in \Sigma\}.$$

Then, we define a new DFA A' by making $(1, 1)$ to be the start state and (m, m) to be the final state and removing all non-reachable states from $(1, 1)$ in G_A . We call A' a state-pair DFA.

We can decide the outfix-freeness of $L(A)$ using its state-pair DFA.

Lemma 19. *Given an FA $A = (Q, \Sigma, \delta, s, f)$, $L(A)$ is outfix-free if and only if $L_{\vec{p}} \cup L_{\vec{q}}$ is suffix-free for all pair states $(p, q) \in Q'$ of its state-pair DFA $A' = (Q', \Sigma, \delta', s', f')$, where $L_{\vec{q}}$ is the right language of state q in A .*

Han and Wood [18] gave a proof for Lemma 19 and proposed an algorithm that checks the outfix-freeness of $L(A)$ as follows; from A and its state-pair DFA $A' = (Q', \Sigma, \delta', s', f')$, we check whether or not $L_{\vec{p}} \cup L_{\vec{q}}$ is suffix-free for all state $(p, q) \in Q'$. Since $L_{\vec{q}}$ in A is computed from its right FA $A_{\vec{q}}$, we construct an FA $B = (Q_B, \Sigma, \delta_B, s_B, f_B)$ for $L_{\vec{p}} \cup L_{\vec{q}}$ from $A_{\vec{p}} = (Q_p, \Sigma, \delta_p, p, f)$ and $A_{\vec{q}} = (Q_q, \Sigma, \delta_q, q, f)$, where

$$Q_B = \{s_B, f_B\} \cup Q_p \cup Q_q,$$

$$\delta_B = \{(s_B, \lambda, p), (s_B, \lambda, q), (f, \lambda, f_B)\} \cup \delta_p \cup \delta_q.$$

Since $O(|B|) = O(|A_{\vec{p}}| + |A_{\vec{q}}|) = O(|A|)$, this algorithm runs in polynomial time and gives the following result [18].

Theorem 20. *Given an FA $A = (Q, \Sigma, \delta, s, f)$, we can determine the outfix-freeness of $L(A)$ in $O(|A|^4)$ worst-case time.*

7 Conclusions

An FA A for a regular language L is more than just an acceptor for L ; A preserves the structural property of L . State-pair graphs were proposed to make use of these properties in FAs. We have surveyed a few number of decision algorithms using state-pair graphs for subfamilies of regular languages defined by code properties. We want to remark that all presented algorithms run in polynomial time. However, we do not know if there exist better time complexity algorithms for any subfamily of regular languages that has been considered. For instance, it is open if we can determine the prefix-freeness of $L(A)$ in subquadratic time.

References

- [1] A. Aho and J. Ullman. *The Theory of Parsing, Translation, and Compiling, Vol. I: Parsing*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972.
- [2] M.-P. Béal, M. Crochemore, F. Mignosi, A. Restivo, and M. Sciortino. Computing forbidden words of regular languages. *Fundamenta Informaticae*, 56(1-2):121–135, 2003.
- [3] J. Berstel and D. Perrin. *Theory of Codes*. Academic Press, Inc., 1985.
- [4] A. Brüggemann-Klein and D. Wood. One-unambiguous regular languages. *Information and Computation*, 140:229–253, 1998.
- [5] P. Caron and D. Ziadi. Characterization of Glushkov automata. *Theoretical Computer Science*, 233(1–2):75–90, 2000.
- [6] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956.
- [7] C. L. A. Clarke and G. V. Cormack. On the use of regular expressions for searching text. *ACM Transactions on Programming Languages and Systems*, 19(3):413–426, 1997.
- [8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.
- [9] M. Crochemore, F. Mignosi, and A. Restivo. Automata and forbidden words. *Information Processing Letters*, 67(3):111–117, 1998.
- [10] D. Giammarresi and R. Montalbano. Deterministic generalized automata. *Theoretical Computer Science*, 215:191–208, 1999.
- [11] D. Giammarresi, R. Montalbano, and D. Wood. Block-deterministic regular languages. In *Proceedings of ICTCS'01*, Lecture Notes in Computer Science 2202, 184–196, 2001.
- [12] D. Giammarresi, J.-L. Ponty, D. Wood, and D. Ziadi. A characterization of Thompson digraphs. *Discrete Applied Mathematics*, 134:317–337, 2004.

- [13] V. Glushkov. The abstract theory of automata. *Russian Mathematical Surveys*, 16:1–53, 1961.
- [14] S. Golomb, B. Gordon, and L. Welch. Comma-free codes. *The Canadian Journal of Mathematics*, 10:202–209, 1958.
- [15] Y.-S. Han, K. Salomaa, and D. Wood. Intercode regular languages. *Fundamenta Informaticae*, 76(1-2):113–128, 2007.
- [16] Y.-S. Han, Y. Wang, and D. Wood. Prefix-free regular-expression matching. In *Proceedings of CPM'05*, Lecture Notes in Computer Science 3537, 298–309, 2005.
- [17] Y.-S. Han, Y. Wang, and D. Wood. Infix-free regular expressions and languages. *International Journal of Foundations of Computer Science*, 17(2):379–393, 2006.
- [18] Y.-S. Han and D. Wood. Outfix-free regular languages and prime outfit-free decomposition. *Fundamenta Informaticae*. To appear.
- [19] Y.-S. Han and D. Wood. The generalization of generalized automata: Expression automata. *International Journal of Foundations of Computer Science*, 16(3):499–510, 2005.
- [20] T. Head and G. Thierrin. Hypercodes in deterministic and slender 0L languages. *Information and Control*, 45(3):251–262, 1980.
- [21] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 2 edition, 1979.
- [22] D. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40:1098–1101, 1952.
- [23] M. Ito, H. Jürgensen, H.-J. Shyr, and G. Thierrin. Outfix and infix codes and related classes of languages. *Journal of Computer and System Sciences*, 43:484–508, 1991.
- [24] H. Jürgensen and S. Konstantinidis. Codes. In G. Rozenberg and A. Salomaa, editors, *Word, Language, Grammar*, volume 1 of *Handbook of Formal Languages*, 511–607. Springer-Verlag, 1997.
- [25] H. Jürgensen, K. Salomaa, and S. Yu. Decidability of the intercode property. *Elektronische Informationsverarbeitung und Kybernetik*, 29(6):375–380, 1993.
- [26] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IEEE Transactions on Electronic Computers*, 9:39–47, 1960.
- [27] H. Shyr and S. Yu. Intercode and some related properties. *Soochow J. Math.*, 16(1):95–107, 1990.
- [28] H.-J. Shyr and G. Thierrin. Hypercodes. *Information and Control*, 24(1):45–54, 1974.
- [29] K. Thompson. Regular expression search algorithm. *Communications of the ACM*, 11:419–422, 1968.
- [30] D. Wood. *Theory of Computation*. John Wiley & Sons, Inc., New York, NY, 1987.
- [31] S. Yu. A characterization of intercodes. *International Journal of Computer Mathematics*, 36:39–45, 1990.

THE FORMAL SPECIFICATION COLUMN

BY

HARTMUT EHRRIG

Technical University of Berlin, Department of Computer Science
Franklinstraße 28/29, D-10587 Berlin, Germany
ehrig@cs.tu-berlin.de

MODEL TRANSFORMATIONS BY GRAPH TRANSFORMATION ARE FUNCTORS

Hartmut Ehrig¹, Karsten Ehrig², Claudia Ermel¹, Ulrike Prange¹

¹Fac. of Electr. Engineering and Comp. Science
Technical University of Berlin, Germany
ehrig|lieske|uprange@cs.tu-berlin.de

² Department of Computer Science
University of Leicester, United Kingdom
karsten@mcs.le.ac.uk

Abstract

In this paper, we extend our ideas on model transformations as functors discussed in the previous issue and embed this concept into the framework of graph transformation systems. We show that under certain restrictions of the rules model transformations by graph transformation are functors.

Introduction

In our previous column [2] we have discussed the claim that "model transformations should be functors" from a mathematical and from a practical point of view. Especially interesting from the practical side was the proposal of the POPL'07

keynote speaker Don Batory [1] that model transformations in his approach of Feature Oriented Model Driven Development (FOMDD) should be functors.

On the theoretical side, we have discussed well-known data type constructions which are functors and can be considered as model transformations. Typical examples are the semantics of parameterized specification and module specification (see [4, 5]). Moreover we have claimed that under suitable conditions model transformations defined by graph transformations are functors. In this column, we justify this last claim.

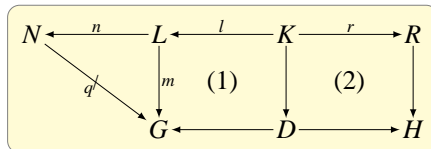
1 Model Transformation by Graph Transformation

In this section, we define model transformation by graph transformation. First we shortly introduce the necessary definitions of typed graphs, rules and transformations (see [3]).

A graph $G = (V, E, src, tar)$ is given by sets V and E of nodes and edges, respectively, and source and target functions $src, tar : E \rightarrow V$. For graphs $G^i = (V^i, E^i, src^i, tar^i)$ with $i = 1, 2$, a graph morphism $f = (f_V, f_E) : G^1 \rightarrow G^2$ is given by mappings $f_V : V^1 \rightarrow V^2$ and $f_E : E^1 \rightarrow E^2$ compatible with the source and target functions, i.e. $f_V \circ src^1 = src^2 \circ f_E$ and $f_V \circ tar^1 = tar^2 \circ f_E$. Graphs and graph morphisms form the category **Graphs**.

A type graph TG is a distinguished graph that defines node and edge types. A typed graph (G, t) is a graph G together with a typing morphism $t : G \rightarrow TG$. If the typing is clear in the context, we denote G as a typed graph without explicitly naming t . A typed graph morphism f between typed graphs (G^1, t^1) and (G^2, t^2) is a graph morphism $f : G^1 \rightarrow G^2$ compatible with the typing, i.e. $t^2 \circ f = t^1$. For a type graph TG , typed graphs and typed graph morphisms form the category **Graphs_{TG}**.

A typed graph rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ consists of three typed graphs L , K and R , called left hand side, gluing graph and right hand side, respectively, and injective typed graph morphisms l and r . Given a rule p and a typed graph morphism $m : L \rightarrow G$, called match, the application of the rule p to G via the match m is given by the following two pushouts (1) and (2) leading to the direct typed graph transformation $G \xRightarrow{p, m} H$. A sequence $G_0 \Rightarrow G_1 \Rightarrow \dots \Rightarrow G_n$ of direct transformations is then called transformation and denoted by $G_0 \xRightarrow{*} G_n$.



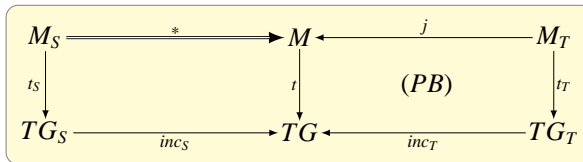
A negative application condition (NAC) for p is a typed graph morphism $n : L \rightarrow N$. The match m satisfies the NAC n if there does not exist an injective morphism $q : N \rightarrow G$ such that $q \circ n = m$. For p we define a set NAC_p of NACs. The application of p with NACs is allowed only if m satisfies all NACs $n \in NAC_p$.

A graph transformation system $GTS = (TG, Prod)$ is now given by a type graph TG and a set $Prod$ of rules with NACs. For graph transformation systems there are many interesting results like the Local Church–Rosser, Parallelism, Embedding, Extension and Local Confluence Theorems [3].

In the following, we only consider nondeleting rules. In this case we have $L = K$ and the rule is completely defined by the morphism r and its NACs.

For the definition of a model transformation by graph transformation, the source and target models have to be given as typed graphs typed over TG_S and TG_T , respectively, where TG_S is the type graph defining the source language L_S and TG_T is the type graph defining the target language L_T . Performing model transformations by typed graph transformations means taking a model as a typed graph and transforming it according to certain rules. The result is a typed graph which represents the target model.

For the model transformation, we define a type graph TG with $TG_S \subseteq TG$ and $TG_T \subseteq TG$. This type graph includes not only TG_S and TG_T , but may contain additional node and edge types which are needed during the transformation process. Due to the inclusion, all source and target models are also automatically typed over TG . Given a graph transformation system $GTS = (TG, Prod)$, for a source model M_S we start the model transformation by applying the rules in $Prod$ as long as possible. If this process terminates, it results in a transformation $M_S \xRightarrow{*} M$ with a graph M typed over TG . M contains the target model, but also the source model and possibly additional nodes and edges. To obtain the target model, we restrict M to the target type graph TG_T by constructing the pullback (PB). The pullback object $M|_{TG_T} = M_T$ is our target model and correctly typed over TG_T .



Given the graph transformation system $GTS = (TG, Prod)$, we define the model transformation $MT : \mathbf{L}_S \Rightarrow \mathbf{L}_T$, where $\mathbf{L}_S \subseteq \mathbf{Graphs}_{TG_S}$ and $\mathbf{L}_T \subseteq \mathbf{Graphs}_{TG_T}$ are subcategories with $M_S \in \mathbf{L}_S$ and $M_T \in \mathbf{L}_T$ if and only if $M_S \xRightarrow{*} M$ is a terminating transformation with $M|_{TG_T} = M_T$, and all injective morphisms.

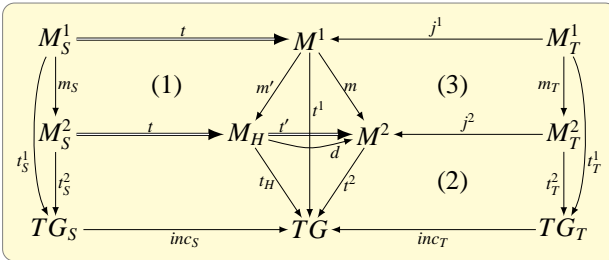
In general, there may be different terminating transformations leading to target models M_T^1 and M_T^2 for the same source models M_S . If $MT(M_S)$ is uniquely

defined, we say MT has functional behaviour.

2 Model Transformation as a Functor

In this section, we consider a model transformation $MT : \mathbf{L}_S \Rightarrow \mathbf{L}_T$ with functional behaviour given by a graph transformation system $GTS = (TG, Prod)$ as described in Section 1. In addition, we restrict the rules in $Prod$ to $TG \setminus TG_S$ -generating rules, where for a rule $p : L \xrightarrow{r} R$ there are only elements $x \in R \setminus r(L)$ with type $t^R(x) \in TG \setminus TG_S$, and define that $NAC_p = \{r\}$, i.e. each rule $p : L \xrightarrow{r} R$ has only one NAC, which is exactly the right hand side. With these restrictions, we want to show that MT becomes a functor.

First, we have to define $MT(m_S)$ for an injective morphism $m_S : M_S^1 \rightarrow M_S^2$ in \mathbf{L}_S . For M_S^1 , we have a transformation sequence $M_S^1 \xRightarrow{t} M^1$ and a restriction M_T^1 leading to the model transformation $MT(M_S^1) = M_T^1$. Since only nondeleting rules are applied in t , m_S is boundary-consistent w.r.t. t [6], i.e. the transformation sequence does not delete any boundary element of n . Moreover, due to the restrictions of the NACs and m_S being injective, m_S is also NAC-consistent w.r.t. t [6], which means that no NAC of t is violated by extending M_S^1 to M_S^2 via m_S . Thus we can apply the Embedding Theorem with NACs [6] leading to a transformation sequence $M_S^2 \xRightarrow{t} M_H$ with a morphism $m' : M^1 \rightarrow M_H$ as shown in diagram (1). Then we apply the rules of our graph transformation as long as possible leading to a transformation sequence $M_H \xRightarrow{t'} M^2$. Since we have only nondeleting rules there is a morphism $d : M_H \rightarrow M^2$ with $t^2 \circ d = t_H$, and we define $m = d \circ m'$. The restriction of M^2 leads to the pullback (2) with pullback object M_T^2 . This pullback and $t^2 \circ m \circ j^1 = inc_T \circ t_T^1$ imply that there exists a unique $m_T : M_T^1 \rightarrow M_T^2$ such that (3) commutes and $t_T^2 \circ m_T = t_T^1$, and by pullback decomposition also (3) is a pullback. Now we define $MT(m_S) = m_T$.

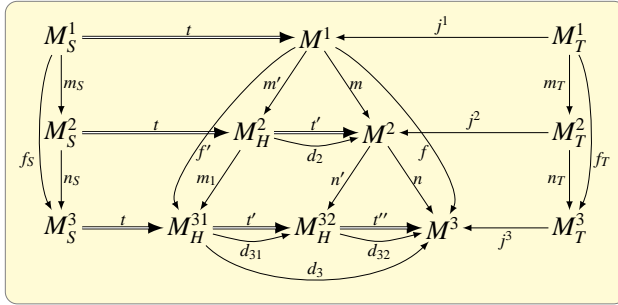


For MT to be a functor we have to show first that $MT(n_S \circ m_S) = MT(n_S) \circ MT(m_S)$ and second that $MT(id_{M_S}) = id_{MT(M_S)}$.

1. Consider the following diagram with

$$n_S \circ m_S = f_S, MT(m_S) = m_T, MT(n_S) = n_T \text{ and } MT(f_S) = f_T,$$

as well as the morphisms m, m', n, n' and f, f' of the construction with $d_2 \circ m' = m, d_{32} \circ n' = n$ and $d_3 \circ f' = f$, respectively. The functional behaviour of MT implies that we can construct the stepwise embedding $M_S^3 \xRightarrow{t} M_H^{31} \xRightarrow{t'} M_H^{32} \xRightarrow{t''} M^3$, since n_S is consistent w.r.t. t and m_1 is consistent w.r.t. t' , and we have that $m_1 \circ m' = f', n' \circ d_2 = d_{31} \circ m_1$ and $d_{32} \circ d_{31} = d_3$. Combining these results we have that $n \circ m = f$. Pullback composition and the uniqueness of pullbacks further implies that $f_T = n_T \circ m_T$, i.e. $MT(n_S \circ m_S) = MT(n_S) \circ MT(m_S)$.



2. $MT(id_{M_S}) = id_{MT(M_S)}$ follows from the functional behaviour of MT .

In the following, we present a model transformation from statecharts to Petri nets (see [3]). In Fig. 1, the integration of the type graphs for the model transformation is shown. In the left hand side, the source model type graph for statecharts is depicted. In the right hand side, the target model type graph for Petri nets is shown. Together with some additional nodes and edges they form the integrated type graph.

The rules for the model transformation are given in Figs. 2 and 3. Each state in the statechart is transformed to a corresponding place in the target Petri net model, where a token in such a place denotes that the corresponding state is active initially (rules `InitState2Place` and `State2Place`). A separate place is generated for each valid event by rule `Event2Place`. Each step in the statechart is transformed into a Petri net transition (rule `Step2Trans`). Naturally, the Petri net should simulate how to exit and enter the corresponding states in the statechart, and therefore input and output arcs of the transition have to be generated accordingly (see rules `StepFrom2PreArc` and `StepTo2PostArc`). Furthermore, firing a transition should consume the token of the trigger event (`Trigger2PreArc`), and

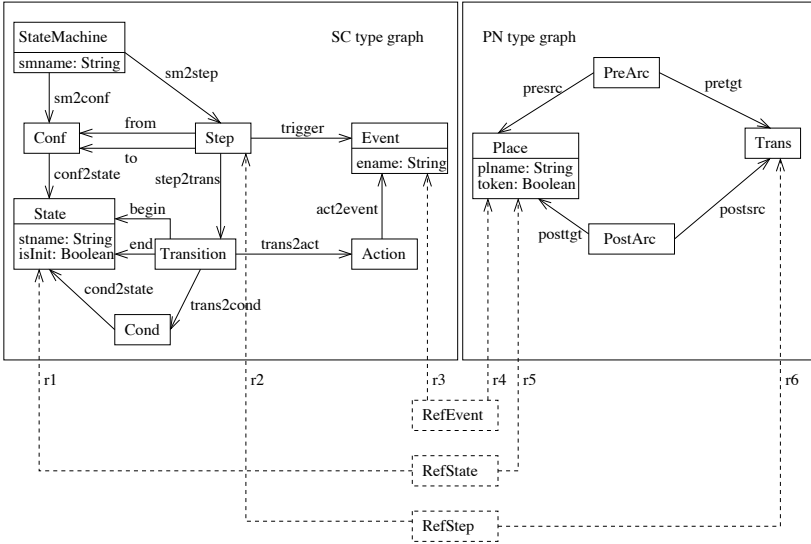


Figure 1: The integration of the type graphs

should generate tokens to (the places related to) the target event indicated as the action (Action2PostArc). All these rules are nondeleting, $TG \backslash TG_S$ -generating and we have $NAC_p = \{r\}$. Moreover, the model transformation has functional behaviour (see [3]), thus we can apply the developed theory and conclude that this model transformation is a functor.

In Fig. 4, the application of the model transformation to the statecharts SC_1 and SC_2 is shown leading to the Petri nets PT_1 and PT_2 , respectively. For the morphism $f_S : SC_1 \rightarrow SC_2$ we get a corresponding morphism $f_T : PT_1 \rightarrow PT_2$.

3 Conclusion

In this column we have continued the discussion of the last column concerning the claim that "model transformations should be functors". We have shown that a suitable class of model transformations based on graph transformation defines a functor between the corresponding visual languages. It remains open to analyze more general classes of model transformations and morphisms between visual models, and to show which kinds of functors can be obtained in these cases.

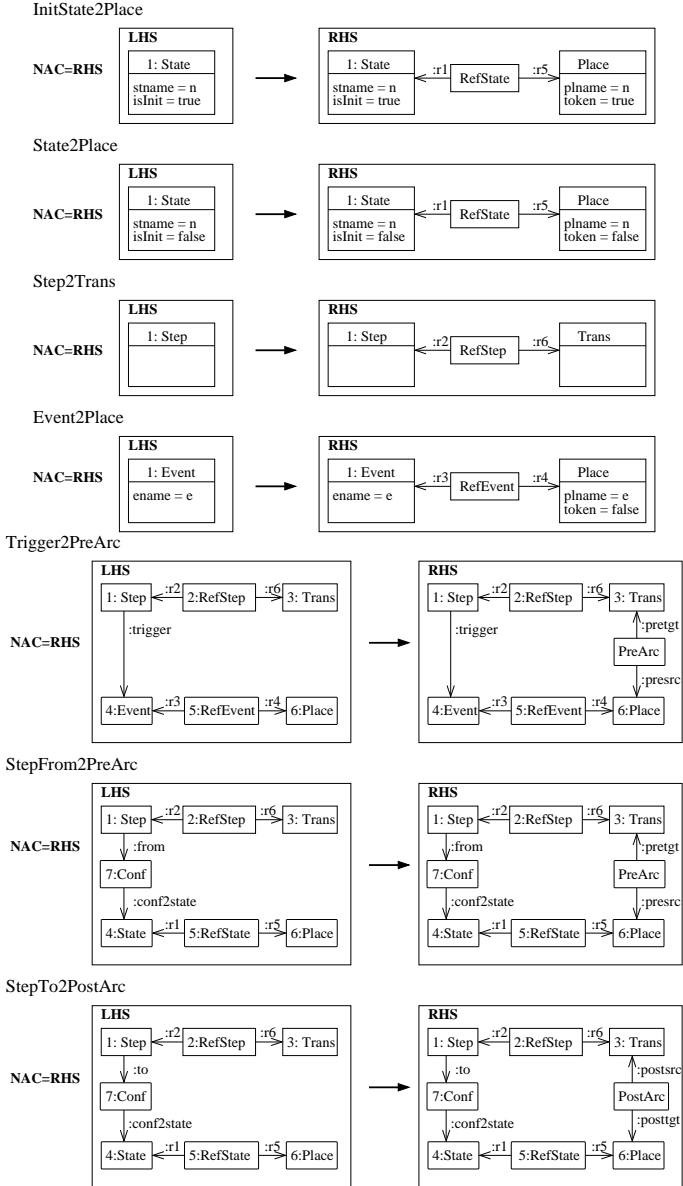


Figure 2: The rules for the model transformation (1)

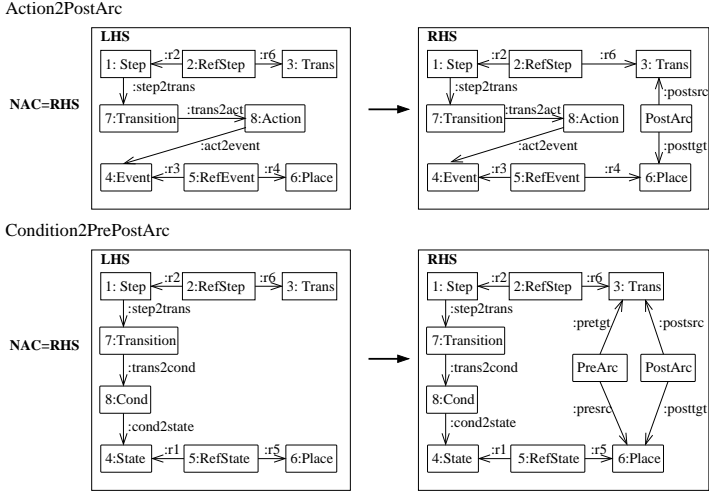


Figure 3: The rules for the model transformation (2)

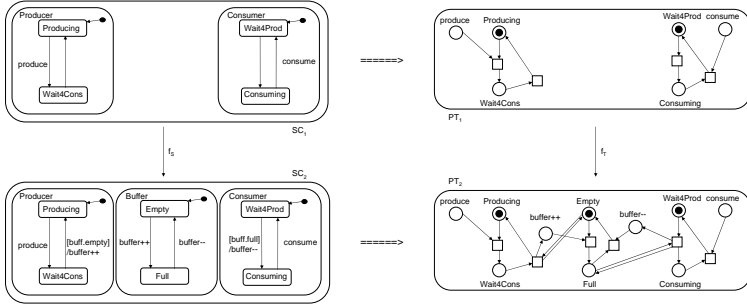


Figure 4: The model transformation and the translated morphism

References

- [1] D. Batory. 2007. *From Implementation to Theory in Product Synthesis*. POPL 2007, Keynote Speech.

- [2] D. Batory, O. Diaz, H. Ehrig, C. Ermel, U. Prange, and G. Taentzer, 2007. *Model Transformations Should Be Functors*. Bulletin of the EATCS, 92:75-81.
- [3] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, 2006. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science, Springer.
- [4] H. Ehrig and B. Mahr, 1985. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of EATCS Monographs on Theoretical Computer Science, Springer.
- [5] H. Ehrig and B. Mahr, 1990. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*, volume 21 of EATCS Monographs on Theoretical Computer Science, Springer.
- [6] L. Lambers, 2007. *Adhesive High-level Replacement Systems with Negative Application Conditions*. Technical Report 2007/14, Technical University of Berlin.

THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

YURI GUREVICH

Microsoft Research
One Microsoft Way, Redmond WA 98052, USA
gurevich@microsoft.com

PROOF INTERPRETATIONS AND THE COMPUTATIONAL CONTENT OF PROOFS IN MATHEMATICS

Ulrich Kohlenbach*

1 Introduction

This survey reports on some recent developments in the project of applying proof theory to proofs in core mathematics. The historical roots, however, go back to Hilbert's central theme in the foundations of mathematics which can be paraphrased by the following question

“How is it that abstract methods (‘ideal elements’) can be used to prove ‘real’ statements e.g. about the natural numbers and is this use necessary in principle?”

Hilbert's original aim, to show the consistency of the use of such ideal elements by finitistic means, which would suffice to eliminate these ideal elements from proofs of purely universal (‘real’) theorems (‘Hilbert's program’), turned out to be impossible for theories containing a sufficient amount of number theory by K. Gödel's 2nd incompleteness theorem. Nevertheless, various partial realizations of

*Fachbereich Mathematik, Technische Universität Darmstadt, Schlossgartenstraße 7, D-64289 Darmstadt, Germany, kohlenbach@mathematik.tu-darmstadt.de

this program and many relative consistency proofs could be achieved. One class of tools used to obtain this are so-called proof interpretations I which transform proofs p in theories \mathcal{T} of theorems A into new proofs p^I in theories \mathcal{T}^I of the interpretation A^I of A . If $(0 = 1)^I \equiv (0 = 1)$, then this yields a consistency proof for \mathcal{T} relative to the assumed consistency of \mathcal{T}^I . Whereas, Hilbert's program focusses exclusively on (ideal proofs of) purely universal theorems, a natural 'shift of emphasis' (G. Kreisel) is to try to apply proof interpretations to interesting proofs of existential theorems with the goal to extract new information from the proof which was not visible beforehand.¹ This new information often consists of effective data such as algorithms or effective bounds (extracted from *prima facie* ineffective proofs) but also continuous dependence or even full independence of solutions from certain parameters (uniformity). Another aspect is the generalization of proofs by a weakening of the premises.

Already in the 50's G. Kreisel had asked

‘What more do we know if we have proved a theorem by restricted means than if we merely know that it is true?’

Kreisel proposed to apply proof theoretic techniques – originally developed for foundational purposes – to concrete proofs in mathematics which mathematicians could not ‘unwind’ themselves (see e.g. [78, 85, 27] and - more recently - [86]).

Kreisel's idea of **unwinding proofs**, i.e. the logical analysis of proofs using techniques from proof theory, has been applied e.g. to number theory ([77, 84]), combinatorics ([32, 3, 47, 100]) (though the latter two papers also apply combinatorics to proof theory) and algebra ([19, 20, 21, 26]). During the last 10-15 years, however, the most systematic development of such an applied proof theory (also called ‘Proof Mining’) took place in connection with applications to numerical analysis (approximation theory) and functional analysis (nonlinear analysis, fixed point theory and ergodic theory), see e.g. [1, 14, 15, 16, 30, 55, 61, 62, 64, 70, 71, 72, 69, 73, 74, 83]. The area of analysis and, in particular, functional analysis seems specially suited for this approach as here issues of representations of analytical objects (usually not made explicit in mathematics) play a crucial role and are systematically addressed by techniques such as proof interpretations.

Moreover, in the context of applications to functional analysis, new logical metatheorems have been developed which not only a priori guarantee the extractability of effective bounds but also qualitatively new uniformity results (see [63, 31, 82]). These metatheorems and the resulting applications to concrete proofs in analysis are all based on suitable extensions and refinements of so-called functional interpretations which have the root in Gödel's ‘Dialectica Interpretation’ ([36, 68]). In

¹As stressed by Kreisel, for this applied purpose, proofs of universal lemmas do not matter at all (but only their truth) so that such lemmas may be taken simply as axioms. So Kreisel's emphasis is sort of opposite to that in Hilbert's program.

particular, a monotone functional interpretation due to the author [56] is crucially used.

Recently, Terence Tao ([98, 99]) arrived, prompted by some of his famous work² on the use of ergodic theory in combinatorics and number theory, at a proposal of so-called ‘hard analysis’ (as opposed to ‘soft analysis’) which roughly can be understood as carrying out analysis on the level of uniform bounds in the sense of monotone functional interpretation which in many cases allows one to ‘finitize’ analytic assumptions and to arrive at qualitatively stronger results. Indeed, one of the main benefits of the metatheorems proved in [63, 31] is that they allow one to do exactly this (e.g. it is shown how to remove assumptions like the existence of fixed points etc. in proofs). Tao illustrates his ideas using two examples: a finite convergence principle and a ‘finitary’ infinite pigeonhole principle. We indicate below how the former and a variant of the latter directly result from monotone functional interpretation.

In this survey we sketch some of the central techniques for applied proof theory and list a few keynote applications in various areas of mathematics. As to the proof theoretic methods we discuss we focus (rather than aiming at a complete list of the various techniques that have been developed over the years) on those which have been instrumental in finding new mathematical results (when applied to specific proofs).

Since the number of applications in functional analysis is rather large and we just published a comprehensive survey on the results obtained up to 2006 in [67] we will mainly restrict our selection in this area to some recent results from 2007 (not covered in [67]).

Obviously, a short paper like this can only touch in a very superficial way most of the issues it addresses. For a more serious and comprehensive treatment of both the logical aspects of proof interpretations as well as their uses in mathematics we have to refer the reader to the forthcoming book [66].

For a proper understanding of the rest of this article we presuppose some basic knowledge of first order logic and type systems as well as of basic notions from abstract analysis such as metric, normed and Hilbert spaces while all other concepts used will be defined. Some of the results are formulated in the general context of so-called hyperbolic spaces (which comprises both normed spaces as well as important structures frequently used in geometric group theory such as CAT(0)-spaces in the sense of Gromov). However, one can get a proper understanding of the main results already by just replacing ‘hyperbolic space’ everywhere by ‘con-

²Tao was awarded a fields medal for this work in 2006. One particularly celebrated result is his ergodic theoretic proof (together with B. Green) that there are arithmetic progressions of arbitrary length in the prime numbers.

vex subset of a normed space' and $'(1 - \lambda)x \oplus \lambda y'$ by $'(1 - \lambda)x + \lambda y'$.

Notation: Throughout this paper $\mathbb{N} := \{0, 1, 2, \dots\}$.

2 Some tools from proof theory

One of the oldest tools from proof theory, which has been effectively used by H. Luckhardt in the unwinding of proofs of the famous finiteness theorem of Roth in diophantine approximation (see section 3 below), is Herbrand's theorem which we formulate here only for the case of Π_3^0 -sentences (which covers finiteness statements such as the one proved by Roth).

Let

$$A \equiv \forall x \exists y \forall z A_{qf}(x, y, z)$$

be sentence where A_{qf} is quantifier-free.

The so-called Herbrand normal form A^H of A is defined as

$$A^H := \forall x \exists y A_{qf}(x, y, f(y)),$$

where f is a new function symbol (also called index function). Herbrand's theorem states two things:

- 1) From a given proof of A in predicate logic without equality one can extract finitely many terms t_1, \dots, t_n which are built up out of x, f and the constants occurring in A such that

$$A^{H,D} := \bigvee_{i=1}^n A_{qf}(x, t_i, f(t_i))$$

is a tautology.

- 2) There is a direct derivation (using only appropriate quantifier introduction rules and contractions) from any disjunction of the form $A^{H,D}$ to A .

Remark 2.1. *Over 2nd order logic where we can quantify over functions and can write A^H as $\forall f, x \exists y A_{qf}(x, y, f(y))$, the Herbrand normal form A^H and A can be shown to be equivalent (using the axiom of choice).*

Let us illustrate things using the following logically valid sentence of the required form

$$A := \forall x \exists y \forall z (P(x, y) \vee \neg P(x, z)),$$

where

$$A^H \equiv \forall x \exists y (P(x, y) \vee \neg P(x, f(y))).$$

Whereas no single term instantiated for ‘ $\exists y$ ’ produces a tautology, two terms $t_1 := x, t_2 := f(x)$ will do since

$$A^{H,D} := (P(x, x) \vee \neg P(x, f(x))) \vee (P(x, f(x)) \vee \neg P(x, f(f(x))))$$

is a tautology.

For the 2nd claim in Herbrand’s theorem one argues as follows: $A^{H,D}$ remains being a tautology if we replace the f -terms starting from the term of greatest depth successively by new variables resulting in

$$A^D := (P(x, x) \vee \neg P(x, y)) \vee (P(x, y) \vee \neg P(x, z))$$

from which we arrive back to A by an obvious direct proof (first introducing the quantifiers in the 2nd disjunct and then the ones in the first disjunct and finally applying a contraction).

It is an easy exercise to show that in general for sentences $A \equiv \forall x \exists y \forall z A_{qf}(x, y, z)$, A^D can always be written in the linearly ordered form

$$(L) (A_{qf}(x, t_1, b_1) \vee A_{qf}(x, t_2, b_2) \vee \dots \vee A_{qf}(x, t_k, b_k),$$

where the b_i are new variables and t_i does not contain any b_j with $i \leq j$ (see [77]). It is this form of Herbrand’s theorem, which – appropriately reformulated – also extends theories having only purely universal axioms, that is used by H. Luckhardt.

For theories which logically complex axioms (e.g. general induction axioms) such as Peano Arithmetic PA, Herbrand’s theorem does not extend. However, the formulation $A^{H,D}$ suggests the following generalization: suppose that we work in a theory with decidable prime formulas (and hence decidable quantifier-free formulas) such as PA. Then the above Herbrand disjunction $A^{H,D}$ can be written as (using function quantifiers)

$$\forall f A_{qf}(x, \Phi(x, f), f(\Phi(x, f))),$$

with

$$\Phi(x, f) := t_i[x, f],$$

where $1 \leq i \leq n$ is least such that

$$A_{qf}(x, t_i[x, f], f(t_i[x, f]))$$

holds. If one is working in a system containing some arithmetic and interested only in a **bound** on a number quantifier ‘ $\exists y$ ’ in A^H one can take simply

$$\Phi^*(x, f) := \max\{t_1[x, f], \dots, t_n[x, f]\}$$

which is even independent from the prime formulas in A .

Whereas for theorems A proved in theories \mathcal{T} having only universal axioms (so-called ‘open’ theories) it suffices to use terms t_i that are built of the A^H - and \mathcal{T} -material, so that for Φ, Φ^* in addition to this only definition-by-case functions resp. a maximum function are required, it is clear that for more complicated theories \mathcal{T} more complicated classes of functionals Φ are required. This leads to the following definition

Definition 2.2 (Kreisel [75, 76]). *A computable functional $\Phi : \mathbb{N} \times \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ satisfies the no-counterexample interpretation (short: n.c.i.) of a sentence*

$$A \equiv \forall x \exists y \forall z A_{qf}(x, y, z) \in \mathcal{L}(\text{PA})$$

if

$$\forall x \in \mathbb{N} \forall f \in \mathbb{N}^{\mathbb{N}} A_{qf}(x, \Phi(x, f), f(\Phi(x, f)))$$

is true.

We illustrate the no-counterexample interpretation by the following example used already by G. Kreisel in [75] and which recently received new attention by T. Tao’s discussion in [98]: Let (a_n) be a nonincreasing sequence in $[0, 1]$. Then, clearly, (a_n) is convergent and so a Cauchy sequence. For convenience, we express this fact in the following form

$$(1) \forall k \in \mathbb{N} \exists n \in \mathbb{N} \forall m \in \mathbb{N} \forall i, j \in [n; n + m] (|a_i - a_j| \leq 2^{-k}),$$

where $[n; n + m] := \{n, n + 1, \dots, n + m\}$.

Then (treating for the moment \leq between real numbers as a primitive predicate and disregarding the bounded quantifiers ‘ $\forall i, j \in [n; n + m]$ ’) the Herbrand normal form of this statement is

$$(2) \forall k \in \mathbb{N} \forall g \in \mathbb{N}^{\mathbb{N}} \exists n \in \mathbb{N} \forall i, j \in [n; n + g(n)] (|a_i - a_j| \leq 2^{-k}).$$

By the well-known counterexamples due to E. Specker (‘Specker sequences’) there exist easily computable such sequences (a_n) even of rational numbers for which there is no computable bound on ‘ $\exists n$ ’ in (1). By contrast, there is a simple primitive recursive (in the sense of [51]) functional $\Phi^*(g, k)$ which provides a bound on (2) (also referred to as ‘metastability’ in Tao [98]):

Proposition 2.3 (see e.g. [66]). *Let (a_n) be any nonincreasing sequence in $[0, 1]$ then*

$$\forall k \in \mathbb{N} \forall g \in \mathbb{N}^{\mathbb{N}} \exists n \leq \Phi^*(g, k) \forall i, j \in [n; n + g(n)] (|a_i - a_j| \leq 2^{-k}),$$

where

$$\Phi^*(g, k) := \tilde{g}^{(2^k)}(0) \text{ with } \tilde{g}(n) := n + g(n).$$

Moreover, there exists an $i < 2^k$ such that n can be taken as $\tilde{g}^{(i)}(0)$.³

³For $g : \mathbb{N} \rightarrow \mathbb{N}$ and $n \in \mathbb{N}$ we define $g^{(0)}(0) := 0$, $g^{(n+1)}(0) := g(g^{(n)}(0))$.

It is of interest here that the bound $\Phi^*(g, k)$ in proposition 2.3 does not depend on (a_n) at all (see also the discussion of this point further below).

Remark 2.4. *If (a_n) is a sequence of rational numbers so that \leq becomes (primitive recursively) decidable, then using the bound Φ^* and primitive recursive bounded search one can obtain a functional Φ (this time depending on (a_n)) that satisfies the no-counterexample interpretation of this principle. Such a solution is also possible for sequences (a_n) of real numbers a_n using rational approximations as the proper n.c.i.-treatment of \leq defined in terms of representatives of real numbers given as Cauchy sequences of rational numbers with some fixed rate of convergence would insist on doing. We bypassed this issue since we are only interested in the bound Φ^* .*

As an immediate consequence of proposition 2.3 we obtain the following (explicit version of the) ‘finite convergence principle’ which recently has been discussed by T. Tao ([98, 99]):

Corollary 2.5. *For all $k \in \mathbb{N}$, $g \in \mathbb{N}^{\mathbb{N}}$ there exists an $M \in \mathbb{N}$ such that for all nonincreasing finite sequences $0 \leq a_M \leq \dots \leq a_0 \leq 1$ of length $M + 1$ in $[0, 1]$ there exists an $n \in \mathbb{N}$ with*

$$n + g(n) \leq M \wedge \forall i, j \in [n; n + g(n)](|a_i - a_j| \leq 2^{-k}).$$

Moreover, we can compute M as $M := \Phi^*(g, k)$, where Φ^* is as in proposition 2.3.

Let us now consider general prenex normal sentences

$$A \equiv \exists x_1 \forall y_1 \dots \exists x_n \forall y_n A_{qf}(x_1, y_1, \dots, x_n, y_n)$$

and their Herbrand normal form (written again with function quantifiers)

$$A^H \equiv \forall f_1, \dots, f_n \exists x_1, \dots, x_n A_{qf}(x_1, f_1(x_1), \dots, x_n, f_n(x_1, \dots, x_n)).$$

Then we say that Φ_1, \dots, Φ_n satisfy the n.c.i. of A if (writing \underline{f} for f_1, \dots, f_n)

$$\forall \underline{f} A_{qf}(\Phi_1(\underline{f}), f_1(\Phi_1(\underline{f})), \dots, \Phi_n(\underline{f}), f_n(\Phi_1(\underline{f}), \dots, \Phi_n(\underline{f})))$$

holds.

The problem with the n.c.i. is that for sentences that no longer are Π_3^0 the Herbrand normal form in general is too much of a weakening of the original sentence such that a witness (or bound) as required in its no-counterexample interpretation would reflect the correct computational contribution the use of such a sentence in a proof of some Π_2^0 -theorem might have. In fact, even for Π_3^0 -sentences

$$A \equiv \forall x \exists y \forall z A_{qf}(x, y, z)$$

the n.c.i. of any prenex normal form $(A \rightarrow B)^{pr}$ of an implication $A \rightarrow B$, where

$$B \equiv \forall u \exists v B_{qf}(u, v)$$

in Π_2^0 is too weak to allow for a solution of the modus ponens by constructing a realizing function for B (out of functionals satisfying the n.c.i. of A and $(A \rightarrow B)^{pr}$) without an explosion in the computational complexity (see [59] for a detailed discussion of the ‘modus ponens’ problem for n.c.i.).

If

$$A \equiv \exists x \forall y \exists z A_{qf}(x, y, z)$$

is in Σ_3^0 (or of higher complexity) then already the n.c.i. of A might fail to give the correct result. This has to do with the fact that in order to infer A back from A^H for such A one needs AC (though – in the case of $A \in \mathcal{L}(\text{PA})$ – only from \mathbb{N} to \mathbb{N} and so rather than a proper choice axiom just a form of arithmetical comprehension) applied to Π_1^0 -formulas (or of higher complexity) whereas in the case of Π_3^0 -sentences it was needed only for quantifier-free (and hence decidable) formulas. This already matters even for bounded quantifiers ‘ $\exists x \leq a$ ’ as can be illustrated by the infinitary pigeonhole principle

$$(\text{IPP}): \forall n \in \mathbb{N} \forall f : \mathbb{N} \rightarrow C_n \exists i \leq n \forall k \in \mathbb{N} \exists m \geq k (f(m) = i),$$

where $C_n := \{0, 1, \dots, n\}$.

The Herbrand normal form of (IPP) is

$$(\text{IPP})^H \equiv \forall n \in \mathbb{N} \forall f : \mathbb{N} \rightarrow C_n \forall F : C_n \rightarrow \mathbb{N} \exists i \leq n \exists m \geq F(i) (f(m) = i)$$

which gives rise to the following computationally almost trivial solution for the n.c.i. of IPP:

$$M(n, f, F) := \max\{F(i) : i \leq n\} \text{ and } I(n, f, F) := f(M(n, f, F))$$

are realizers for ‘ $\exists i$ ’ and ‘ $\exists m$ ’ in $(\text{IPP})^H$, despite of the fact that the proof of IPP requires some substantial amount of induction (more precisely the so-called bounded collection principle for Π_1^0 -formulas whose strength is in between Σ_2^0 - and Σ_1^0 -induction).

We use this principle to motivate the Gödel functional (‘Dialectica’) interpretation D (and its monotone variant), where we refer here always to its combination ND with some negative translation N (e.g. one may use the so-called Shoenfield variant from [95], see [97]). This interpretation produces a $A^{ND} \equiv \forall X \exists Y A_*(X, Y)$ normal form for arbitrary formulas A , where A_* is quantifier-free, such that the equivalence between A and A^{ND} follows using only (classical logic and) **quantifier-free** choice

$$\text{QF-AC} : \forall x \exists y F_{qf}(x, y) \rightarrow \exists Y \forall x F_{qf}(x, Y(x)) \quad (F_{qf} \text{ quantifier-free}).$$

The price to be paid for this is that we need QF-AC for objects of arbitrary finite type (over some base type such as \mathbb{N}) and also X, Y in A^{ND} will be functionals of higher type (where the types depend on the logical complexity of A).

The ND-interpretation of (IPP) is arrived at in the following way (strictly speaking A^{ND} is defined as the $\exists\forall$ -form resulting from a final QF-AC application to our $\forall\exists$ -form which we omit here for better readability)

$$\begin{aligned}
 (\text{IPP}) & \stackrel{\text{QF-AC}}{\Leftrightarrow} \\
 & \forall n \in \mathbb{N} \forall f : \mathbb{N} \rightarrow \mathbb{N} \exists i \leq n \exists g : \mathbb{N} \rightarrow \mathbb{N} \forall k \in \mathbb{N} (g(k) \geq k \wedge f(g(k)) = i) \stackrel{\text{QF-AC}}{\Leftrightarrow} \\
 & \forall n \in \mathbb{N} \forall f : \mathbb{N} \rightarrow \mathbb{N} \forall K : \mathbb{N} \times \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N} \exists i \leq n \exists g : \mathbb{N} \rightarrow \mathbb{N} \\
 & \quad (g(K(i, g)) \geq K(i, g) \wedge f(g(K(i, g))) = i) \equiv: (\text{IPP})^{ND}.
 \end{aligned}$$

The functional interpretation of (IPP) requires functionals $I(n, f, K)$ and $G(n, f, K)$ realizing ‘ $\exists i$ ’ and ‘ $\exists g$ ’.

Note that the implication ‘ \Leftarrow ’ in the second equivalence above only needs computable (in f) and hence continuous functionals K .

These functionals reflect the correct computational contribution of the use of (IPP) in a proof as follows from the general soundness theorem of functional interpretation. I, K can be defined using primitive recursion of type level 1. However, to get a usable notation for these functionals it is advisable to use a finite form of bar recursion as is done in [90]. We refer to that paper as well as [66] for a detailed discussion and urge the reader to try to come up directly with a solution for I, G to appreciate the highly nontrivial task performed by functional interpretation even restricted to a principle as simple as (IPP).

In the applications to analysis discussed in sections 5, 6 and 7 below, one actually is interested in the extraction of bounds (rather than realizers) which, however, need to be **uniform**, i.e. independent from various parameters, to be useful. This can be achieved by modifying Gödel’s functional interpretation in such a way that instead of realizers for the functional interpretation so-called majorants (a sort of hereditarily monotone bounds) are extracted. This variant has been introduced in [56] under the name of **monotone functional interpretation**.⁴ Not only does this interpretation directly extract uniform bounds, it also nicely extends to important subsystems of analysis (based on the binary ‘weak’ König’s lemma WKL) and simplifies the extraction algorithm (see e.g. [41] for a detailed complexity analysis). In its simplest form the notion of majorizability (due to W.A. Howard [44]) is defined as follows for functionals of finite type over \mathbb{N} :

Definition 2.6. *Between functionals of type ρ the following binary relation is de-*

⁴A related so-called bounded functional interpretation was recently designed in Ferreira-Oliva [28] and has interesting applications to systems of feasible analysis (see [29]).

fined by induction on ρ :

$$\begin{cases} x^* \gtrsim_{\mathbf{N}} x \equiv x^* \geq x, \\ x^* \gtrsim_{\rho \rightarrow \tau} x \equiv \forall y^*, y (y^* \gtrsim_{\rho} y \rightarrow x^* y^* \gtrsim_{\tau} xy). \end{cases}$$

$x \leq_{\rho} y$ is defined as the pointwise inequality relation.

Remark 2.7. Sometimes, a variant of \gtrsim due to Bezem [8] with a clause ' $x^* y^* \gtrsim_{\tau} x^* y$ ' added is useful too.

Let us recall that a metric space (X, d) is called *complete* if every Cauchy sequence in X converges and it is called *separable* if there exists a countable subset $X_{count} \subseteq X$ such that each $x \in X$ can be arbitrarily good approximated by elements in X_{count} . A complete separable metric space is also called a *Polish metric space*. One way of defining the *compactness* of a complete metric space is by its *total boundedness* which means that for each $\varepsilon > 0$ there are finitely many points $x_1, \dots, x_n \in X$ such that any $x \in X$ is ε -close to one of these points. E.g. $[0, 1]^n$ is compact (w.r.t. the Euclidean metric) whereas \mathbb{R}^n is not. Obviously, any compact metric space is separable and so a Polish space. Via an appropriate so-called standard representation of Polish (i.e. complete separable) metric spaces P and compact (i.e. complete and totally bounded) metric spaces K via the Baire space $\mathbb{N}^{\mathbb{N}}$ respectively an appropriate compact subspace $\{f \in \mathbb{N}^{\mathbb{N}} : \forall n \in \mathbb{N} (f(n) \leq M(n))\}$ (for some effective M) it follows that the monotone functional interpretation extracts bounds Φ^* that are independent from parameters ranging over compact metric spaces but depend on representatives $f_x \in \mathbb{N}^{\mathbb{N}}$ for parameters $x \in P$ in Polish spaces P (see theorem 2.10 below).

The uniformity of the bound in proposition 2.3 above (and hence Tao's finite convergence principle) can be seen as an instance of monotone functional interpretation using the representation of the space of sequences in $[0, 1]$ as the compact metric space $[0, 1]^{\mathbb{N}}$ (with the product metric).

In the case of (IPP) one obtains majorants I^* and G^* for I, G which – using that the constant- n function majorizes $f : \mathbb{N} \rightarrow C_n$ – no longer depend on f but require a majorant K^* for the argument K in the case of G^* (as I^* we simply can take the constant- n functional). Not every functional K does possess a majorant K^* but among others e.g. all K 's that are continuous in g (w.r.t. to the product topology on $\mathbb{N}^{\mathbb{N}}$) do. For continuous K one can even replace g by some initial segment encoded in a number m (we use $[m]$ to denote the function which continues this initial segment by zeroes). Then as a consequence of the monotone interpretation and the uniform continuity of K (in g) on $\{g : g \leq_1 G^*(n, K^*)\}$ we get the following

semi-finite form

$$\begin{aligned} \forall n \in \mathbb{N} \forall K : C_n \times \mathbb{N}^{\mathbb{N}} &\xrightarrow{\text{cont.}} \mathbb{N} \\ \exists M \in \mathbb{N} \forall f : C_M &\rightarrow C_n \exists i \leq n \exists m \leq M \\ &(\text{Image}([m]) \subseteq C_M \wedge [m](K(i, [m])) \geq K(i, [m]) \wedge \\ &f([m](K(i, [m]))) = i) \end{aligned}$$

of (IPP) which is a kind of reformulation of Tao's ‘finitary’ infinite pigeonhole principle discussed in [98, 99].

Both for carrying out (monotone) functional interpretation as well as for formalizing proofs in analysis, formal systems based on (fragments of) arithmetic PA^ω in the language of functionals in all finite types over \mathbb{N} augmented by suitable analytical principles such as WKL are most convenient. For many such systems \mathcal{T}^ω , e.g. $\text{PA}^\omega + \text{QF-AC} + \text{WKL}$, one can prove results of the following type (using ‘1’ to denote the type $\mathbb{N} \rightarrow \mathbb{N}$):

Theorem 2.8 (Kohlenbach [53, 56, 57]). *Let $A_\exists(x^1, y^1, z^{\mathbb{N}}) \equiv \exists \underline{z} A_{qf}(x, y, z, \underline{z})$, where A_{qf} a quantifier-free formula and A_\exists contains only x, y, z free. Here \underline{z} is any tuple of variables up to type 2. Let s be a closed term.*

$$\left\{ \begin{array}{l} \text{From a } \mathcal{T}^\omega\text{-proof of } \forall x^1 \forall y \leq_1 s x \exists z^{\mathbb{N}} A_\exists(x, y, z) \\ \text{monotone functional interpretation extracts a closed term } \Phi \text{ of } \mathcal{T}^\omega \text{ such that} \\ \mathcal{T}^\omega \vdash \forall x^1 \forall y \leq_1 s x \exists z \leq \Phi(x) A_\exists(x, y, z). \end{array} \right.$$

Note that the bound $\Phi(x)$ does not depend on y .

Remark 2.9. *In fact, even the existential quantifiers $\exists \underline{z}$ can be bounded (in the sense of \leq_ρ where $\rho(\leq 2)$ is the type of the respective variable).*

If \mathcal{T}^ω is based on a weak form of extensionality given by a quantifier-free rule (see below), then y might have any type.

Instead of single variables x, y, z we may have tuples as well.

In the case of $\text{PA}^\omega + \text{QF-AC} + \text{WKL}$ the bound Φ will be a primitive recursive functional in the sense of Hilbert [43] and Gödel [36]. If PA^ω is restricted to Σ_1^0 -induction and primitive recursion in the sense of Kleene only, then Φ will be primitive recursive in the sense of Kleene. If one uses systems of bounded arithmetic such as $\text{G}_2\text{A}^\omega$ (from [57]) instead of PA^ω , then $\Phi(x)$ will be given by a term built up out of 0, S , $+$, \cdot and $x^M(k) := \max\{x(i) : i \leq k\}$ only, i.e. a polynomial in x^M . So in the presence of WKL (corresponding to the Heine-Borel compactness of spaces such as $[0, 1]^n$) as the only genuine analytical principle the complexity of Φ is determined solely by the strength of the underlying arithmetical system. In the presence of (instances of) sequential compactness (corresponding to instances of arithmetical comprehension) this is no longer the case but the contribution of

this form of compactness still is rather limited in many cases (see [58]). If higher comprehension over numbers is used or even dependent choice in all types – we denote the resulting system by \mathcal{A}^ω – then Φ will be a so-called bar recursive functional in the sense of C. Spector ([96], see [66] for a modern treatment).

In order to apply theorem 2.8 to actual proofs in analysis one first has to verify that the theorem to be proved (and to a certain extent its proof)⁵ are can be formalized in a suitable formal system which requires a representation of the various analytic objects involved. After that formalization one has to check whether the statement has or can be transformed into (using, if necessary, an appropriate enrichment of data) the required logical form. The process is much simplified using the following kind of ‘macro’: using the standard (or ‘Cauchy’-) representation of Polish spaces P and compact metric spaces K mentioned already above one can reduce quantification over P (resp. over K) to quantification $\forall x^1$ resp. $(\forall y \leq_1 M)$ without introducing new quantifiers. So it suffices to formalize the proof up to quantification over such spaces (provided they are representable in the formal system at hand). This yields the following applied form of theorem 2.8:

Theorem 2.10 (Kohlenbach [54, 56]).

$$\left\{ \begin{array}{l} \text{From a } \mathcal{T}^\omega\text{-proof of } \forall x \in P \forall y \in K \exists z^{\mathbb{N}} A_{\exists}(x, y, z) \\ \text{monotone functional interpretation extracts a closed term } \Phi \text{ of } \mathcal{T}^\omega \text{ such that} \\ \mathcal{T}^\omega \vdash \forall x \in P \forall y \in K \exists z \leq \Phi(f_x) A_{\exists}(x, y, z), \end{array} \right.$$

where Φ depends on a given representative $f_x \in \mathbb{N}^{\mathbb{N}}$ of $x \in P$ but is independent of $y \in K$. Here A_{\exists} is a purely existential formula as above (when expressed in terms of the representation of P, K) that is (provably in \mathcal{T}^ω) extensional in x, y with respect to $=_x, =_y$.⁶

For the special Polish spaces \mathbb{N} and $\mathbb{N}^{\mathbb{N}}$ no representation is necessary since \mathcal{T}^ω contains quantification over these spaces as a primitive concept. Let us illustrate the use of theorem 2.10 by two extremely simple examples:

$$(3) \text{ PA}^\omega \vdash \forall x \in \mathbb{R} \exists n \in \mathbb{N} (n > x)$$

and

$$(4) \text{ PA}^\omega \vdash \forall x \in (0, 1] \exists n \in \mathbb{N} (1 < n \cdot x).$$

Clearly, \mathbb{R} has a standard representation as Polish space via the Cauchy completion of \mathbb{Q} , i.e. real numbers are represented as Cauchy sequences of rational

⁵Proofs of universal lemmas and even of lemmas of the form $\forall u \exists v \leq tu \forall w F_{gf}$ for functionals u, v, w of moderate types can be disregarded as such lemmas can be treated simply as axioms, see the remarks in the introduction and [53, 56].

⁶This extensionality is not needed for the extraction of Φ but only to justify that A_{\exists} indeed speaks about $x \in P$ and $y \in K$ rather than $x, y \in \mathbb{N}^{\mathbb{N}}$.

numbers with fixed rate of convergence (say 2^{-n}) and $>$ (expressed on such representatives) is in Σ_1^0 . Hence theorem 2.10 applies and there exists a primitive recursive functional which computes given a representative (r_n) of x a number $n \in \mathbb{N}$ satisfying (3). E.g. we may take $\Phi(r_n) := \lceil r_0 \rceil + 1$.

In (4) there does not exist any subrecursive such Ψ which – given a representative (r_n) of a strictly positive real number x – would produce an n satisfying (4). This corresponds to the fact that one cannot treat $\forall x \in (0, 1]$ as a primitive notion (since $(0, 1]$ is not complete) but only

$$\forall x \in \mathbb{R} (\exists m \in \mathbb{N} (x \geq 1/(m+1)) \rightarrow \exists n \in \mathbb{N} (1 < n \cdot x)),$$

i.e.

$$\forall x \in \mathbb{R} \forall m \in \mathbb{N} \exists n \in \mathbb{N} (x \geq 1/(m+1) \rightarrow 1 < n \cdot x),$$

where now ‘ $x \geq 1/(m+1) \rightarrow 1 < n \cdot x$ ’ is (equivalent to) a Σ_1^0 -formula and ‘ $\forall x \in \mathbb{R} \forall n \in \mathbb{N}$ ’ **can** be treated as primitive quantifiers. Indeed, in (r_n) **plus** m as input the solution is trivial: $\Psi((r_n), m) := m + 2$. Alternatively, one can rewrite (4) using the Polish space $[1, \infty)$ as

$$\text{PA}^\omega \vdash \forall x \in [1, \infty) \exists n \in \mathbb{N} (1 < n \cdot (1/x)).$$

We then can take $\Psi(r_n) := \Phi(r_n)$ with Φ from the first example (note that $1/x$ is primitive recursively definable on $[1, \infty)$ but not on $(0, \infty)$ in the Cauchy representation (r_n)). This example a-fortiori shows that if the compactness of K is weakened to total boundedness than one does not have in general uniform bounds anymore (independent from $y \in K$). This also is the case if K is only Polish and bounded (but not totally bounded): Consider the closed unit ball B in the space $C[0, 1]$ of all continuous functions $f : [0, 1] \rightarrow \mathbb{R}$ w.r.t. the uniform norm $\|f\|_\infty := \sup\{|f(x)| : x \in [0, 1]\}$ (note that w.r.t. the metric induced by $\|\cdot\|_\infty$, $C[0, 1]$ is a Polish space and B a bounded Polish space). Then

$$\text{PA}^\omega \vdash \forall f \in B \exists n \in \mathbb{N} (n \text{ is code of some } p \in \mathbb{Q}[x] \text{ with } \|f - p\|_\infty < 1/2).$$

Clearly, there is no uniform bound on ‘ $\exists n$ ’ (i.e. a bound independent of $f \in B$) as the sequence $f_n(x) := \sin(nx) \in B$ shows. However, given a representative of f (in the sense of the standard representation of the Polish space $C[0, 1]$) one can easily compute a suitable n .

At first sight, these results essentially seem to show that the requirements on K being complete and totally bounded are necessary to be guaranteed the extractability of bounds Φ that do not depend on parameters y from K . However, these counterexamples only show that this can be the case for concrete spaces K . Looking at the example concerning $B \subset C[0, 1]$ above one realizes that it is the (provable) separability of $C[0, 1]$ that was crucially used (likewise it was the provable

incompleteness of $(0, 1]$ that was used in the first counterexample). In fact, to have a B -uniform bound on the very property of separability is nothing else but requiring B to be totally bounded. This opens up the possibility to extract uniform bounds from proofs that do not use the separability of K but rather treat K as an abstract metric space (X, d) . Since the only direct way to talk about metric spaces in systems like \mathcal{T}^ω is via their standard representation which is based on separability, one then has to extend \mathcal{T}^ω by an abstract space (X, d) as a kind of atom resulting in a new system $\mathcal{T}^\omega[X, d]$. This is done by introducing a new ground type X (for objects in X) and all finite types over \mathbb{N} and X together with a constant d_X plus axioms expressing that d_X is a pseudo-metric. Equality $x =_X y$ is defined as $d_X(x, y) =_{\mathbb{R}} 0_{\mathbb{R}}$, where the real numbers still are represented as type-1 objects and $=_{\mathbb{R}}$ is the equivalence relation on those corresponding to equality in \mathbb{R} . Higher type equality is defined extensionally. Whereas in the previous cases the issue of extensionality was not important due to the availability of an elimination-of-extensionality procedure (note that functional interpretation is not sound for full extensionality in higher types) it now becomes crucial that we only include a weak quantifier-free **rule** of extensionality which allows one to infer $r[s] =_{\tau} r[t]$ only once $s =_{\rho} t$ has been established. Fortunately, for most applications in fixed point theory and ergodic theory this does not cause any problems since the extensionality of functions $f : X \rightarrow X$ usually follows from the f -properties assumed such as f being nonexpansive. In [63] such systems were introduced and very general metatheorems on the extractability of bounds that are independent from parameters $x \in X$ and even $f : X \rightarrow X$ were obtained under the only assumption that (X, d) was bounded. In [31] this is further refined and it is shown that the global boundedness assumption on (X, d) can be replaced by local bounds. This was achieved by a new majorizability relation $x^* \geq_{\rho}^a x$ that is parametrized by some reference point $a \in X$. For a finite type ρ over \mathbb{N}, X , the majorant x^* always has a finite type over just \mathbb{N} resulting from replacing in ρ the type X by \mathbb{N} . For the ground type X , the relation is defined as follows

$$x^* \geq_X^a x \equiv x^* \geq_{\mathbb{R}} d_X(a, x),$$

where x has type X while x^* has the type \mathbb{N} . For $\rho := X \rightarrow X$ the relation is defined between objects f^* of type 1 = $(\mathbb{N} \rightarrow \mathbb{N})$ and f of type $X \rightarrow X$ as follows

$$f^* \geq_{X \rightarrow X}^a f \equiv \begin{cases} \forall n \in \mathbb{N} (f^*(n+1) \geq f^*(n)) \wedge \\ \forall x^* \in \mathbb{N} \forall x \in X (x^* \geq_{\mathbb{R}} d_X(a, x) \rightarrow f^*(x^*) \geq_{\mathbb{R}} d_X(a, f(x))) \end{cases}$$

Our approach extracts effective bounds in terms of such majorants $x^* \in \mathbb{N}$, $f^* : \mathbb{N} \rightarrow \mathbb{N}$ etc. rather than $x \in X$, $f : X \rightarrow X$ and does not presuppose at all that (X, d) comes together with some notion of effectivity on X .

In [63, 31] not only metric spaces (X, d) but also other classes of structures such as

hyperbolic spaces, CAT(0)-spaces, normed spaces, uniformly convex spaces and inner product spaces are treated. Further examples (\mathbb{R} -trees, δ -hyperbolic spaces and uniformly convex hyperbolic spaces) are discussed in [82]. Since hyperbolic spaces play an important role in section 6 below we give the definition

Definition 2.11 ([63, 34, 92]). *(X, d, W) is called a hyperbolic space if (X, d) is a metric space and $W : X \times X \times [0, 1] \rightarrow X$ a function satisfying*

- (i) $\forall x, y, z \in X \forall \lambda \in [0, 1] (d(z, W(x, y, \lambda)) \leq (1 - \lambda)d(z, x) + \lambda d(z, y)),$
- (ii) $\forall x, y \in X \forall \lambda_1, \lambda_2 \in [0, 1] (d(W(x, y, \lambda_1), W(x, y, \lambda_2)) = |\lambda_1 - \lambda_2| \cdot d(x, y)),$
- (iii) $\forall x, y \in X \forall \lambda \in [0, 1] (W(x, y, \lambda) = W(y, x, 1 - \lambda)),$
- (iv) $\left\{ \begin{array}{l} \forall x, y, z, w \in X, \lambda \in [0, 1] \\ (d(W(x, z, \lambda), W(y, w, \lambda)) \leq (1 - \lambda)d(x, y) + \lambda d(z, w)). \end{array} \right.$

Definition 2.12. *Let (X, d, W) be a hyperbolic space. The set*

$$seg(x, y) := \{ W(x, y, \lambda) : \lambda \in [0, 1] \}$$

is called the metric segment with endpoints x, y .

We usually write $(1 - \lambda)x \oplus \lambda y$ for $W(x, y, \lambda)$.

Remark 2.13. *Every convex subset C of a normed linear space is a hyperbolic space with $W(x, y, \lambda) := (1 - \lambda)x + \lambda y$.*

In the case of theorems 2.8 and 2.10 it is mainly the concrete numerical information obtained from specific extracted bounds Φ of (in most cases) relatively low complexity which is of interest (see section 5 below) as the existence of a computable uniform bound follows (in the case where all \exists -quantifiers in A_{gf} have type \mathbb{N}) by unbounded search and the fact that computable functionals $\mathbb{N}^{\mathbb{N}} \times 2^{\mathbb{N}} \rightarrow \mathbb{N}$ have computable moduli of uniform continuity $\omega_{\Phi} : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ for $\Phi(x, \cdot)$ restricted to $y \in 2^{\mathbb{N}}$ so that also $\Phi_M(x) := \max\{\Phi(x, y) : y \in 2^{\mathbb{N}}\}$ (and even $\Phi_M(x) := \max\{\Phi(x, y) : y \leq_1 sx\}$) is computable as well. In the case of noncompact spaces, however, even the existence of a uniform bound at all is of interest. We, therefore, use in the following the strongest system \mathcal{A}^{ω} extended by an abstract hyperbolic space resulting in $\mathcal{A}^{\omega}[X, d, W]_{-b}$, where ‘ $-b$ ’ indicates that we do not assume (X, d) to be bounded.

Rather than formulating here one of the general metatheorems from [31] we confine ourselves to some special case which, however, is typical for the kind of results used in obtaining the bounds in sections 6 and 7 below.

Definition 2.14. Let (X, d) be a metric space. A mapping $f : X \rightarrow X$ is called *nonexpansive* (short: *n.e.*) if

$$\forall x, y \in X (d(f(x), f(y)) \leq d(x, y)).$$

Theorem 2.15 (Gerhardy - Kohlenbach [31]). Let A_\exists be an \exists -formula and P, K Polish resp. compact metric spaces in standard representation by \mathcal{A}^ω -definable terms.

If $\mathcal{A}^\omega[X, d, W]_{-b}$ proves a sentence

$$\forall x \in P \forall y \in K \forall z^X, \tilde{z}^X, f^{X \rightarrow X} (f \text{ nonexpansive} \rightarrow \exists v^{\mathbb{N}} A_\exists)$$

then one can extract a (bar recursively) computable functional $\Phi(g_x, b)$ s.t. for all $x \in P, g_x \in \mathbb{N}^{\mathbb{N}}$ representative of $x, b \in \mathbb{N}$

$$\forall y \in K \forall z, \tilde{z} \in X \forall f : X \rightarrow X (f \text{ n.e.} \wedge d(z, f(z)), d(z, \tilde{z}) \leq b \rightarrow \exists v \leq \Phi(g_x, b) A_\exists)$$

holds in **any** nonempty hyperbolic space (X, d, W) .

The most crucial thing about theorem 2.15 is that the bound Φ depends on (X, d, W) , f, z and \tilde{z} **only** via an upper bound $b \geq d(z, \tilde{z}), d(z, f(z))$. In particular, if (X, d) is b -bounded one obtains a bound that is fully independent from f, z, \tilde{z} (this was already proved in Kohlenbach [63]).

Theorem 2.15 also holds for normed spaces (as well as subclasses of those such as Hilbert spaces) if we additionally require that $b \geq \|z\|$ (due to the fact that we now have to take the reference point a in our majorization relation as the zero vector O_X in this case). This is unavoidable as e.g. the following trivial example

$$\forall z^X, \tilde{z}^X \exists y^{\mathbb{N}} (y > \|z\| + \|\tilde{z}\|)$$

shows.

Remark 2.16. For systems based on $\text{PA}^\omega + \text{QF-AC} + \text{WKL}$ etc. instead of \mathcal{A}^ω we obtain bounds of the respective limited complexity classes discussed above.

3 Applications of proof mining in number theory

A famous theorem of K.F. Roth says

Theorem 3.1 (Roth [93]). An algebraic irrational number α has only finitely many exceptionally good rational approximations, i.e. for $\varepsilon > 0$ there are only finitely many $q \in \mathbb{N}$ such that

$$R(q) := q > 1 \wedge \exists! p \in \mathbb{Z} : (p, q) = 1 \wedge |\alpha - pq^{-1}| < q^{-2-\varepsilon}.$$

Weaker results (with smaller exponents) had been obtained before by Dirichlet 1842, Liouville 1844, Thue 1909, Siegel 1921, Schneider 1936 and Dyson and Gelfond 1947 (see [84, 85]).

In 1983, Esnault and Viehweg found a new proof of Roth's theorem. Both proofs are ineffective and *prima facie* give no bounds neither on the size nor on the number of q 's. Nevertheless, Davenport and Roth obtained an exponential bound on the number of q 's analyzing Roth's proof.

In 1985 Luckhardt (see [84]) applied a systematic logical analysis (based on Herbrand's theorem in the form (L) discussed in section 2 and using prior work of Kreisel [77]) to both proofs of Roth's theorems, obtaining from Roth's proof a bound which roughly is the fourth root of the bound found by Davenport and Roth and from the proof due to Esnault and Viehweg the first polynomial bound on the number of q 's, more precisely:

Theorem 3.2 (Luckhardt [84]). The following upper bound on $\#\{q : R(q)\}$ holds:

$$\#\{q : R(q)\} < \frac{7}{3} \varepsilon^{-1} \log N_\alpha + 6 \cdot 10^3 \varepsilon^{-5} \log^2 d \cdot \log(50 \varepsilon^{-2} \log d),$$

where $N_\alpha < \max(21 \log 2h(\alpha), 2 \log(1 + |\alpha|))$ and h is the logarithmic absolute homogeneous height.

Independently, Bombieri and van der Poorten obtained in 1988 [10] a roughly similar bound using a more ad hoc strategy of proof.

4 Applications of proof mining in algebra

Artin's solution of Hilbert's 17th problem can be formulated as follows (see e.g. [85]):

Let k be an ordered field and let R be a real-closed order extension of k . If $f \in k[x_1, \dots, x_n]$ of degree d is positive semi-definite over R , then f can be represented as a non-negative weighted sum of squares of the form

$$f(x_1, \dots, x_n) = \sum_{i=1}^{\lambda} p_i \cdot g_i(x_1, \dots, x_n)^2,$$

where $p_i \in k$, $p_i \geq 0$ and $g_i \in k(x_1, \dots, x_n)$.

Artin's proof can be formalized in weak formal systems based on the weak König's lemma WKL (which are conservative over Primitive Recursive Arithmetic PRA as can be shown e.g. by monotone functional interpretation, see [53, 57]). This was already observed by Kreisel in the late 50's who concluded from this the existence of primitive recursive bounds (in n, d) for λ and the degrees

of the rational functions involved. If one carries this out one obtains an exponential tower whose height is given by the number of variables. Later L. Henkin showed that the p_i and the coefficients of the g_i can be piecewise-rationally constructed from the coefficients of f . Kreisel asked whether such a case distinction is necessary and whether there would exist a continuous solution. These questions were settled completely by Delzell in a series of papers ([22, 23, 24, 25], see [26] for a thorough discussion of the role of proof theory played in these developments). The details are too technical to be stated here.

Another type of applications of proof theory is presented by recent work of T. Coquand and H. Lombardi (see e.g. [19, 20, 21]). Here the use of ideal (so-called strict Π_1^1 -) statements (on objects such as prime ideals or maximal ideals) in abstract algebra is replaced by elementary (Σ_1^0) syntactical ones which can effectively be witnessed.⁷ This effective reduction of strict Π_1^1 -formulas to Σ_1^0 -formulas again can be viewed as a WKL-elimination mentioned in the introduction (WKL in this work shows up indirectly via the completeness theorem for propositional logic).

Among other things this has led to a new non-Noetherian version of Serre's splitting-off theorem (1958) and the Forster-Swan theorem (1964-67) improving results of Heitmann from 1984.

5 Application of proof mining in approximation theory

An important classical theorem in best approximation theory is the following:

Theorem 5.1 (Jackson [46]). Let $f \in C[0, 1]$ and $n \in \mathbb{N}$. There exists a unique polynomial $p_b \in P_n$ of degree $\leq n$ that approximates f best in the L_1 -norm, i.e.

$$\|f - p_b\|_1 = \inf_{p \in P_n} \|f - p\|_1 =: \text{dist}_1(f, P_n).$$

Here $\|f\|_1 := \int_0^1 |f(x)| dx$.

Both the (easy) existence as well as the (difficult) uniqueness part are proved ineffectively involving noncomputable real numbers in the form of – logically speaking – WKL. Applying the extraction algorithm provided by the proof of theorem 2.10, the following result was extracted from another ineffective uniqueness proof due to E.W. Cheney [18]:

⁷A is strict Π_1^1 if it has the form $\forall X \exists y A_{qf}$, where X and y are set resp. number variables and A_{qf} is quantifier-free. Alternatively, one can use quantification over 0/1-functions instead of X .

Theorem 5.2 (Kohlenbach-Oliva [73]). Let

$$\Phi(\omega, n, \varepsilon) := \min\left\{\frac{c_n \varepsilon}{8(n+1)^2}, \frac{c_n \varepsilon}{2} \omega_n\left(\frac{c_n \varepsilon}{2}\right)\right\},$$

where

$$c_n := \frac{|n/2|! \lceil n/2 \rceil!}{2^{4n+3} (n+1)^{3n+1}} \quad \text{and} \quad \omega_n(\varepsilon) := \min\left\{\omega\left(\frac{\varepsilon}{4}\right), \frac{\varepsilon}{40(n+1)^4 \lceil \frac{1}{\omega(1)} \rceil}\right\}.$$

Then $\Phi(\omega, n, \varepsilon)$ is an effective rate of strong unicity for the best L_1 -approximation of any function f in $C[0, 1]$ having modulus of uniform continuity ω from P_n , i.e. for all n and $f \in C[0, 1]$

$$\forall p_1, p_2 \in P_n; \varepsilon \in \mathbb{Q}_+^* \left(\bigwedge_{i=1}^2 (\|f - p_i\|_1 \leq \text{dist}_1(f, P_n) + \Phi(\omega, n, \varepsilon)) \rightarrow \|p_1 - p_2\|_1 \leq \varepsilon \right),$$

where ω is a modulus of uniform continuity of the function f , i.e.

$$\forall \varepsilon \in \mathbb{Q}_+^* \forall x, y \in [0, 1] (|x - y| < \omega(\varepsilon) \rightarrow |f(x) - f(y)| < \varepsilon).$$

Note that Φ only depends on f only via the modulus ω .

What makes Φ a rate of ‘strong unicity’ in the sense of numerical analysis is that it does not depend at all on $p_1, p_2 \in P_n$. This can be explained in terms of theorem 2.10 as it is easy to see that it suffices to construct such a rate on the compact subset $K_{f,n} := \{p \in P_n : \|p\|_1 \leq \frac{5}{2} \|f\|_1\}$ since such a rate can be extended to whole P_n (see [54, 73, 66]). Strong unicity plays a vast role in approximation theory. In particular, any rate of strong unicity provides a modulus of pointwise continuity (‘stability’) of the corresponding projection operator that maps a function to its unique best approximation. The fact that Φ does not depend on f as such but only on ω is also guaranteed a priori based on theorem 2.10 (see [73]).

Remark 5.3. *The fact that the bound depends on ω comes from the representation of $C[0, 1]$ as Polish space w.r.t. $\|\cdot\|_\infty$. ($C[0, 1], \|\cdot\|_1$) is easily seen to be incomplete and hence not a Polish space!*

Although the uniqueness of the best L_1 -approximation was known since 1921, only in 1975 Björnestrål [9] proved the existence of a rate of strong unicity Φ having the form $c_{f,n} \varepsilon \omega_n(c_{f,n} \varepsilon)$, for some constant $c_{f,n}$ depending on f and n . Björnestrål’s proof is ineffective and does not describe $c_{f,n}$ explicitly. In 1978, Kroó [80] improved Björnestrål’s results by showing that a constant $c_{\omega,n}$, depending only on the modulus of uniform continuity of f and n exists. Again, the constant is not presented. Kroó also showed that the ε -dependency established by

Björnestal is optimal. Since the above rate also has this optimal dependency, theorem 5.2 is an explicit effective version of Kroó's result (see the detailed discussion in [66]). This effective rate of strong unicity allows one for the first time to give a subrecursive algorithm for the computation of the best approximation for general $f \in C[0, 1]$. The complexity of that procedure is analyzed in [89].

Effective bounds on strong unicity for best Chebycheff approximation were extracted in [54, 55] improving earlier results of D. Bridges and K.-I. Ko.

6 Application of proof mining in metric fixed point theory

A substantial part of metric fixed point theory studies the fixed point (and approximate fixed point) property of nonexpansive selfmappings $f : C \rightarrow C$ of convex subsets of normed spaces $(X, \|\cdot\|)$ or – more generally – hyperbolic spaces (which includes the class of CAT(0)-spaces), see e.g. [50] and – for fixed point theory in the context of CAT(0)-spaces – [48, 49]. Whereas the fixed point theory of contractions, i.e. of mappings $f : X \rightarrow X$ satisfying

$$\forall x, y \in X (d(f(x), f(y)) \leq c \cdot d(x, y))$$

for some $c \in (0, 1)$, essentially is trivial and fully effective due to the Banach fixed point theory, this is radically different for the more general class of nonexpansive mappings. Here in general neither do fixed points exist nor are they necessarily unique in cases where they do exist. Moreover, even in the case of a unique fixed point, the trivial iteration $f^n(x)$ might not converge to the fixed point as the example $f : [0, 1] \rightarrow [0, 1]$, $f(x) := 1 - x$ shows: $f^n(x)$ converges to the unique fixed point $1/2$ iff $x = 1/2$. As a result of these difficulties, the fixed point theory of nonexpansive mappings is one of the most active research areas in nonlinear analysis. However, the fixed point theory of such mappings still has a certain computational flavor as one can define other effective iteration schemata which under general conditions converge towards a fixed point or – at least – provide approximate fixed point sequences.

The most common schema is the so-called Krasnoselski-Mann iteration which for a given sequence (λ_n) in $[0, 1]$ and starting point $x_0 \in C$ is defined as follows (for the general case of hyperbolic spaces)

$$x_{n+1} := (1 - \lambda_n)x_n \oplus \lambda_n f(x_n).$$

In the following theorems we assume (following [45]) that $(\lambda_k)_{k \in \mathbb{N}}$ satisfies the following conditions:

- (λ_k) is divergent in sum,
- $\exists K \in \mathbb{N} \forall k \in \mathbb{N} (\lambda_k \in [0, 1 - \frac{1}{K}])$.

Theorem 6.1 (Ishikawa [45]).

Under the assumptions above the following holds:

$$(x_n)_{n \in \mathbb{N}} \text{ bounded} \rightarrow \|x_n - f(x_n)\| \xrightarrow{n \rightarrow \infty} 0.$$

The theorem has been extended to hyperbolic spaces in [34]. Both proofs are *prima facie* ineffective and do not provide any rate of convergence.

Another important result is the Borwein-Reich-Shafrir theorem

Theorem 6.2 (Borwein-Reich-Shafrir [11]). *Let (X, d, W) be a hyperbolic space, $f : X \rightarrow X$ nonexpansive and (λ_n) as above. Then*

$$d(x_n, f(x_n)) \xrightarrow{n \rightarrow \infty} r(f) := \inf\{d(y, f(y)) : y \in X\}.$$

Note that in the Borwein-Reich-Shafrir theorem, (x_n) is not assumed to be bounded. The Ishikawa-Gobel-Kirk theorem combined with the Borwein-Reich-Shafrir theorem implies that

$$d(x_n, f(x_n)) \xrightarrow{n \rightarrow \infty} 0$$

if there exists an x^* such that the Krasnoselski-Mann iteration starting from x^* is bounded. In [70] (and – for the normed case – in [61]) a logical analysis of the ineffective proof of the Borwein-Reich-Shafrir theorem based on the extraction algorithm from the proof of theorem 2.15 has been carried out resulting in a quantitative version of the latter guaranteed a priori by theorem 2.15. Combined with the (mere truth of the) Ishikawa-Goebel-Kirk theorem the following bound on this convergence is extracted (since $(d(x_n, f(x_n)))_n$ is nonincreasing the convergence towards 0 can be written as a Π_2^0 -statement so that theorem 2.15 applies):

Theorem 6.3 (Kohlenbach-Leuştean [70]). Let (X, d, W) be a hyperbolic space and $f : X \rightarrow X$ be a nonexpansive mapping, $(\lambda_n)_{n \in \mathbb{N}}$, α and K be such that $\alpha : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ be such that $\lambda_n \in [0, 1 - \frac{1}{K}]$ and

$$\forall i, n \in \mathbb{N} ((\alpha(i, n) \leq \alpha(i + 1, n)) \wedge (n \leq \sum_{s=i}^{i+\alpha(i,n)-1} \lambda_s)).$$

Let $b > 0$, $x, x^* \in X$ be such that

$$d(x, x^*) \leq b \wedge \forall n, m \in \mathbb{N} (d(x_n^*, x_m^*) \leq b).$$

Then the following holds

$$\forall \varepsilon > 0 \forall n \geq h(\varepsilon, b, K, \alpha)(d(x_n, f(x_n)) \leq \varepsilon),$$

where

$$\begin{aligned} h(\varepsilon, b, K, \alpha) &:= \widehat{\alpha}(\lceil 10b \cdot \exp(K(M+1)) \rceil - 1, M), \text{ with} \\ M &\in \mathbb{N} \text{ such that } M \geq \frac{1+4b}{\varepsilon} \text{ and} \\ \widehat{\alpha}(0, n) &:= \tilde{\alpha}(0, n), \quad \widehat{\alpha}(i+1, n) := \tilde{\alpha}(\widehat{\alpha}(i, n), n) \text{ with} \\ \tilde{\alpha}(i, n) &:= i + \alpha(i, n) \quad (i, n \in \mathbb{N}). \end{aligned}$$

Note that the rate of convergence depends on x, f, X only via b and on (λ_n) only via α, K . In [70], many more results of this type are proved.

Since our notion of hyperbolic space, in particular, contains all CAT(0)-spaces, the result applies to these spaces as well.

For the special case of (convex subsets of) normed spaces the result was proved already in [61] and [62]. For the general logical background see [63, 74]. In [63, 31] it is shown that the quantitative version of such a kind is guaranteed by a general logical metatheorem whose proof provides an algorithm for the extraction of the bound in this and many other contexts.

The result implies, in particular, that for bounded X the convergence $d(x_n, f(x_n)) \rightarrow 0$ is uniform in x and f (ineffectively this is due to [34]).

Since (x_n^*) is assumed to be bounded, a natural question to be addressed by proof mining is to analyze how much of this boundedness actually is needed. As a consequence of the fact that the above result is based on the logical analysis of the Borwein-Reich-Shafrir theorem (which does not involved any boundedness assumption) and uses only the truth of the Ishikawa-Goebel-Kirk theorem (rather than its proof), this question is not answered by these results but requires a direct logical analysis of the Ishikawa-Goebel-Kirk theorem which (for $x^* := x$) gives the following answer:

Theorem 6.4 (Kohlenbach [66]). *Let (X, d, W) be a nonempty hyperbolic space, $f : X \rightarrow X$ a nonexpansive mapping, $(\lambda_n), K, \alpha$ as in theorem 6.3, $x \in X$ and (x_n) the Krasnoselski-Mann iteration of f starting from x and $\tilde{b} > 0$ so that $d(x, f(x)) \leq \tilde{b}$. Then for every $\varepsilon, b > 0$ the following holds (abbreviating $h^*(\varepsilon, b, \tilde{b}, K, \alpha)$ by h^*):*

$$\forall i \leq h^* \forall j \leq \alpha(h^*, M) (d(x_i, x_{i+j}) \leq b) \rightarrow \forall n \geq h^* (d(x_n, f(x_n)) < \varepsilon),$$

where

$$h^*(\varepsilon, b, \tilde{b}, K, \alpha) := \widehat{\alpha} \left(\left\lceil \frac{\tilde{b} \cdot \exp \left(K \cdot \left(\frac{3\tilde{b}+b}{\varepsilon} + 1 \right) \right)}{\varepsilon} \right\rceil - 1, M \right)$$

with $\widehat{\alpha}$ as before.

For the case of sequences (λ_n) in $[a, b]$ with $0 < a < b < 1$ we obtain from theorem 6.4 the following qualitative improvement of the Ishikawa-Goebel-Kirk theorem concerning the requirement of (x_n) being bounded (which for the case of constant $\lambda_n := \lambda \in (0, 1)$ and convex subsets of normed spaces was first observed in [2] (theorem 2.1)):

Corollary 6.5 (Kohlenbach [66]). *Let (X, d, W) be a hyperbolic space and $f : X \rightarrow X$ nonexpansive. For $x \in X$ and (λ_n) in $[a, b]$, where $0 < a < b < 1$, let (x_n) be the corresponding Krasnoselski-Mann iteration of f starting from x . Let*

$$c(n) := \max\{d(x, x_j) : j \leq n\}.$$

Then

$$\lim_{n \rightarrow \infty} \frac{c(n)}{n} \rightarrow 0$$

implies that

$$\lim_{n \rightarrow \infty} d(x_n, f(x_n)) = 0.$$

The proof of this corollary is based on the fact that for $K \in \mathbb{N}$, $K \geq 2$ satisfying $\lambda_n \in [\frac{1}{K}, 1 - \frac{1}{K}]$ for all $n \in \mathbb{N}$ one can take $\alpha(i, M) := K \cdot M$.

Remark 6.6. *The previous result shows that $d(x_n, f(x_n)) \rightarrow 0$ provided (x_n) grows with a lower than linear (in n) rate. This is optimal in the sense that linear growth does not suffice as follows from the following simple example: $X := \mathbb{R}$, $f(x) := x + 1$ and $\lambda := \frac{1}{2}$. For the starting point $x_0 := 0$ we have for the Krasnoselski-Mann iteration (x_n) that $x_n = \frac{n}{2}$, but $d(x_n, f(x_n)) = 1$ for all $n \in \mathbb{N}$.*

Since the fundamental paper Goebel-Kirk [33], an extension of the class of nonexpansive functions has been studied extensively in fixed point theory:

Definition 6.7. *Let (X, d) be a metric space. A function $f : X \rightarrow X$ is called asymptotically nonexpansive if for some sequence (k_n) in $[0, \infty)$ with $\lim_{n \rightarrow \infty} k_n = 0$ one has (with f^n denoting the n -th iteration of f)*

$$d(f^n x, f^n y) \leq (1 + k_n)d(x, y), \quad \forall n \in \mathbb{N}, \forall x, y \in X.$$

In the case of asymptotically nonexpansive mappings one considers the following version of the Krasnoselski-Mann iteration:

$$x_0 := x, \quad x_{n+1} := (1 - \lambda_n)x_n + \lambda_n f^n(x_n).$$

Definition 6.8 ([35, 82]). *A hyperbolic space (X, d, W) is uniformly convex if for any $r > 0$ and any $\varepsilon \in (0, 2]$ there exists $\delta \in (0, 1]$ such that for all $a, x, y \in X$,*

$$\left. \begin{array}{l} d(x, a) \leq r \\ d(y, a) \leq r \\ d(x, y) \geq \varepsilon r \end{array} \right\} \Rightarrow d\left(\frac{1}{2}x \oplus \frac{1}{2}y, a\right) \leq (1 - \delta)r. \quad (1)$$

A mapping $\eta : (0, \infty) \times (0, 2] \rightarrow (0, 1]$ providing such a $\delta := \eta(r, \varepsilon)$ for given $r > 0$ and $\varepsilon \in (0, 2]$ is called a modulus of uniform convexity.

We say that η is **monotone** if it decreases with r (for any fixed ε).

Examples of uniformly convex hyperbolic spaces (with monotone modulus of uniform convexity) are Hilbert spaces as well as CAT(0)-spaces which are the generalization of Hilbert spaces satisfying instead of the parallelogram equality only a parallelogram inequality (also called Bruhat-Tits inequality; see e.g. [12]).

For Krasnoselski-Mann iterations (x_n) of asymptotically nonexpansive mappings f we in general no longer have that $(d(x_n, f(x_n)))_n$ is nonincreasing. Hence we this time have to formulate our quantitative bound in terms of ‘metastability’ rather than the Π_3^0 -form of convergence. Using an appropriate form of theorem 2.15 the following result has been obtained (see also Kohlenbach-Lambov [69] which treats the uniformly convex **normed** case and the logical discussion provided there):

Theorem 6.9 (Kohlenbach-Leuştean [72]). *Let (X, d, W) be a uniformly convex hyperbolic space with a monotone modulus of uniform convexity η and $f : X \rightarrow X$ be asymptotically nonexpansive with sequence (k_n) .*

Assume that $K \geq 0$ is such that $\sum_{n=0}^{\infty} k_n \leq K$ and that $L \in \mathbb{N}, L \geq 2$ is such that

$$\frac{1}{L} \leq \lambda_n \leq 1 - \frac{1}{L} \text{ for all } n \in \mathbb{N}.$$

Let $x \in X$ and $b > 0$ be such that for any $\delta > 0$ there is $p \in X$ with

$$d(x, p) \leq b \wedge d(f(p), p) \leq \delta. \quad (2)$$

Then for all $\varepsilon \in (0, 1]$ and for all $g : \mathbb{N} \rightarrow \mathbb{N}$,

$$\exists N \leq \Phi(K, L, b, \eta, \varepsilon, g) \forall m \in [N, N + g(N)] (d(x_m, f(x_m)) < \varepsilon), \quad (3)$$

where

$$\Phi(K, L, b, \eta, \varepsilon, g) := h^{(M)}(0), \quad h(n) := g(n+1) + n + 2,$$

$$M := \left\lceil \frac{3 \left(5KD + D + \frac{11}{2} \right)}{\delta} \right\rceil, \quad D := e^K (b + 2),$$

$$\delta := \frac{\varepsilon}{L^2 F(K)} \cdot \eta \left((1 + K)D + 1, \frac{\varepsilon}{F(K)((1 + K)D + 1)} \right),$$

$$F(K) := 2(1 + (1 + K)^2(2 + K)).$$

Moreover, $N = h^{(i)}(0) + 1$ for some $i < M$.

The special case $g(n) := 0$ yields that for M as defined above

$$(*) \exists N \leq 2M (d(x_N, f(x_N)) < \varepsilon).$$

In the case of CAT(0)-spaces one obtains a quadratic bound on N in (*) (see [72]) which even for nonexpansive mappings and the special case of Hilbert space X is expected to be optimal.

For further results obtained by proof mining in the context of fixed point theory see [13, 14, 15, 16, 30, 60, 62, 64, 66, 69, 70, 71, 72, 83, 81]. Most of these results (except for [72] partly summarized above) are included in the survey [67].

7 Applications of proof mining in ergodic theory

Ergodic theory has close connections to metric fixed point theory and nonexpansive mappings f again play an important role. In particular, one studies the asymptotic behavior of the averaging operator defined by

$$A_n(x) := \frac{1}{n} S_n(x), \text{ where } S_n(x) := \sum_{i=0}^{n-1} f^i(x).$$

The context, typically, is that of Hilbert spaces (or, more generally, uniformly convex Banach spaces).

A classical result is the following

Theorem 7.1 (von Neumann mean ergodic theorem). *Let X be a Hilbert space and $f : X \rightarrow X$ a nonexpansive linear operator. Then for any point $x \in X$ the sequence $(A_n(x))_n$ defined above converges (in the Hilbert space norm).*

In [1], a detailed unwinding of a standard (ineffective) proof of this theorem is given. Since, as the authors show, there is (in general) no computable bound on the convergence itself, one again has (in order to obtain computational information) to move to the no-counterexample version (i.e. metastability in the sense of Tao) of this result:

$$\forall g : \mathbb{N} \rightarrow \mathbb{N} \forall \varepsilon > 0 \forall x \in X \exists n \forall i, j \in [n; n + g(n)] (\|A_i(x) - A_j(x)\| < \varepsilon).$$

From the general logical metatheorem proved in [31] and discussed in section 2 above (see theorem 2.15) it can be inferred (after some preprocessing of the proof) that one can extract a computable bound $\Phi(d, \varepsilon, g)$ on $\exists n$ that only depends on a norm upper bound $d \geq \|x\|$, ε and g (note that since f is linear and nonexpansive, one has $\|f(x)\| \leq \|x\|$ so that $\|f(x) - x\| \leq 2\|x\|$).

In [1] the following bound Φ is extracted:

Theorem 7.2 (Avigad-Gerhardy-Towsner [1]). *Let X and f be as in theorem 7.1. Then*

$$\forall g : \mathbb{N} \rightarrow \mathbb{N} \forall \varepsilon > 0 \forall x \in X \exists n \leq \Phi \forall i, j \in [n; n + g(n)] (\|A_i(x) - A_j(x)\| \leq \varepsilon),$$

where $\Phi = h^{(k)}(0)$ with $\rho := \lceil \frac{\|x\|}{\varepsilon} \rceil$, $k := 2^9 \rho^2$, $h(n) := n + 2^{13} \rho^4 \tilde{g}((n+1)\tilde{g}(2n\rho)\rho^2)$ and $\tilde{g}(n) := \max\{i + g(i) : i \leq n\}$.

An area closely related to ergodic theory is ‘topological dynamics’. For an early use of proof mining in connection with Furstenberg and Weiss’ proof (based on topological dynamics) of van der Waerden’s theorem see [32].

8 Applications of proof mining in computer science

In the previous section we have shown show interesting new computational information can be extracted using proof theoretic tools even from *prima facie* highly ineffective proofs in different areas of mathematics. This extraction of computational information in itself is an application of proof mining that it relevant to computer science as it opens up a general logic-based approach toward computational mathematics.

In this section we mention briefly some more specific uses of proof mining to proofs **inside** of theoretical computer science and logic as well experiments with machine-extractions of algorithms from (simple) proofs.

A normalization algorithm for typed λ -terms (and various extensions thereof) that has received quite some attention is the so-called ‘normalization by evaluation’ algorithm first described in Berger-Schwichtenberg [6] (though it has some roots in early work of Martin-Löf as well). While this algorithm was first found without the use of proof mining it was shown by U. Berger in [4] that it in fact coincides with the algorithm extractable from the standard Tait-Troelstra strong normalization proof using Kreisel’s so-called modified realizability (which can be viewed as a simplified form of functional interpretation that suffices here since the normalization proof can be carried out with constructive logic; see [66] for a detailed discussion of modified realizability and its relation to functional interpretation). Whereas the extraction in [4] was done by hand, more recently, a machine extraction of this algorithm has been carried out in [5] thereby comparing different tools: MinLog, Coq and Isabelle/HOL.

Normalization by evaluation can be viewed as a kind of reduction-free algorithm (based on a logical relation used to define the computability predicate).

In [52] we gave a reduction-free proof (based on monotone functional interpretation) for the fact that primitive recursive functionals in the sense of Gödel of type $\mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$ are uniformly continuous on $\{f : f \leq_1 g\}$ with a primitive recursive (in g) modulus of uniform continuity. Recently, M.-D. Hernest used his implementation ([38]) of (an optimized so-called ‘light’ version [37] of) monotone functional interpretation to machine-extract such moduli for concrete functionals with essentially optimal results (see [40]).

Various experiments with machine extractions of algorithms from (though rather simple) proofs in elementary arithmetic, combinatorics and algebra have been carried out e.g. in Berger-Schwichtenberg [7] (using a combination of modified realizability and a so-called refined Friedman/Dragalin-translation) which, in particular, treats Dickson's lemma and in Hernest [39] (using 'light' functional interpretation). An approach based on functional interpretation of Dickson's lemma and the Hilbert Basis Theorem, carried out (on paper) by A. Hertz [42], provides particularly good results in terms of complexity theory. From a practical point of view the algorithm extracted in Raffalli [91] (based on methods related to Krivine [79]) seems to be quite efficient in test cases. In [94], Schwichtenberg uses functional interpretation to extract an algorithm close to Euclid's from an almost trivial proof.

An interesting use of the Shoenfield variant of functional interpretation for the extraction of a new cut-elimination algorithm from the ineffective semantical proof of cut-elimination was recently given by G. Mints [87]. Conservation results of appropriate forms of weak König's lemma over systems of feasible analysis have been proved in Ferreira-Oliva [29] using their bounded functional interpretation ([28]).

References

- [1] Avigad, J., Gerhardy, P., Towsner, H., Local stability of ergodic averages. arXiv:0706.1512v1 [math.DS] (2007).
- [2] Baillon, J.B., Bruck, R.E., Reich, S., On the asymptotic behavior of nonexpansive mappings and semigroups in Banach spaces. *Houston J. Math.* **4**, pp. 1-9 (1978).
- [3] Bellin, G., Ramsey interpreted: a parametric version of Ramsey's theorem. In: *Logic and computation* (Pittsburgh, PA, 1987), pp. 17-37, *Contemp. Math.*, 106, Amer. Math. Soc., Providence, RI, (1990).
- [4] Berger, U., Program extraction from normalization proofs. In: M. Bezem et al.(eds.), *TLCA'93*, pp. 91-106. Springer LNCS **664** (1993).
- [5] Berger, U., Berghofer, S., Letouzey, P., Schwichtenberg, H., Program extraction from normalization proofs. *Studia Logica* **82**, pp. 25-49 (2006).
- [6] Berger, U., Schwichtenberg, H., An inverse of the evaluation functional for typed λ -calculus. In: Vemuri, R. (ed.), *Proc. of the Sixth Annual IEEE Symposium on Logic in Computer Science*, pp. 203-211. IEEE Computer Society, Los Alamitos, 1991.
- [7] Berger, U., Schwichtenberg, H., Seisenberger, M., The Warshall Algorithm and Dickson's Lemma: Two examples of realistic program extraction. *Journal of Automated Reasoning* **26**, pp. 205-221 (2001).

- [8] Bezem, M., Strongly majorizable functionals of finite type: a model for bar recursion containing discontinuous functionals. *J. Symbolic Logic* **50** pp. 652–660 (1985).
- [9] Björnestal, B.O., Continuity of the metric projection operator I-III. The preprint series of Department of Mathematics. Royal Institute of Technology. Stockholm, TRITA-MAT **17** (1974), **20** (1974), **12** (1975).
- [10] Bombieri, E., van der Poorten, A.J., Some quantitative results related to Roth's theorem. *J. Austral. Math. Soc. (Series A)* **45**, pp. 233-248 (1988).
- [11] Borwein, J., Reich, S., Shafrir, I., Krasnoselski-Mann iterations in normed spaces. *Canad. Math. Bull.* **35**, pp. 21-28 (1992).
- [12] Bridson, M.R., Haefliger, A., Metric spaces of non-positive curvature. Springer Verlag, Berlin (1999).
- [13] Briseid, E., Proof Mining Applied to Fixed Point Theorems for Mappings of Contractive Type. Master Thesis, 70pp., Oslo 2005.
- [14] Briseid, E.M., Fixed points of generalized contractive mappings. To appear in: *J. Nonlinear and Convex Analysis*.
- [15] Briseid, E.M., A rate of convergence for asymptotic contractions. *J. Math. Anal. Appl.* **330**, pp. 364-376 (2007).
- [16] Briseid, E.M., Some results on Kirk's asymptotic contractions. *Fixed Point Theory* **8**, No.1, pp. 17-27 (2007).
- [17] Briseid, E.M., Logical aspects of rates of convergence in metric spaces. In preparation.
- [18] Cheney, E.W., Approximation Theory. AMS Chelsea Publishing, 1966.
- [19] Coquand, Th., Sur un théorème de Kronecker concernant les variétés algébriques. *C.R. Acad. Sci. Paris, Ser. I* **338**, pp. 291-294 (2004).
- [20] Coquand, Th., Lombardi, H., Quitte, C., Generating non-Noetherian modules constructively. *Manuscripta mathematica* **115**, pp. 513-520 (2004).
- [21] Coste, M., Lombardi, H., Roy, M.F., Dynamical methods in algebra: effective Nullstellensätze. *Ann. Pure Appl. Logic* **111**, pp. 203-256 (2001).
- [22] Delzell, C., Continuous sums of squares of forms. In: *Proc. L.E.J. Brouwer Centenary Symposium*. Noordwijkerhout, pp. 65-75 (1981).
- [23] Delzell, C., Case distinctions are necessary for representing polynomials as sums of squares. *Proc. Herbrand Symposium*, pp. 87-103 (1981).
- [24] Delzell, C., A finiteness theorem for open semi-algebraic sets, with applications to Hilbert's 17th problem. *Contemporary Math.* **8**, pp. 79-97 (1982).
- [25] Delzell, C., A continuous, constructive solution to Hilbert's 17th problem. *Invent. math.* **76**, pp. 365-384 (1984).

- [26] Delzell, C., Kreisel's unwinding of Artin's proof-Part I. In: Odifreddi, P., Kreiseliana, 113-246, A K Peters, Wellesley, MA (1996).
- [27] Feferman, S., Kreisel's 'Unwinding Program'. In: P. Odifreddi (ed.), Kreiseliana: about and around Georg Kreisel, A.K. Peters, Wellesley Massachusetts, pp. 247-273 (1996).
- [28] Ferreira, F., Oliva, P., Bounded functional interpretation. *Ann. Pure Appl. Logic* **135**, pp. 73-112 (2005).
- [29] Ferreira, F., Oliva, P., Bounded functional interpretation and feasible analysis. *Ann. Pure Appl. Logic* **145**, pp. 115-129 (2007).
- [30] Gerhardy, P., A quantitative version of Kirk's fixed point theorem for asymptotic contraction. *J. Math. Anal. Appl.* **316**, pp. 339-345 (2006).
- [31] Gerhardy, P., Kohlenbach, U., General logical metatheorems for functional analysis. To appear in: *Trans. Amer. Math. Soc.*
- [32] Girard, J.-Y., *Proof Theory and Logical Complexity Vol.I. Studies in Proof Theory.* Bibliopolis (Napoli) and Elsevier Science Publishers (Amsterdam) 1987.
- [33] Goebel, K., Kirk, W.A., A fixed point theorem for asymptotically nonexpansive mappings, *Proc. Amer. Math. Soc.* **35** (1972), 171-174.
- [34] Goebel, K., Kirk, W.A., Iteration processes for nonexpansive mappings. In: Singh, S.P., Thomeier, S., Watson, B., eds., *Topological Methods in Nonlinear Functional Analysis. Contemporary Mathematics* **21**, AMS, pp. 115-123 (1983).
- [35] Goebel, K., Reich, S., *Uniform convexity, hyperbolic geometry, and nonexpansive mappings*, Marcel Dekker, Inc., New York and Basel, 1984.
- [36] Gödel, K., Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica* **12**, pp. 280-287 (1958).
- [37] Hernest, M.-D., Light functional interpretation. An optimization of Gödel's technique towards the extraction of (more) efficient programs from (classical) proofs. In: Ong, L. (ed.), *CSL 2005, Springer LNCS* **3634**, pp. 477-492 (2005).
- [38] Hernest, M.-D., The MinLog proof-system for Dialectica program-extraction. Free software available at <http://www.brics.dk/~danher/MinLogForDialectica>.
- [39] Hernest, M.-D., Light Dialectica program extraction for a classical Fibonacci proof. *Electronic Notes in Theoretical Computer Science* **171**, pp. 43-53 (2007).
- [40] Hernest, M.-D., Synthesis of moduli of uniform continuity by the monotone Dialectica interpretation in the proof-system MinLog. *Electronic Notes in Theoretical Computer Science* **174**, pp. 141-149 (2007).
- [41] Hernest, M.-D., Kohlenbach, U., A complexity analysis of functional interpretations. *Theoretical Computer Science* **338**, pp. 200-246 (2005).
- [42] Hertz, A., A constructive version of the Hilbert basis theorem. Master Thesis, Carnegie Mellon University (2004).

- [43] Hilbert, D., Über das Unendliche. *Math. Ann.* **95**, pp. 161-190 (1926).
- [44] Howard, W.A., Hereditarily majorizable functionals of finite type. In: Troelstra (ed.), *Metamathematical investigation of intuitionistic arithmetic and analysis*, pp. 454-461. Springer LNM 344 (1973).
- [45] Ishikawa, S., Fixed points and iterations of a nonexpansive mapping in a Banach space. *Proc. Amer. Math. Soc.* **59**, pp. 65-71 (1976).
- [46] Jackson, D., Note on a class of polynomials of approximation. *Trans. Amer. Math. Soc.* **22**, pp. 320-326 (1921).
- [47] Ketonen, J., Solovay, R., Rapidly growing Ramsey functions. *Ann. Math.* **113**, pp. 267-314 (1981).
- [48] Kirk, W.A., Geodesic geometry and fixed point theory. *Seminar of Mathematical Analysis (Malaga/Seville, 2002/2003)*, pp. 195-225, Colecc. Abierta, 64, Univ. Seville Secr. Publ., Seville (2003).
- [49] Kirk, W.A., Geodesic geometry and fixed point theory II. In: G-Falset, J., L-Fuster, E., Sims, B. (eds.), *Proc. International Conference on Fixed Point Theory, Valencia 2003*, pp. 113-142, Yokohama Press 2004.
- [50] Kirk, W.A., Sims, B. (eds.), *Handbook of Metric Fixed Point Theory*. Kluwer Academic Publishers, Dordrecht, Boston, London, xi+703pp. (2001).
- [51] Kleene, S.C., *Introduction to Metamathematics*. North-Holland (Amsterdam), Noordhoff (Groningen), Van Nostrand (New-York) 1952.
- [52] Kohlenbach, U., Pointwise hereditary majorization and some applications. *Arch. Math. Logic* **31**, pp. 227-241 (1992).
- [53] Kohlenbach, U., Effective bounds from ineffective proofs in analysis: an application of functional interpretation and majorization. *J. Symbolic Logic* **57**, pp. 1239-1273 (1992).
- [54] Kohlenbach, U., Effective moduli from ineffective uniqueness proofs. An unwinding of de La Vallée Poussin's proof for Chebycheff approximation. *Ann. Pure Appl. Logic* **64**, pp. 27-94 (1993).
- [55] Kohlenbach, U., New effective moduli of uniqueness and uniform a-priori estimates for constants of strong unicity by logical analysis of known proofs in best approximation theory. *Numer. Funct. Anal. and Optimiz.* **14**, pp. 581-606 (1993).
- [56] Kohlenbach, U., Analysing proofs in analysis. In: W. Hodges, M. Hyland, C. Steinhorn, J. Truss, editors, *Logic: from Foundations to Applications. European Logic Colloquium* (Keele, 1993), pp. 225-260, Oxford University Press (1996).
- [57] Kohlenbach, U., Mathematically strong subsystems of analysis with low rate of growth of provably recursive functionals. *Arch. Math. Logic* **36**, pp. 31-71 (1996).
- [58] Kohlenbach, U., Arithmetizing proofs in analysis. In: Larrazabal, J.M., Lascar, D., Mints, G. (eds.), *Logic Colloquium '96*, Springer Lecture Notes in Logic **12**, pp. 115-158 (1998).

- [59] Kohlenbach, U., On the no-counterexample interpretation. *J. Symbolic Logic* **64**, pp. 1491-1511 (1999).
- [60] Kohlenbach, U., On the computational content of the Krasnoselski and Ishikawa fixed point theorems. In: *Proceedings of the Fourth Workshop on Computability and Complexity in Analysis*, J. Blanck, V. Brattka, P. Hertling (eds.), Springer LNCS **2064**, pp. 119-145 (2001).
- [61] Kohlenbach, U., A quantitative version of a theorem due to Borwein-Reich-Shafrir. *Numer. Funct. Anal. and Optimiz.* **22**, pp. 641-656 (2001).
- [62] Kohlenbach, U., Uniform asymptotic regularity for Mann iterates. *J. Math. Anal. Appl.* **279**, pp. 531-544 (2003).
- [63] Kohlenbach, U., Some logical metatheorems with applications in functional analysis. *Trans. Amer. Math. Soc.* vol. 357, no. 1, pp. 89-128 (2005).
- [64] Kohlenbach, U., Some computational aspects of metric fixed point theory. *Nonlinear Analysis* **61**, pp. 823-837 (2005).
- [65] Kohlenbach, U., A logical uniform boundedness principle for abstract metric and hyperbolic spaces. *Electronic Notes in Theoretical Computer Science* **165** (Proc. WoLLIC 2006), pp. 81-93 (2006).
- [66] Kohlenbach, U., *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*. Book in preparation for 'Springer Monographs in Mathematics'. Approx. 500pp. Expected to appear: Spring 2008.
- [67] Kohlenbach, U., Effective uniform bounds from proofs in abstract functional analysis. To appear in: Cooper, B., Loewe, B., Sorbi, A. (eds.), 'CiE 2005 New Computational Paradigms: Changing Conceptions of What is Computable'. Springer Publisher.
- [68] Kohlenbach, U., Gödel's functional interpretation and its use in current mathematics. To appear in: *Horizons of Truth, Gödel Centenary*. Cambridge University Press.
- [69] Kohlenbach, U., Lambov, B., Bounds on iterations of asymptotically quasi-nonexpansive mappings. In: Falset, J.G., Fuster, E.L., Sims, B. (eds.), *Proc. International Conference on Fixed Point Theory and Applications*, Valencia 2003, pp. 143-172, Yokohama Publishers (2004).
- [70] Kohlenbach, U., Leuştean, L., Mann iterates of directionally nonexpansive mappings in hyperbolic spaces. *Abstr. Appl. Anal.* vol. 2003, no.8, pp. 449-477 (2003).
- [71] Kohlenbach, U., Leuştean, L., The approximate fixed point property in product spaces. *Nonlinear Analysis* **66**, pp. 806-818 (2007).
- [72] Kohlenbach, U., Leuştean, L., Asymptotically nonexpansive mappings in uniformly convex hyperbolic spaces. arXiv:0707.1626 [math.LO] (2007).
- [73] Kohlenbach, U., Oliva, P., Proof mining in L_1 -approximation. *Ann. Pure Appl. Logic* **121**, pp. 1-38 (2003).

- [74] Kohlenbach, U., Oliva, P., Proof mining: a systematic way of analysing proofs in mathematics. *Proc. Steklov Inst. Math.* **242**, pp. 1-29 (2003).
- [75] Kreisel, G., On the interpretation of non-finitist proofs, part I. *J. Symbolic Logic* **16**, pp.241-267 (1951).
- [76] Kreisel, G., On the interpretation of non-finitist proofs, part II: Interpretation of number theory, applications. *J. Symbolic Logic* **17**, pp. 43-58 (1952).
- [77] Kreisel, G., Finiteness theorems in arithmetic: an application of Herbrand's theorem for Σ_2 -formulas. *Proc. of the Herbrand symposium (Marseille, 1981)*, North-Holland (Amsterdam), pp. 39-55 (1982).
- [78] Kreisel, G., Macintyre, A., Constructive logic versus algebraization I. *Proc. L.E.J. Brouwer Centenary Symposium (Noordwijkerhout 1981)*, North-Holland (Amsterdam), pp. 217-260 (1982).
- [79] Krivine, J.-L., Dependent choice, 'quote' and the clock. *Theoretical Computer Science* **308**, pp. 259-276 (2003).
- [80] Kroó, A., On the continuity of best approximations in the space of integrable functions. *Acta Mathematica Academiae Scientiarum Hungaricae* **32**, pp. 331-348 (1978).
- [81] Lambov, B., Rates of convergence of recursively defined sequences. In: Brattka, V., Staiger, L., Weihrauch, E., *Proc. of the 6th Workshop on Computability and Complexity in Analysis*, vol. 120 of *Electronic Notes in Theoretical Computer Science*, pp. 125-133 (2005).
- [82] Leuştean, L., Proof mining in \mathbb{R} -trees and hyperbolic spaces. *Electronic Notes in Theoretical Computer Science* **165** (*Proc. WoLLIC 2006*), pp. 95-106 (2006).
- [83] Leuştean, L., A quadratic rate of asymptotic regularity for CAT(0)-spaces. *J. Math. Anal. Appl.* **325**, pp. 386-399 (2007).
- [84] Luckhardt, H., Herbrand-Analysen zweier Beweise des Satzes von Roth: Polynomiale Anzahlschranken. *J. Symbolic Logic* **54**, pp. 234-263 (1989).
- [85] Luckhardt, H., Bounds extracted by Kreisel from ineffective proofs. In: Odifreddi, P., *Kreiseliana*, 289-300, A K Peters, Wellesley, MA, 1996.
- [86] Macintyre, A., The mathematical significance of proof theory. *Phil. Trans. R. Soc. A* **363**, pp. 2419-2435 (2005).
- [87] Mints, G.E., Unwinding a non-effective cut elimination proof. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (Eds.): *Computer Science - Theory and Applications, First International Computer Science Symposium in Russia, CSR 2006, St. Petersburg, Russia, June 8-12, 2006, Proceedings*. Springer LNCS **3967**, pp. 259-269 (2006).
- [88] Odifreddi, P. (ed.), *Kreiseliana. About and around Georg Kreisel*. A K Peters, Wellesley, Massachusetts, xiii+495 pp. (1996).
- [89] Oliva, P., On the computational complexity of best L_1 -Approximation. *Math. Logic. Quart.* **48**, suppl. I, pp. 66-77 (2002).

- [90] Oliva, P., Understanding and using Spector's bar recursive interpretation of classical analysis. In: Proceedings of CiE 2006, Springer LNCS **3988**, pp. 423-434 (2006).
- [91] Raffalli, C., Getting results from programs extracted from classical proofs. Theoretical Computer Science **323**, pp. 49-70 (2004).
- [92] Reich, S., Shafir, I., Nonexpansive iterations in hyperbolic spaces. Nonlinear Analysis, Theory, Methods and Applications **15**, pp. 537-558 (1990).
- [93] Roth, K.F., Rational approximations to algebraic numbers. *Mathematika* **2**, pp. 1-20 (1955).
- [94] Schwichtenberg, H., Dialectica interpretation of well-founded induction. To appear in: *Math. Log. Quart.*
- [95] Shoenfield, J.S., *Mathematical Logic*. Addison-Wesley Publishing Company (Reading, Massachusetts) 1967.
- [96] Spector, C., Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In: *Recursive function theory, Proceedings of Symposia in Pure Mathematics*, vol. 5 (J.C.E. Dekker (ed.)), AMS, Providence, R.I., pp. 1-27 (1962).
- [97] Streicher, T., Kohlenbach, U., Shoenfield is Gödel after Krivine. *Math. Log. Quart.* **53**, pp. 176-179 (2007).
- [98] Tao, T., Soft analysis, hard analysis, and the finite convergence principle. Essay posted May 23, 2007. Available at: <http://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/>.
- [99] Tao, T., Norm convergence of multiple ergodic averages for commuting transformations. arXiv:0707.1117v1 [math.DS] (2007).
- [100] Weiermann, A., A classification of rapidly growing Ramsey functions. *Proc. Amer. Math. Soc.* 132, no. 2, pp. 553-561 (2004).
- [101] Weiermann, A., Phasenübergänge in Logik und Kombinatorik. *DMV-Mitteilungen* **13**, no. 3, pp. 152-156 (2005).

THE NATURAL COMPUTING COLUMN

BY

GRZEGORZ ROZENBERG

Leiden University, Leiden Center for Natural Computing
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
`rozenber@liacs.nl`

MACHINES OF SYSTEMS BIOLOGY*

Luca Cardelli, Microsoft Research

Abstract

Living cells are extremely well-organized autonomous systems, consisting of discrete interacting components. Key to understanding and modeling their behavior is modeling their system organization. Four distinct chemical toolkits (classes of macromolecules) have been characterized, each combinatorial in nature. Each toolkit consists of a small number of simple components that are assembled (polymerized) into complex structures that interact in rich ways. Each toolkit abstracts away from chemistry; it embodies an abstract machine with its own instruction set and its own peculiar interaction model. These interaction models are highly effective, but are not ones commonly used in computing: proteins stick together, genes have fixed output, membranes carry activity on their surfaces. Biologists have invented a number of notations attempting to describe these abstract machines and the processes they implement. Moving up from molecular biology, *systems biology* aims to understand how these interaction models work, separately and together.

*Published in Transactions on Computational Systems Biology. III, LNBI 3737, pp 145-168, © Springer-Verlag Berlin Heidelberg 2005. With kind permission of Springer Science and Business Media.

Preface to this reprint. In this paper from 2005 I tried to summarize what I found most remarkable about cellular organization, that is the fact that *cells compute*, and that computation is one of their most important functions. We can easily understand “why” cells need to compute: unicellular organisms, in the active search for food and the active avoidance of predators, need to deploy increasingly sophisticated and highly optimized biochemical *algorithms*, in an information processing arms race with each other. (It’s hard for me to use a different term than *algorithms*, although *pathways* and *networks* are common terms.) However, we do not really understand “how” cells compute, except in very basic cases. Among the many, many things that are *not* known about biology, this is the one that should concern us most, particularly because information processing at the cellular level is so basic. Unlike more evolved information processing systems in higher organisms, here we know a lot about cellular information coding and basic processing steps. And yet, the programming models that arise from those “abstract machines” of biochemistry are unfamiliar ones. What I find even more remarkable is that cells have three separate Turing-complete (in principle) mechanisms at their disposal: proteins, genes, and membranes, which all often cooperate on information processing tasks.

1 Introduction

Following the discovery of the structure of DNA, just over 50 years ago, molecular biologists have been unraveling the functioning of cellular components and networks. The amount of molecular-level knowledge accumulated so far is absolutely amazing. And yet we cannot say that we understand *how a cell works*, at least not to the extent of being able to easily modify or repair a cell. The process of understanding cellular components is far from finished, but it is becoming clear that simply obtaining a full part list will not tell us how a cell works. Rather, even for substructures that have been well characterized, there are significant difficulties in understanding how components interact as *systems* to produce the observed behaviors. Moreover, there are just too many components, and too few biologists, to analyze each component in depth in reasonable time. Similar problems occur also at each level of biological organization above the cellular level.

Enter *systems biology*, which has two aims. The first is to obtain massive amounts of information about whole biological systems, via high-throughput experiments that provide relatively shallow and noisy data. The Human Genome Project is a prototypical example: the knowledge it accumulated is highly valuable, and was obtained in an automated and relatively efficient way, but is just the beginning of understanding the human genome. Similar effort are now un-

derway in *genomics* (finding the collection of all genes, for many genomes), in *transcriptomics* (the collection of all actively transcribed genes), in *proteomics* (the collection of all proteins), and in *metabolomics* (the collection of all metabolites). *Bioinformatics* is the rapidly growing discipline tasked with collecting and analyzing such *omics* data.

The other aim of systems biology is to build, with such data, a science of the *principles of operation* of biological systems, based on the *interactions between components*. Biological systems are obviously well-engineered: they are very complex and yet highly structured and robust. They have only one major engineering defect: they have not been designed, in any standard sense, and so are not laid out as to be easily understood. It is not clear that any of the engineering principles of operations we are currently familiar with are fully applicable. Understanding such principles will require an interdisciplinary effort, using ideas from physics, mathematics, and computing. These, then, are the promises of systems biology: it will teach us new principles of operation, likely applicable to other sciences, and it will leverage other sciences to teach us *how cells work* in an actionable way.

In this paper, we look at the organization of biological systems from an information science point of view. The main reason is quite pragmatic: as we increasingly map out and understand the complex interactions of biological components, we need to *write down* such knowledge, in such a way that we can inspect it, animate it, and understand its principles. For genes, we can write down long but structurally simple strings of nucleotides in a 4-letter alphabet, that can be stored and queried. For proteins we can write down strings of amino acids in a 20-letter alphabet, plus three-dimensional information, which can be stored and queried with a little more difficulty. But how shall we write down *biological processes*, so that they can be stored and queried? It turns out that biologists have already developed a number of informal notation, which will be our starting points. These notations are abstractions over chemistry or, more precisely, are abstractions over a number of biologically relevant chemical toolkits.

2 Biochemical Toolkits

Apart from small molecules such as water and some metabolites, there are four large classes of *macromolecules* in a cell. Each class is formed by a small number of units that can be combined systematically to produce structures of great complexity. That is, to produce both individual molecules of essentially unbounded size, and multi-molecular complexes.

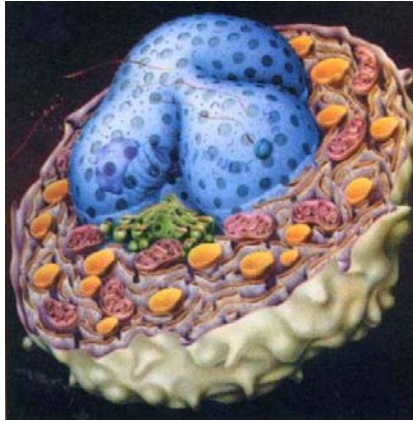
The four classes of macromolecules are as follows. Different members of each class can have different functions (structure, energy storage, etc.). We focus on

the most combinatorial, information-bearing, members of each class:

- *Nucleic acids.* Five kinds of *nucleotides* combine in ordered sequences to form two nucleic acid polymers: *DNA* and *RNA*. As data structures, *RNA* is **lists**, and *DNA* is **doubly-linked lists**. Their most prominent role is in coding information, although they also have other important functions.
- *Proteins.* About 20 kinds of *amino acids* combine linearly to form proteins. Each protein folds in a specific three-dimensional shape (sometimes from multiple strings of amino acids). The main and most evolutionary stable property of a protein is not the exact sequence of amino acids that make it up, nor the exact folding process, but its collection of surface *features* that determine its function. As data structures, proteins are **records** of features and, since these features are often active and stateful, they are **objects** in the object-oriented programming sense.
- *Lipids:* Among the lipids, *phospholipids* have a modular structure and can self-assemble into closed double-layered sheets (membranes). Membranes differ in the proportion and orientation of different phospholipids, and in the kinds of proteins that are attached to them. As data structures, membranes are **containers**, but with an active surface that acts as an **interface** to its contents.
- *Carbohydrates:* Among the carbohydrates, *oligosaccharides* are sugars linked in a branching structure. As data structures, oligosaccharides are **trees**. They have a vast number of configurations, and a complex assembly processes. *Polysaccharides* form even bigger structures, although usually of a semi-regular kind (rods, meshes). We do not consider carbohydrates further, although they are probably just as rich and interesting as the other toolkits. They largely have to do with energy storage and with cell surface and extra-cellular structures. But it should be noted that they too have a computational role, in forming unique surface structures that are subject to recognition. Many proteins are grafted with carbohydrates, through a complex assembly process called glycosylation.

Out of these four toolkits arises all the *organic chemicals*, composing, e.g., eukaryotic cells (Figure 1, [32] p.1). Each toolkit has specific structural properties (as emphasized by the bolded words above), systematic functions, and a peculiarly rich and flexible mode of operation. These peculiar modes of operation and systematic functions are what we want to emphasize, beyond their chemical realization.

Cells are without doubt, in many respects, information processing devices.



From MOLECULAR CELL BIOLOGY, 4/e by Harvey Lodish, et. al. ©1986, 1990, 1995, 2000 by W.H. Freeman and Company. Figure 1-1, Page 1. Used with permission.

Figure 1: Eukaryotic Cell. Eukaryotic cells have an extensive array of membrane-bound compartments and organelles with up to 4 levels of nesting. The nucleus is a double membrane. The external membrane is less than 10% of the total.

Without properly processing information from their environment, they soon die for lack of nutrients or for predation. The blueprint of a cell, needed for its functioning and reproduction, is stored as digital information in the genome; an essential step of reproduction is the copying of that digital information. There are hints that information processing in the genome of higher organisms is much more sophisticated than currently generally believed [33].

We could say that cells are based on chemistry that also perform some information processing. But we take a more extreme position, namely that cells are chemistry *in the service* of information processing. Hence, we look for information processing machinery within the cellular machinery, and we try to understand the functioning of the cell in terms of information processing, instead of chemistry. In fact, we can readily find such information processing machinery in the chemical toolkits that we just described, and we can switch fairly smoothly from the classical description of cellular functioning in terms of classes of macromolecules, to a description based on abstract information-processing machines.

3 Abstract Machines

An *abstract machine* is a fictional information-processing device that can, in principle, have a number of different physical realizations (mechanical, electronic,

biological, or even software). An abstract machine is characterized by:

- A collection of discrete states.
- A collection of operations (or events) that cause discrete transitions between states.

The evolution of states through transitions can in general happen concurrently. The adequacy of this generic model for describing complex systems is argued, e.g., in [22].

Each of the chemical toolkits we have just described can be seen as a separate abstract machine with an appropriate set of states and operations. This abstract interpretations of chemistry is by definition fictional, and we must be aware of its limitation. However, we must also be aware of the limitations of *not* abstracting, because then we are in general limited to work at the lowest level of reality (quantum mechanics) without any hope of understanding higher principles of organization. The abstract machines we consider are each grounded in a different chemical toolkit (nucleotides, amino acids, and phospholipids), and hence have some grounding in reality. Moreover, each abstract machine corresponds to a different kind of informal *algorithmic notation* that biologists have developed (Figure 2, bubbles): this is further evidence that abstract principles of organization are at work.

The *Gene Machine* (better known as Gene Regulatory Networks) performs information processing tasks within the cell. It regulates all other activities, including assembly and maintenance of the other machines, and the copying of itself. The *Protein Machine* (better known as Biochemical Networks) performs all mechanical and metabolic tasks, and also some signal processing. The *Membrane Machine* (better known as Transport Networks) separates different biochemical environments, and also operates dynamically to transport substances via complex, discrete, multi-step processes.

These three machines operate in concert and are highly interdependent. Genes instruct the production of proteins and membranes, and direct the embedding of proteins within membranes. Some proteins act as messengers between genes, and others perform various gating and signaling tasks when embedded in a membrane. Membranes confine cellular materials and bear proteins on their surfaces. In eukaryotes, membranes confine the genome, so that local conditions are suitable for regulation, and confine other reactions carried out by proteins in specialized vesicles.

Therefore, to understand the functioning of a cell, one must understand also how the various machines interact. This involves considerable difficulties (e.g. in simulations) because of the drastic difference in time and size scales: proteins

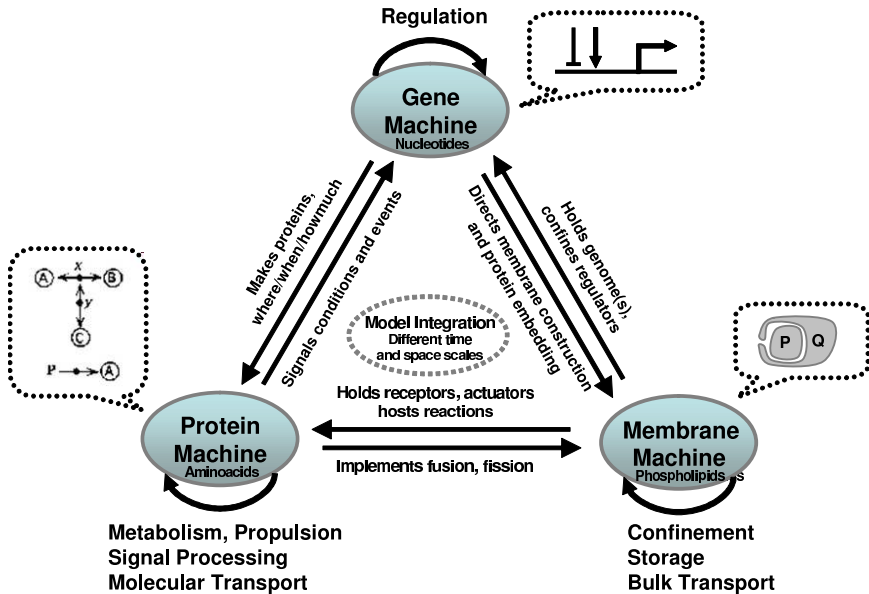


Figure 2: **Abstract Machines, Molecular Basis, and Notations**

interacts in tiny fractions of a second, while gene interactions take minutes; proteins are large molecules, but are dwarfed by chromosomes, and membranes are larger still. Before looking at the interactions among the different machine in more detail, we start by discussing each machine separately.

4 The Protein Machine (Biochemical Networks)

4.1 Principles of Operation

Proteins are long folded-up strings of amino acids with precisely determined, but often mechanically flexible, three-dimensional shapes. If two proteins have surface regions that are complementary (both in shape and in charge), they may stick to each other like Velcro, forming a protein **complex** where a multitude of small atomic forces creates a strong bond between individual proteins. They can similarly stick highly selectively to other substances. During a **complexation** event, a protein may be bent or opened, thereby revealing new interaction surfaces. Through complexation many proteins act as enzymes: they bring together compounds, including other proteins, and greatly facilitate chemical reactions between them without being themselves affected.

Proteins may also chemically modify each other by attaching or removing small phosphate groups at specific sites. Each such site acts as a boolean switch: over a dozen of them can be present on a single protein. Addition of a phosphate group (**phosphorylation**) is performed by an enzyme that is then called a **kinase**. Removal of a phosphate group (**dephosphorylation**) is performed by an enzyme that is then called a **phosphatase**. For example, a *protein phosphatase kinase kinase* is a protein that phosphorylates a protein that phosphorylates a protein that dephosphorylates a protein. Each (de-)phosphorylation may reveal new interaction surfaces, and each surface interaction may expose new phosphorylation sites.

It turns out that a large number of protein interactions work at the level of abstraction just described. That is, we can largely ignore chemistry and the protein folding process, and think of each protein as a collection of features (binding sites and phosphorylation sites) whose availability is affected by (de-)complexation and (de-)phosphorylation interactions. This abstraction level is emphasized in Kohn's Molecular Interaction Maps graphical notation [29][27] (Figure 4).

We can describe the operation of the *protein machine* as follows (Figure 3). Each protein is a collection of *sites* and *switches*; each of those can be, at any given time, either *available* or *unavailable*. Proteins can join at matching sites, to form bigger and bigger *complexes*. The availability of sites and switches in a complex is the *state* of the complex. A *system* is a multiset of (disjoint) complexes, each in a given state.

The protein machine has two kinds of operations. (1) An available switch on a complex can be turned on or off, resulting in a new state where a new collection of switches and sites is available. (2) Two protein complexes can combine at available sites, or one complex can split into two, resulting in a new state where a new collection of switches and sites is available.

Who is driving the switching and binding? Other proteins do. There are tens of thousands of proteins in a cell, so the protein machine has tens of thousands of “primitive instructions”; each with a specific way of acting on other proteins (or metabolites). For each cellular subsystem one must list the proteins involved, and how each protein interacts with the other proteins in terms of switching and binding.

4.2 Notations

Finding a suitable language in which to cast such an abstraction is a non-trivial task. Kohn designed a graphical notation, resulting in pictures such as Figure 4 [29]. This was a tremendous achievement, summarizing hundreds of technical papers in page-sized pictures, while providing a sophisticated and expressive notation that could be translated back into chemical equations according to semi-formal guidelines. Because of this intended chemical semantics, the dynamics of

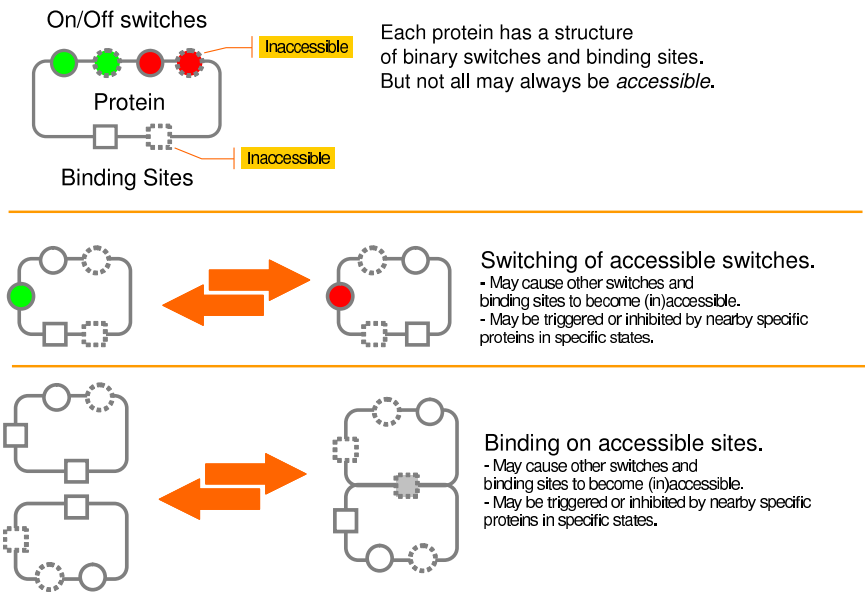


Figure 3: The Protein Machine Instruction Set

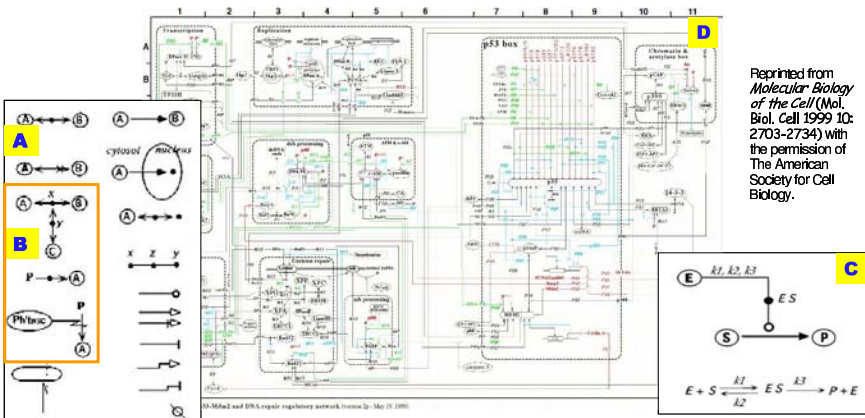


Figure 4: **Molecular Interaction Maps Notation.** From [29]. **A:** graphical primitives. **B:** complexation and phosphorylation. **C:** enzymatic diagram and equivalent chemical reactions. **D:** map of the p53-Mdm2 and DNA Repair Regulatory Network.

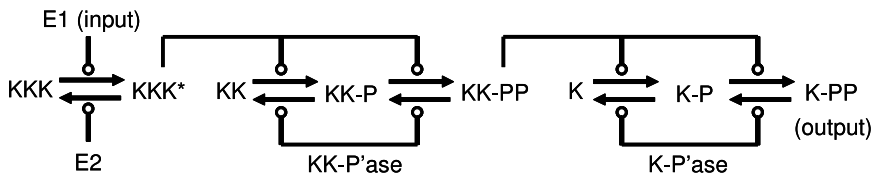


Figure 5: **MAPK Cascade**

a systems is implied in Kohn’s notation, but only by translation to chemical (and hence kinetic) equations. The notation itself has no dynamics, and this is one of its main limitation. The other major limitation is that, although graphically appealing, it tends to stop being useful when overflowing the borders of a page or of a whiteboard (the original Kohn maps span several pages).

Other notations for the protein machine can be devised. Kitano, for example, improved on the conciseness, expressiveness, and precision of Kohn’s notation [28], but further sophistication in graphical notation is certainly required along the general principles of [18]. A different approach is to devise a textual notation, which inherently has no “page-size” limit and can better capture dynamics; examples are Bio-calculus [38], and most notably κ -calculus [14][15], whose dynamics is fully formalized. But one may not need to invent completely new formalisms. Regev and Shapiro, in pioneering work [49][47], described how to represent chemical and biochemical interactions within existing process calculi (π -calculus). Since process calculi have a well understood dynamics (better understood, in fact, than most textual notations that one may devise just for the purpose), that approach also provides a solid basis for studying systems expressed in such a notation. Finally, some notations incorporate both continuous and discrete aspects, as in Charon [3] and dL-systems [45].

4.3 Example: MAPK Cascade

The relatively simple Kohn map in Figure 5 (adapted from [25]) describes the behavior of a circuit that causes Boolean-like switching of an output signal in presence of a very weak input signal. (It can also be described as a list of 10 chemical reactions, or of 25 differential/ algebraic equations, but then the network structure is not so apparent.) This network, generically called a MAPK cascade, has multiple biochemical implementations and variations. The components are proteins (enzymes, kinases, phosphatases, and intermediaries). The circle-arrow Kohn symbol for “enzyme-assisted reaction” can signify here either a complexation that facilitates a reaction, or a phosphorylation/dephosphorylation, depending on the specific proteins.

The system initially contains reservoirs of chemicals KKK, KK, and K (say, 100 molecules each), which are transformed by the cascade into the kinases KKK*, KK-PP, and K-PP respectively. Enzymes E2, KK-Phosphatase and K-Phosphatase are always available (say, 1 molecule each), and tend to drive the reactions back. Appearance of the input enzyme E1 in very low numbers (say, less than 5) causes a sharp (Boolean-like 0-100) transition in the concentration of the output K-PP. The concentrations of the intermediaries KK-PP, and especially KKK*, raise in a much smoother, non-Boolean-like, fashion. Given the mentioned concentrations, the network works fine by setting all reaction rates to equal values.

To notice here is that the detailed description of each of the individual proteins, with their folding processes, surface structures, interaction rates under different conditions, etc. could take volumes. But what makes this signal processing network work is the structure of the network itself, and the relatively simple interactions between the components.

4.4 Summary

The fundamental flavor of the Protein Machine is: *fast synchronous binary interactions*. Binary because interactions occur between two complementary surfaces, and because the likelihood of three-party instantaneous chemical interactions can be ignored. Synchronous because both parties potentially feel the effect of the interaction, when it happens. Fast because individual chemical reactions happen at almost immeasurable speeds. The parameters affecting reaction speed, in a well-stirred solution, are just a reaction-specific rate constant having to do with surface affinity, plus the concentrations of the reagents (and the temperature of the solution, which is usually assumed constant). Concentration affects the likelihood of molecules randomly finding each other by Brownian motion. Note that Brownian motion is surprisingly effective at a cellular scale: a molecule can “scan” the equivalent volume of a bacteria for a match in 1/10 of a second, and it will in fact scan such a bounded volume because random paths in 3D do not return to the origin.

5 The Gene Machine (Gene Regulatory Networks)

5.1 Principles of Operation

The *central dogma of molecular biology* states that DNA is transcribed to RNA, and RNA is translated to proteins (and then proteins do all the work). This dogma no longer paints the full picture, which has become considerably more detailed

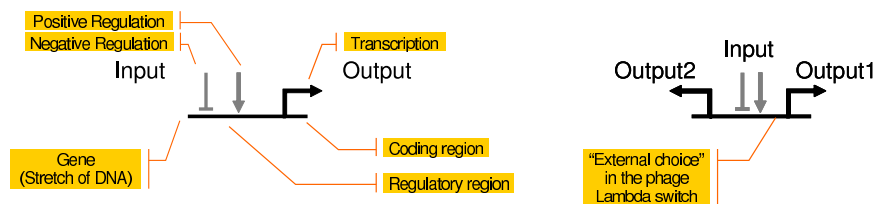


Figure 6: The Gene Machine Instruction Set

in recent years. Without entering into a very complex topic [33], let us just note that some proteins go back and bind to DNA. Those proteins are called **transcription factors** (either **activators** or **repressors**); they are produced for the purpose of allowing one gene (or signaling pathway) to communicate with other genes. Transcription factors are not simple messages: they are proteins, which means they are subject to complexation, phosphorylation, and programmed degradation, which all have a role in gene regulation.

A **gene** is a stretch of DNA consisting of two (not necessarily contiguous or unbroken) regions: an *input (regulatory) region*, containing **protein binding sites** for transcription factors, and an *output (coding) region*, coding for one or more proteins that the gene produces. Sometimes there are two coding regions, in opposite directions [46], on count of DNA being a doubly-linked list. Sometimes two genes overlap on the same stretch of DNA.

The output region functions according to the **genetic code**: a well understood and almost universal table mapping triplets of nucleotides to one of about 20 amino acids, plus start and stop triplets. The input region functions according to a much more complex code that is still poorly understood: transcription factors, by their specific 3D shapes, bind to specific nucleotide sequences in the input region, with varying binding strength depending of the precision of the match.

Thus, the gene machine, although entirely determined by the digital information coded in DNA, is not entirely digital in functioning: a digitally encoded protein, translated and folded-up, uses its “analog” shape to recognize another digital string and promote the next step of translation. Nonetheless, it is customary to ignore the details of this process, and simply measure the effectiveness with which (the product of) a gene affects another gene. This point of view is reflected in standard notation for gene regulatory networks (Figure 7).

In Figure 6, a gene is seen as a hardware gate, and the genome can be seen as a vast circuit composed of such gates. Once the performance characteristics of each gate is understood, one can understand or design circuits by combining gates, almost as one would design digital or analog hardware circuits. The performance characteristics of each gene in a genome is probably unique. Hence, as in

the protein machine, we are going to have thousands of “primitive instructions”: one for each gene.

A peculiarity of the gene machine is that a set of gates also determines the network connectivity. This is in contrast with a hardware circuit, where there is a collection of gates out of a very small set of “primitive gates”, and then a separate wiring list. Each gene has a *fixed output*; the protein the gene codes for (although post-processing may vary such output). Similarly, a gene has a *fixed input*: the fixed set of binding sites in the input region. Therefore, by knowing the nucleotide sequence of each gene in a genome, one (in principle) also knows the network connectivity without further information. This situation is similar to a software assembly-language program: “Line 3: Goto Line 5” where both the input and output addresses are fixed, and the flow graph is determined by the instructions in the program. However, a further difference is that the output of a gene is not the “address” of another gene: it is a protein that can bind with varying strength to a number of other genes.

The state of a gene machine is the concentrations of the transcription factors produced by each gene (or arriving from the environment). The operations, again, are the input-output functions of each gene. But what is the “execution” of a gene machine? It is not as simple as saying that one gene stimulates or inhibits another gene. It is known that certain genes perform complex computations on their inputs that are a mixture of boolean, analog, and multi-stage operators (Figure 7-B [54]). Therefore, the input region of each gene can itself be a sophisticated machine.

Whether the execution of a gene machine should be seen as a continuous or discrete process, both in time and in concentration levels, is already a major question. Qualitative models (e.g.: random and probabilistic Boolean networks [26][50], asynchronous automata [52], network motifs [36]) can provide more insights than quantitative models, whose parameters are hard to come by and are possibly not critical. On the other hand, it is understood that pure Boolean models are inadequate in virtually all real situations. Continuous, stochastic, and decay aspect of transcription factor concentrations are all critical in certain situations [34][53].

5.2 Notations

Despite all these difficulties and uncertainties, a single notation for the gene machine is in common use, which is the gene network notation of Figure 7-A. There, the gates are connected by either “excitatory” (pointed arrow) or “inhibitory” (blunt arrow) links. What such relationships might mean is often left unspecified, except that, in a common model, a single constant weight is attached to each link.

Any serious publication would actually start from a set of ordinary differential

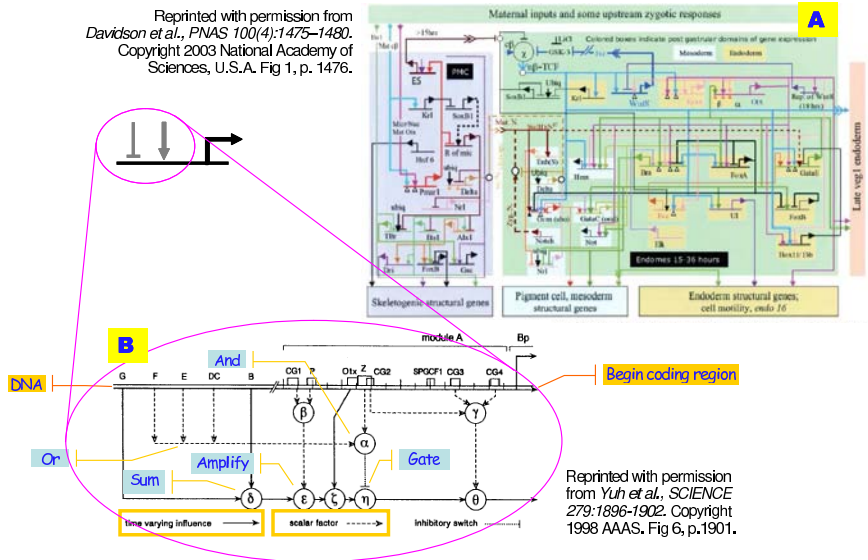
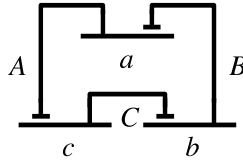


Figure 7: **Gene Regulatory Networks Notation.** **A** [16]: gene regulatory network involved in sea urchin embryo development: **B** [54]: boolean/arithmetic diagram of module A, the last of 6 interlinked modules in the regulatory region of the *endo16* sea urchin gene; G,F,E,DC,B are module outputs feeding into A, the whole region is 2300 base pairs.

equations relating concentrations of transcription factors, and use pictures such as at Figure 7-A only for illustration, but this approach is only feasible for small networks. The best way to formalize the *notation* of gene regulatory networks is still subject to debate and many variations, but there is little doubt that formalizing such a notation will be essential to get a grasp on gene machines the size of genomes (the smallest of which, *M.Genitalium*, is on the order of 150 Kilobytes, and one closer to human cellular organization, *Yeast*, is 3 Megabytes).

5.3 Example: Repressilator

The circuit in Figure 8, artificially engineered in *E.Coli* bacteria [19], is a simple oscillator (given appropriate parameters). It is composed of three genes with single input that inhibit each other in turn. The circuit gets started by *constitutive transcription*: each gene autonomously produces output in absence of inhibition, and the produced output decays at a certain stochastic rate. The symmetry of the circuit is broken by the underlying stochastic behavior of chemical reactions. Its

Figure 8: **Repressilator Circuit**

behavior can be understood as follows. Assume that gene a is at some point not inhibited (i.e. the product B of gene b is absent). Then gene a produces A , which shuts down gene c . Since gene c is no longer producing C , gene b eventually starts producing B , which shuts down gene a . And so on.

5.4 Summary

The fundamental flavor of the Gene Machine is: *slow asynchronous stochastic broadcast*. The interaction model is really quite strange, by computing standards. Each gene has a fixed output, which is not quite an address for another gene: it may bind to a large number of other genes, and to multiple locations on each gene. The transcription factor is produced in great quantities, usually with a well-specified time-to-live, and needs to reach a certain threshold to have an effect. On the other hand, various mechanisms can guarantee Boolean-like switching when the threshold is crossed, or, very importantly, when a message is *not* received. Activation of one gene by another gene is slow by any standard: typically one to five minutes, to build up the necessary concentration¹. However, the genome can slowly direct the assembly-on-need of protein machines that then act fast: this “swap time” is seen in experiments that switch available nutrients. The stochastic aspect is fundamental because, e.g., with the same parameters, a circuit may oscillate under stochastic/discrete semantics, but not under deterministic/continuous semantics [53]. One reason is that a stochastic system may decay to *zero* molecules of a certain kind at a given time, and this can cause switching behavior, while a continuous system may asymptotically decay only to a non-zero level.

¹Consider that bacteria replicate in only 20 minutes while cyclically activating hundreds of genes. It seems that, at least for bacteria, the gene machine can make “wide” but not very “deep” computations [36].

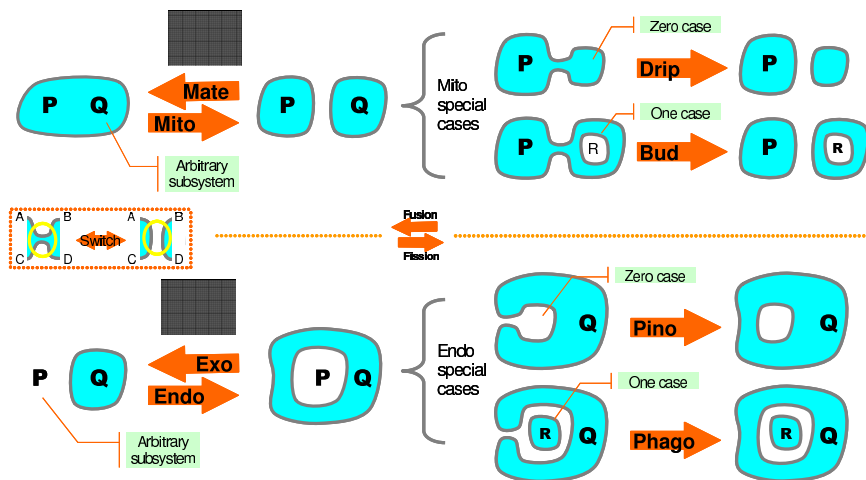


Figure 9: The Membrane Machine Instruction Set (2D)

6 The Membrane Machine (Transport Networks)

6.1 Principles of Operation

A cellular membrane is an oriented closed surface that performs various molecular functions. Membranes are not just containers: they are coordinators and sites of major activity² Large functional molecules (proteins) are embedded in membranes with consistent orientation, and can act on both sides of the membrane simultaneously. Freely floating molecules interact with membrane proteins, and can be sensed, manipulated, and pushed across by active molecular channels. Membranes come in different kinds, distinguished mostly by the proteins embedded in them, and typically consume energy to perform their functions. The consistent orientation of membrane proteins induces an orientation on the membrane.

One of the most remarkable properties of biological membranes is that they form a two-dimensional fluid (a lipid bilayer) embedded in a three-dimensional fluid (water). That is, both the structural components and the embedded proteins freely diffuse on the two-dimensional plane of the membrane (unless they are held together by specific mechanisms). Moreover, membranes float in water, which may contain other molecules that freely diffuse in that three-dimensional fluid. Membrane themselves are impermeable to most substances, such as water and protons, so that they partition the three-dimensional fluid. This organization

²“For a cell to function properly, each of its numerous proteins must be localized to the correct cellular membrane or aqueous compartment.” [32] p.675.

provides a remarkable combination of freedom and structure.

Many membranes are highly dynamic: they constantly shift, merge, break apart, and are replenished. But the transformations that they support are naturally limited, partially because membranes must preserve their proper orientation, and partially because membrane transformations need to be locally-initiated and continuous. For example, it is possible for a membrane to gradually buckle and create a bubble that then detaches, or for such a bubble to merge back with a membrane. But it is not possible for a bubble to “jump across” a membrane (only small molecules can do that), or for a membrane to turn itself inside-out.

The basic operations on membranes, implemented by a variety of molecular mechanisms, are *local fusion* (two patches merging) and *local fission* (one patch splitting in two) [8]. We discuss first the 2D case, which is instructive and for which there are some formal notations, and then the 3D case, the real one for which there are no formal notations.

In two dimensions (Figure 9), at the local scale of membrane patches, fusion and fission become indistinguishable as a single operation, *switch*, that takes two membrane patches, i.e. to segments A-B and C-D, and switches their connecting segments into A-C and B-D (crossing is not allowed). We may say that, in 2D, a switch is a fusion when it decreases the number of whole membranes, and is a fission when it increases such number.

When seen on the global scale of whole 2D membranes, switch induces four operations: in addition to the obvious splitting (Mito) and merging (Mate) of membranes, there are also operation, quite common in reality, that cause a membrane to “eat” (Endo) or “spit” (Exo) another subsystem (P). There are common special cases of Mito and Endo, when the subsystem P consists of zero (Drip, Pino) or one (Bud, Phago) membranes. All these operations *preserve bitonality* (dual coloring); that is, if a subsystem P is on a dark (or light) background before a reaction, it will be on a dark (or light) background after the reaction. Bitonality is related to preservation of membrane orientation, and to locality of operations (a membrane jumping across another one does not preserve bitonality). Bitonal operations ensure that what is or was *outside* the cell (light) never gets mixed with what is *inside* (dark). The main reactions that violate bitonality are destructive and non-local ones (such a digestion, not shown). Note that Mito/Mate preserve the nesting depth of subsystems, and hence they cannot encode Endo/Exo; instead, Endo/Exo can encode Mito/Mate [12].

In three dimensions, the situation is more complex (Figure 10). There are 2 distinct local operations on surface patches, inducing 8 distinct global operations that change surface topology. *Fusion* joins two Positively curved patches (in the shapes of domes) into one Negatively curved patch (in the shape of a hyperbolic cooling tower) by allowing the P-patches to kiss and merge. *Fission* instead splits one N-patch into two P-patches by pinching the N-patch. Fusion does not neces-

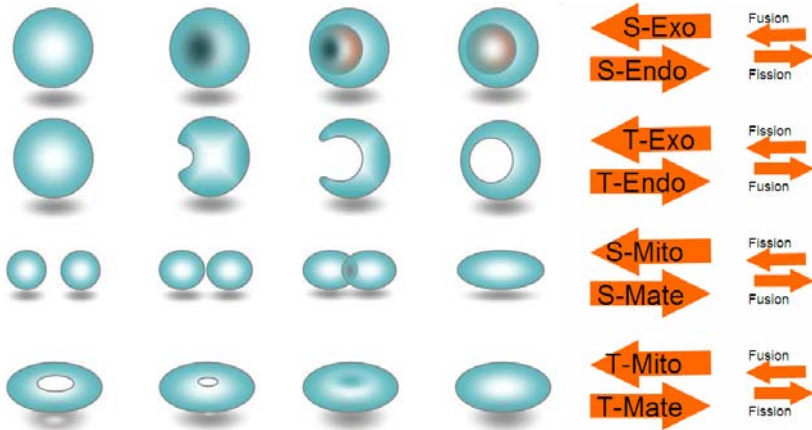


Figure 10: **The Membrane Machine Instruction Set (3D)**. Each row consists of initial state, two intermediate states, and final state (and back).

sarily decrease the number of membranes in 3D (it may turn a sphere into a torus in two different ways: T-Endo T-Mito), and Fission does not necessarily increase the number of membranes (it may turn a torus into a sphere in two different ways: T-Exo, T-Mate). In addition, Fusion may merge two spheres into one sphere in two different ways (S-Exo, S-Mate), and Fission may split one sphere into two spheres in two different ways (S-Endo, S-Mito). Note that S-Endo and T-Endo have a common 2D cross section (Endo), and similarly for the other three pairs.

Cellular structures have very interesting dynamic topologies: the eukaryotic nuclear membrane, for example, is two nested spheres connected by multiple toroidal holes (and also connected externally to the Endoplasmic Reticulum). This whole structure is disassembled, duplicated, and reassembled during cellular mitosis. Developmental processes based on cellular differentiation are also within the realm of the Membrane Machine, although geometry, in addition to topology, is an important factor there.

6.2 Notations

The informal notation used to describe executions of the Membrane Machine does not really have a name, but can be seen in countless illustrations (e.g., Figure 11, [32] p.730). All the stages of a whole process are summarized in a single snapshot, with arrows denoting operations (Endo/Exo etc.) that cause transitions between states. This kind of depiction is natural because often all the stages of a process *are* observed at once, in photographs, and much of the investigation has to do with

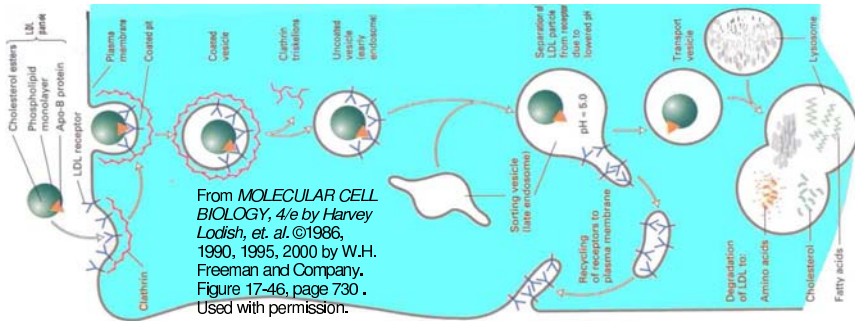


Figure 11: **Transport Networks Notation.** LDL particle (left) is recognized, ingested, and transported to a lysosome vesicle (right). [32], p.730.

determining their proper sequence and underlying mechanisms. These pictures are usually drawn in two colors, which is a hint of the semantic invariant we call bitonality.

Some membrane-driven processes are semi-regular, and tend to return to something resembling a previous configuration, but they are also stochastic, so no static picture or finite-state-automata notation can tell the real story. Complex membrane dynamics can be found in the protein secretion pathway, through the Golgi system, and in many developmental processes. Here too there is a need for a precise dynamic notation that goes beyond static pictures; currently, there are only a few such notations [42][48][12].

6.3 Example: LDL Cholesterol Degradation

The membrane machine runs real algorithms: Figure 11 depicts LDL-cholesterol degradation. The “problem” this algorithm solves is to transport a large object (an LDL particle) to an interior compartment where it can be degraded; the particle is too big to just cross the membrane. The “solution”, by a precise sequence of discrete steps and iterations, utilizes proteins embedded in the external cellular membrane and in the cytosol to recognize, bind, incorporate, and transport the particle inside vesicles to the desired compartment, all along recycling the active proteins.

6.4 Summary

The fundamental flavor of the Membrane Machine is: *fluid-in-fluid architecture, membranes with embedded active elements, and fusion and fission of compart-*

ments preserving bitonality. Although dynamic compartments are common in computing, operations such as endocytosis and exocytosis have never explicitly been suggested as fundamental. They embody important invariants that help segregate cellular materials from environmental materials. The distinction between active elements *embedded* on the surface of a compartment, vs. active elements *contained* in the compartment, becomes crucial with operations such as Exo. In the former case, the active elements are retained, while in the latter case they are lost to the environment.

7 Three Machines, One System

7.1 Principles of Operation

We have discussed how three classes of chemicals, among others, are fundamental to cellular functioning: nucleotides (nucleic acids), amino acids (proteins), and phospholipids (membranes). Each of our abstract machines is based primarily on one of these classes of chemicals: amino acids for the protein machine, nucleotides for the gene machine, and phospholipids for the membrane machine.

These three classes of chemicals are however heavily interlinked and interdependent. The gene machine “executes” DNA to produce proteins, but some of those proteins, which have their own dynamics, are then used as control elements of DNA transcription. Membranes are fundamentally sheets of pure phospholipids, but in living cells they are heavily doped with embedded proteins which modulate membrane shape and function. Some protein translation happens only through membranes, with the RNA input on one side, and the protein output on the other side or threaded into the membrane.

Therefore, the abstract machines are interlinked as well, as illustrated in Figure 2. Ultimately, we will need a single notation in which to describe all three machines (and more), so that a whole organism can be described.

7.2 Notations

What could a single notation for all three machines (and more) look like? All formal notations known to computing, from Petri Nets to term-rewriting systems, have already been used to represent aspects of biological systems; we shall not even attempt a review here. But none, we claim, has shown the breadth of applicability and scalability of process calculi [35], partially because they are not a single notation, but a coherent conceptual framework in which one can derive suitable notations. There is also a general theory and notation for such calculi [37], which can be seen as the formal umbrella under which to unify different abstract

machines.

Major progress in using process calculi for describing biological systems was achieved in Aviv Regev's Ph.D. thesis [47], where it is argued that one of the standard existing process calculi, π -calculus, enriched with a stochastic semantics [24][43][44], is extraordinarily suitable for describing both molecular-level interactions and higher levels of organization. The same stochastic calculus is now being used to describe genetic networks [30]. For membrane interactions, though, we need something beyond basic process calculi, which have no notion of compartments. Ambient Calculus [13] (which extends π -calculus with compartments) has been adapted [47][48] to represent biological compartments and complexes. A more recent attempt, Brane Calculus [12], embeds the biological invariants and 2D operations from Section 6.

These experiences point at process calculi as, at least, one of the most promising notational frameworks for unifying different aspects of biological representation. In addition, the process calculus framework is generally suitable for relating different levels of abstractions, which is going to be essential for feasibly representing biological systems of high architectural complexity.

Figure 12 gives a hint of the difference in notational approach between process calculi and more standard notations. Ordinary chemical reaction notation is a process calculus: it is a calculus of chemical processes. But it is a notation that focuses on reactions instead of components; this becomes a disadvantage when components have rich structure and a large state space (like proteins). In chemical notation one describes each state of a component as a different chemical species (Na, Na⁺), leading to an combinatorial blowup in the *description* of the system (the blowup carries over to related descriptions in terms of differential equations). In process calculus notation, instead, the components are described separately, and the reactions (occurring through complementary event pairs such as !r and ?r) come from the interactions of the components. Interaction leads to a combinatorial blowup in the *dynamics* of interactions, but not in the description of the systems, just like in ordinary object-oriented programming

On the left of Figure 12 we have a chemical description of a simple system of reactions, with a related (non-compositional) Petri Nets description. On the right we have a process calculus description of the same system, with a related (compositional) description in terms of interacting automata (e.g., Statecharts [22] with sync pseudostates). Both kinds of descriptions can take into account stochastic reaction rates (k_1, k_2), and both can be mapped to the same stochastic model (Continuous-Time Markov Chains), but the descriptions themselves have different structural properties. From a simulation point of view, the left-hand-side approach leads to large sparse matrices of chemical species vs. chemical reactions, while the right-hand-side approach leads to large multisets of interacting objects.

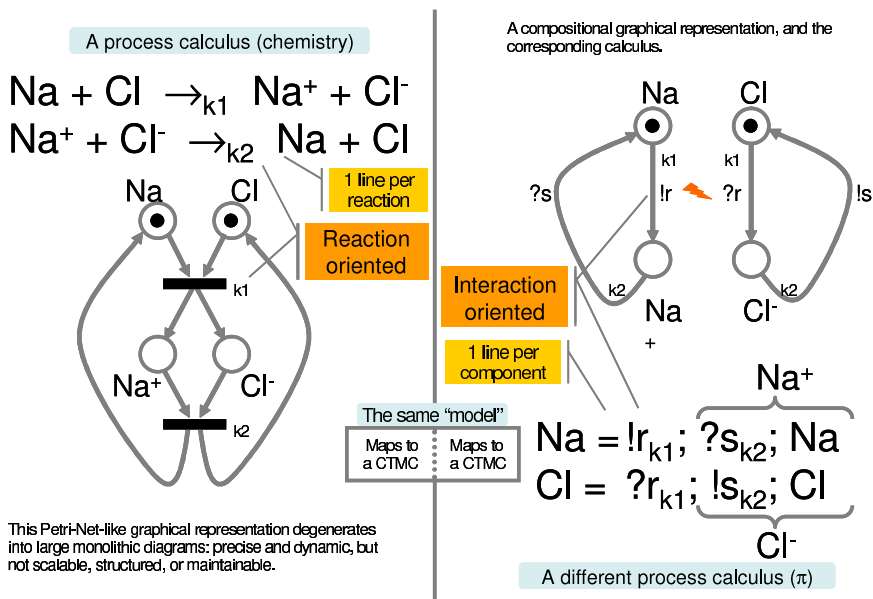


Figure 12: **Chemical vs. Process Calculi Notations**

7.3 Example: Viral Infection

The example in Figure 13 (adapted from [2], p.279) is the “algorithm” that a specific virus, the Semliki Forest virus, follows to replicate itself. It is a sequence of steps that involve the dynamic merging and splitting of compartments, the transport of materials, the operation of several proteins, and the interpretation of genetic information. The algorithm is informally described in English below. A concise description in Brane Calculus is presented in [12], which encodes the infection process at high granularity, but *in its entirety*, including the membrane, protein, and gene aspects.

A virus is too big to cross a cellular membrane. It can either punch its RNA through the membrane or, as in this example, it can enter a cell by utilizing standard cellular phagocytosis machinery. The virus consists of a capsid containing the viral RNA (the nucleocapsid). The nucleocapsid is surrounded by a membrane that is similar to the cellular membrane (in fact, it is obtained from it “on the way out”). This membrane is however enriched with a special protein that plays a crucial trick on the cellular machinery, as we shall see shortly.

Infection: The virus is brought into the cell by phagocytosis, wrapped in an additional membrane layer; this is part of a standard transport pathway into the

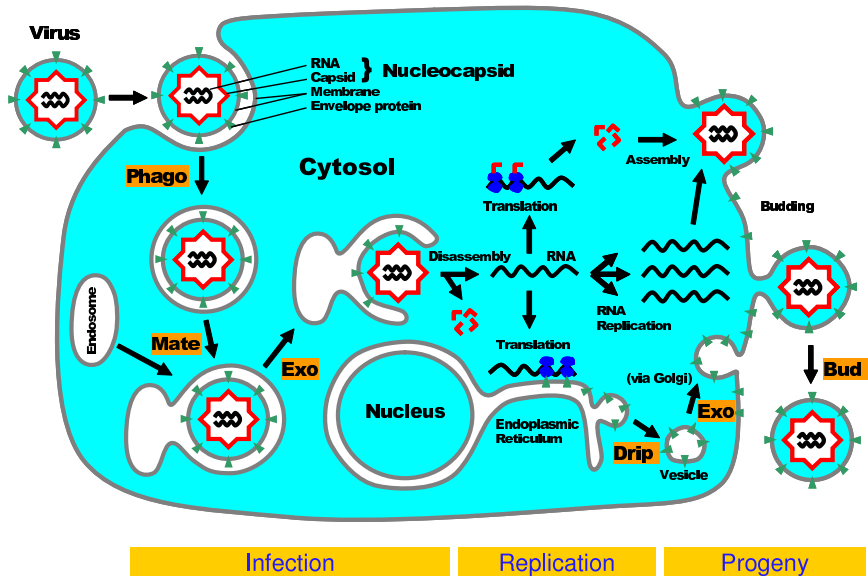


Figure 13: Viral Replication

cell. As part of that pathway, an endosome merges with the wrapped-up virus. At this point, usually, the endosome causes some reaction to happen in the material brought into the cell. In this case, though, the virus uses its special membrane protein to trigger an exocytosis step that deposits the naked nucleocapsid into the cytosol. The careful separation of internal and external substances that the cell usually maintains has now been subverted.

Replication: The nucleocapsid is now in direct contact with the inner workings of the cell, and can begin doing damage. First, the nucleocapsid disassembles, depositing the viral RNA into the cytosol. This vRNA then follows three distinct paths. First it is replicated to provide the vRNA for more copies of the virus. The vRNA is also translated into proteins, by standard cellular machinery. The proteins forming the capsid are synthesized in the cytosol. The virus envelope protein is instead synthesized in the Endoplasmic Reticulum, and through various steps (through the Golgi apparatus) ends up lining transport vesicles that merge with the cellular membrane, along another standard transport pathway.

Progeny: In the cytosol, the capsid proteins self-assemble and incorporate copies of the vRNA to form new nucleocapsids. The newly assembled nucleocapsids make contact with sections of the cellular membrane that are now lined with the viral envelope protein, and bud out to recreate the initial virus structure outside the cell.

7.4 Summary

The fundamental flavor of cellular machinery is: *chemistry in the service of materials, energy, and information processing*. The processing of energy and materials (e.g., in metabolic pathways) need not be emphasized here, rather we emphasize the processing of information, which is equally vital for survival and evolution [1]. Information processing tasks are distributed through a number of interacting abstract machines with wildly different architectures and principles of operation.

8 Outlook: Model Construction and Validation

The biological systems we need to describe are massively concurrent, heterogeneous, and asynchronous: notoriously the hardest kinds of systems to cope with in programming. They have stochastic behavior and high resilience to drastic changes of environmental conditions. What organizational principles make these systems work reliably, and what conditions make them fail? These are the questions that computational modeling needs to answer.

There are two main aspects to modeling biological systems. *Model construction*, requires first an understanding of the principles of operation. This is what we have largely been discussing here: understanding the abstract machines of systems biology should lead us to formal notations that can be used to build (large, complex) biological models. But then there is *model validation*: a good scientific model has to be verified or falsified through postdiction and prediction. We briefly list different techniques that are useful for model validation, once a specific model has been written up in a specific precise notation.

Stochastic simulation of biochemical systems is a common technique, typically based on the physically well-characterized Gillespie algorithm [21], which originally was devised for reaction-oriented descriptions. The same algorithm can be used also for component-oriented (compositional) descriptions with a dynamically unbounded set of chemical species [44]. Stochastic simulation is particularly effective for systems with a relatively low number of interactions of any given kind, as is frequently the case in cellular-scale systems. It produces a single (high-likelihood) trace of the system for each run. It frequently reveals behavior that is difficult to anticipate, and that may not even correspond to continuous deterministic approximations [34]. It can be quantitatively compared with experiments.

Static analysis techniques of the kind common in programming can be applied to the description of biological systems [40]. Control-flow analysis and mobility analysis can reveal subsystems that cannot interact [7][41]. Causality analysis can reconstruct the familiar network diagrams from process description

[11]. Abstract interpretation can be used to study specific facets of a complex model [39], including probabilistic aspects [17].

Modelchecking is now used routinely in the analysis of hardware and software systems that have huge state spaces; it is based on the state and transition model we emphasized during the discussion of abstract machines. Modelchecking consists of a model description language for building models, a query language for asking questions about models (typically temporal logic), and an efficient state exploration engine. The basic technology is very advanced, and is beginning to be applied to descriptions of biological systems too, in various flavors. **Temporal** modelchecking asks qualitative questions such as whether the systems can reach a certain state (and how), or whether a state is a necessary checkpoint for reaching another state [9][20]. **Quantitative** modelchecking asks quantitative questions about, e.g., whether a certain concentration can eventually equal or double some other concentration in some state [4][6]. **Stochastic** modelchecking, based, e.g., on discrete or continuous-time Markov chain models, can ask questions about the probability of reaching a given state [31].

Formal reasoning is the most powerful and hardest technique to use, but already there is a long tradition of building tools for verifying properties of concurrent systems. Typical activities in this area are checking behavioral equivalence between different systems, or between different abstraction levels of the same system, including now biological systems [10][5].

While computational approaches to biology and other sciences are now common, several of the techniques outlined above are unique to computer science and virtually unknown in other fields; hopefully they will bring useful tools and perspectives to biology.

9 Conclusions

Many aspects of biological organization are more akin to discrete hardware and software systems than to continuous systems, both in hierarchical complexity and in algorithmic-like information-driven behavior. These aspects need to be reflected in the modeling approaches and in the notations used to describe such systems, in order to make sense of the rapidly accumulating experimental data.

“The data are accumulating and the computers are humming, what we are lacking are the words, the grammar and the syntax of a new language...”

Dennis Bray (TIBS 22(9):325-326, 1997)

“The most advanced tools for computer process description seem to be also the best tools for the description of biomolecular systems.”

Ehud Shapiro (Biomolecular Processes as Concurrent Computation, Lecture Notes, 2001)

“Although the road ahead is long and winding, it leads to a future where biology and medicine are transformed into precision engineering.”

Hiroaki Kitano (Nature 420:206-210, 2002)

“The problem of biology is not to stand aghast at the complexity but to conquer it.”

Sydney Brenner (Interview, Discover Vol. 25 No. 04, April 2004)

10 References

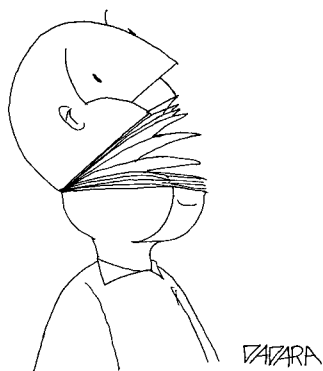
- [1] C.Adami. **What is complexity?** BioEssays 24:1085-1094, Wiley, 2002.
- [2] B.Alberts, D.Bray, J.Lewis, M.Raff, K.Roberts, J.D.Watson. **Molecular biology of the cell.** Third Edition, Garland.
- [3] R.Alur, C.Belta, F.Ivancic, V.Kumar, M.Mintz, G.J.Pappas, H.Rubin, and J.Schug,. **Hybrid modeling of biomolecular networks.** Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, Rome Italy, March 28-30, 2001. LNCS 2034.
- [4] M.Antoniotti, B.Mishra, F.Park, A.Policriti, N.Ugel. **Foundations of a query and simulation system for the modeling of biochemical and biological processes.** In L.Hunter, T.A.Jung, R.B.Altman, A.K.Dunker, T.E.Klein, editors, The Pacific Symposium on Biocomputing (PSB 2003) 116-127. World Scientific, 2003.
- [5] M.Antoniotti, C.Piazza, A.Policriti, M.Simeoni, B.Mishra. **Modeling cellular behavior with hybrid automata: bisimulation and collapsing.** In Int. Workshop on Computational Methods in Systems Biology (CMSB'03), LNCS. Springer, 2003. To appear.
- [6] M.Antoniotti, A.Policriti, N.Ugel, B.Mishra. **Model building and model checking for biochemical processes.** In Cell Biochemistry and Biophysics, 2003. In press.
- [7] C.Bodei, P.Degano, F.Nielson, H.R.Nielson. **Control flow analysis for the pi-calculus.** Proc. 9th International Conference on Concurrency Theory, LNCS 1466:84-98. Springer, 1998.
- [8] K.N.J. Burger. **Greasing membrane fusion and fission machineries.** Traffic 1:605-613. 2000.
- [9] G.Ciobanu, V.Ciubotariu, B.Tanasa. **A π -calculus model of the Na pump.** Genome Informatics 13:469-471, 2002.

- [10] G.Ciobanu. **Software verification of biomolecular systems**. In G.Ciobanu, G.Rozenberg (Eds.): *Modelling in Molecular Biology*, Natural Computing Series, Springer, 40-59, 2004.
- [11] M.Curti, P.Degano, C.Priami, C.T.Baldari. **Modelling biochemical pathways through enhanced pi-calculus**. *Theoretical Computer Science* 325(1):111-140.
- [12] L.Cardelli. **Brane calculi - Interactions of biological membranes**. *Proc. Computational Methods in Systems Biology 2004*. Springer. To appear.
- [13] L.Cardelli, A.D.Gordon. **Mobile ambients**. *Theoretical Computer Science, Special Issue on Coordination*, D. Le Métayer Editor. Vol 240/1, June 2000. pp 177-213.
- [14] V.Danos, M.Chiaverini. **A core modeling language for the working molecular biologist**. 2002.
- [15] V.Danos, C.Laneve. **Formal molecular biology**. *Theoretical Computer Science*, to Appear.
- [16] E.H.Davidson, D.R.McClay, L.Hood. **Regulatory gene networks and the properties of the developmental process**. *PNAS* 100(4):1475-1480, 2003.
- [17] A.Di Pierro, H.Wiklicky. **Probabilistic abstract interpretation and statistical testing**. *Proc. Second Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*. LNCS 2399:211-212. Springer, 2002.
- [18] S.Efroni, D.Harel and I.R.Cohen. **Reactive animation: realistic modeling of complex dynamic systems**. *IEEE Computer*, to appear, 2005.
- [19] M.B.Elowitz, S.Leibler. **A synthetic oscillatory network of transcriptional regulators**. *Nature* 403:335-338, 2000.
- [20] F.Fages, S.Soliman, N.Chabrier-Rivier. **Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM**. *J. Biological Physics and Chemistry* 4(2):64-73, 2004.
- [21] D.T.Gillespie. **Exact stochastic simulation of coupled chemical reactions**, *Journal of Physical Chemistry* 81:2340-2361. 1977.
- [22] D.Harel. **Statecharts: a visual formalism for complex systems**. *Science of Computer Programming* 8:231-274. North-Holland 1987.
- [23] L.H.Hartwell, J.J.Hopfield, S.Leibler, A.W.Murray. **From molecular to modular cell biology**. *Nature*. 1999 Dec 2;402(6761 Suppl):C47-52.
- [24] J. Hillston. **A compositional approach to performance modelling**. Cambridge University Press, 1996.
- [25] C.-Y.F.Huang, J.E.Ferrell Jr. **Ultrasensitivity in the mitogen-activated protein kinase cascade**. *PNAS* 93:10078-10083, 1996.
- [26] S.Kauffman, C.Peterson, B.Samuelsson, C.Troein. **Random Boolean network models and the yeast transcriptional network**. *PNAS* 100(25):14796-14799, 2003.

- [27] H.Kitano. **The standard graphical notation for biological networks.** The Sixth Workshop on Software Platforms for Systems Biology, 2002.
- [28] H.Kitano. **A graphical notation for biochemical networks.** BIOSILICO 1:169-176, 2003.
- [29] K.W.Kohn. **Molecular interaction map of the mammalian cell cycle control and DNA repair systems.** Molecular Biology of the Cell 10(8):2703-34, 1999.
- [30] C.Kuttler, J.Niehren, R.Blossey. **Gene regulation in the pi calculus: simulating cooperativity at the Lambda switch.** BioConcur 2004, ENTCS.
- [31] M.Kwiatkowska, G.Norman, D.Parker. **Probabilistic symbolic model checking with PRISM: a hybrid approach.** J. Software Tools for Technology Transfer (STTT), 6(2):128-142. Springer-Verlag, 2004.
- [32] H.Lodish, A.Berk, S.L.Zipursky, P.Matsudaira, D.Baltimore, J.Darnell. **Molecular cell biology.** Fourth Edition, Freeman, 2002.
- [33] J.S.Mattick. **The hidden genetic program of complex organisms.** Scientific American p.31-37, October 2004.
- [34] H.H.McAdams, A.Arkin. **It's a noisy business! Genetic regulation at the nanomolar scale.** Trends Genet. 1999 Feb;15(2):65-9.
- [35] R.Milner. **Communicating and mobile systems: the π -calculus.** Cambridge University Press, 1999.
- [36] R.Milo, S.Shen-Orr, S.Itzkovitz, N.Kashtan, D.Chklovskii, U.Alon. **Network motifs: simple building blocks of complex networks.** Science 298:824-827, 2002.
- [37] R.Milner. **Biographical reactive systems.** CONCUR 2001, Proc. 12th International Conference in Concurrency Theory, LNCS 2154:16-35, 2001.
- [38] M.Nagasaki, S.Onami, S.Miyano, H.Kitano: **Bio-calculus: its concept and molecular interaction.** Genome Informatics 10:133-143, 1999. PMID: 11072350.
- [39] F.Nielson, R.R.Hansen, H.R.Nielson. **Abstract interpretation of mobile ambients.** Science of Computer Programming, 47(2-3):145-175, 2003.
- [40] F.Nielson, H.R.Nielson, C.Priami, D.Rosa. **Static analysis for systems biology.** Proc. ACM Winter International Symposium on Information and Communication Technologies. Cancun 2004.
- [41] F.Nielson, H.R.Nielson, C.Priami, D.Rosa. **Control flow analysis for BioAmbients.** Proc. BioCONCUR 2003, to appear.
- [42] G.Paun. **Membrane computing.** Springer, 2002.
- [43] C.Priami. **The stochastic pi-calculus.** The Computer Journal 38: 578-589, 1995.
- [44] C.Priami, A.Regev, E.Shapiro, W.Silverman. **Application of a stochastic name-passing calculus to representation and simulation of molecular processes.** Information Processing Letters, 80:25-31, 2001.

- [45] P.Prusinkiewicz, M.Hammel, E.Mjolsness. **Animation of plant development**. Proceeding of SIGGRAPH 93. ACM Press, 351:360, 1993.
- [46] M.Ptashne. **Genetic switch: phage Lambda revisited**. Cold Spring Harbor Laboratory Press. 3rd edition, 2004.
- [47] A.Regev. **Computational systems biology: a calculus for biomolecular knowledge**. Ph.D. Thesis, Tel Aviv University, 2002.
- [48] A.Regev, E.M.Panina, W.Silverman, L.Cardelli, E.Shapiro. **BioAmbients: an abstraction for biological compartments**. Theoretical Computer Science, to Appear.
- [49] A.Regev, E.Shapiro. **Cells as computation**. Nature, 419:343, 2002.
- [50] I.Shmulevich, E.R.Dougherty, W.Zhang. **From Boolean to probabilistic Boolean networks as models of genetic regulatory networks**. Proceedings of the IEEE 90(11):1778-1792, 2002.
- [51] **Systems biology markup language**. <http://www.sbml.org>.
- [52] D.Thieffry, R.Thomas. **Qualitative analysis of gene networks**. Pacific Symposium on Biocomputing 1998:77-88. PMID: 9697173.
- [53] J.M.Vilar, H.Y.Kueh, N.Barkai, S.Leibler. **Mechanisms of noise-resistance in genetic oscillators**. PNAS, 99(9):5988-5992, 2002.
- [54] C-H.Yuh, H.Bolouri, E.H.Davidson. **Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene**. Science 279:1896-1902, 1998. www.sciencemag.org

TECHNICAL CONTRIBUTIONS



A SIMPLE COMPLETENESS PROOF FOR THE AXIOMATISATIONS OF WEAK BEHAVIOURAL EQUIVALENCES

Yuxin Deng
Shanghai Jiaotong University
yuxindeng@sjtu.edu.cn

Abstract

This paper presents a simple but uniform completeness proof for the axiomatisations of five weak behavioural equivalences: branching congruence, η -congruence, delay congruence, quasi-branching congruence, and weak congruence in the basic CCS without recursion. For the first three congruences, our result improves van Glabbeek and Weijland's completeness proof of using graph rewriting by a more direct proof that is fully equational. For quasi-branching congruence, our completeness result is, to the best of our knowledge, new.

1 Introduction

Strong bisimilarity, the widely used notion of equivalence for process algebra [7], provides a definition of equality that can capture similarities between processes without forcing them to be syntactically the same. The idea is to match any transition in one process with a transition, labelled by the same action, in the other process.

An important feature in process algebra is abstraction, which deems some of the actions in a process invisible or silent. Consequently, any consecutive execution of invisible transitions is not observable. It turns out that there exist many possibilities for extending strong bisimilarity with invisible transitions. The first extension is Milner's weak bisimilarity [7], which resembles strong bisimilarity, but allows arbitrary sequences of invisible τ -transitions to be inserted before or after an atomic transition. Van Glabbeek and Weijland have introduced branching bisimilarity [11] as an equivalence on processes that preserves the branching structure of processes. It distinguishes slightly more processes than weak

bisimilarity. Situated between the two equivalences are two incomparable bisimilarities: η -bisimilarity and delay bisimilarity [11]. Cherief [3] has characterised quasi-branching bisimilarity (which is slightly coarser than branching bisimilarity) as the coarsest equivalence that is preserved under refinement and finer than η -bisimilarity and delay bisimilarity.

Unlike strong bisimilarity, all the five weak notions of bisimilarity are not closed under the summation operator, but for each of them one can define a corresponding congruence relation in a standard way. In the framework of basic CCS, where processes are built from inaction, prefixing, and summation operators, both strong bisimilarity and weak congruence can be completely axiomatised in an elegant way. The completeness property for strong bisimilarity is easy to prove. For weak congruence (that is called observation congruence in [7]), the completeness proof is also not difficult, thanks to a result attributed to Hennessy in [7], called the *Hennessy Lemma* which says that if $P \approx Q$ then either $\tau.P \simeq Q$ or $P \simeq Q$ or $P \simeq \tau.Q$, for weak bisimilarity \approx and weak congruence \simeq . For all the weak notions of congruences mentioned above, except for quasi-branching congruence, complete axiomatisations are proposed in [11]. The completeness proof for branching congruence is achieved by using an advanced *graph rewriting technique* due to Bergstra and Klop [2]. The basic idea is to establish a graph rewriting system on finite process graphs, which is confluent and terminating. Then one proves that: (i) two normal forms of the graph rewriting system are bisimilar iff they are equal (i.e., isomorphic), (ii) every rewriting step in the system preserves bisimilarity, and (iii) every rewriting step corresponds to a proof step of the axiom system in question. The completeness proofs for delay, weak, and η -congruence are then derived from the proof for branching congruence in a uniform way. As far as finite processes are concerned, this technique seems a bit heavy and discouraging to non-experts in process algebra. One may wonder whether it is possible to give a simpler completeness proof which is uniform for all congruences but only involves equational reasoning similar to that in [7].

In this paper we give a positive answer to the above question and we are able to add quasi-branching congruence into the picture; we present a simple but uniform completeness proof that works for all the five weak notions of congruence. Our proof is very simple because it involves direct equational reasoning instead of graph rewriting. Our result is also very uniform in the sense that by mild modifications to the completeness proof for branching congruence we obtain the proofs for other four congruences. We also find a subtle difference between weak behavioural equivalences that are to some extent sensitive to the branching structure of processes (e.g. branching bisimilarity, quasi-branching bisimilarity and η -bisimilarity) and that are insensitive (e.g. delay bisimilarity and weak bisimilarity): the Hennessy Lemma is valid for delay bisimilarity and weak bisimilarity but not for branching bisimilarity, quasi-branching bisimilarity and η -bisimilarity.

Due to the difference, our proof schema for completeness deviates from that given in [7]: instead of using the Hennessy Lemma, we exploit a *Promotion Lemma* which says that if $P \approx Q$ then $\tau.P = \tau.Q$ is provably in some axiom system. The Promotion Lemma is less demanding than Hennessy Lemma and thus valid for all the five bisimilarities, based upon which we achieve a uniform proof.

This paper highlights the power of the Promotion Lemma because it shows another situation where the Hennessy Lemma fails but the Promotion Lemma leads to completeness. Similar phenomenon has already occurred in the π -calculus [6] and probabilistic process algebra [5], but none of the weak equivalences investigated in those papers is sensitive to the branching structure of processes.

2 Weak behavioural equivalences

Following [11], we consider a simple language, the basic CCS. But the result in this paper can be easily generalised (see the discussions in Section 5). Processes are built from inaction ($\mathbf{0}$), prefixing ($\alpha.P$), and summation ($P + Q$). The operational semantics of processes is standard. We write $P \Longrightarrow P'$ if there are processes P_0, \dots, P_n with $n \geq 0$ and $P \equiv P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} P_n \equiv P'$. If in that sequence $n \geq 1$ then we write $P \xRightarrow{\tau} P'$.

We recall the definitions of several weak notions of bisimulation appeared in the literature (see e.g. [7, 11]).

Definition 2.1. *A binary relation \mathcal{R} over processes is a branching simulation if PRQ implies that whenever $P \xrightarrow{\alpha} P'$ then*

(C_τ): either $\alpha = \tau$ and $P'\mathcal{R}Q$

(C_b): or there exist Q', Q'' such that $Q \Longrightarrow Q' \xrightarrow{\alpha} Q''$ with PRQ'' and $P'\mathcal{R}Q'$.

The relation \mathcal{R} is a branching bisimulation if both \mathcal{R} and \mathcal{R}^{-1} are branching simulations. Two processes P and Q are branching bisimilar, denoted $P \approx_b Q$, if there exists a branching bisimulation relating P and Q .

There are some variants of the above matching conditions:

(C_τ^s): either $\alpha = \tau$ and there exists Q' such that $Q \Longrightarrow Q'$ with PRQ' and $P'\mathcal{R}Q'$

(C_τ^q): either $\alpha = \tau$ and there exists Q' such that $Q \Longrightarrow Q'$ with $P'\mathcal{R}Q'$

(C_η): or there exist Q', Q'' such that $Q \Longrightarrow Q'' \xrightarrow{\alpha} Q'$ with PRQ'' and $P'\mathcal{R}Q'$

(C_d): or there exists Q' such that $Q \xRightarrow{\alpha} Q'$ with $P'\mathcal{R}Q'$

(C_w) : or there exists Q' such that $Q \xRightarrow{\alpha} \xRightarrow{\alpha} Q'$ with $P' \mathcal{R} Q'$

Semi-branching bisimilarity (\approx_s) is defined in terms of (C_τ^s) and (C_b) .

Quasi-branching bisimilarity (\approx_q) is defined in terms of (C_τ^q) and (C_b) .

η -bisimilarity (\approx_η) is defined in terms of (C_τ) and (C_η) .

Delay bisimilarity (\approx_d) is defined in terms of (C_τ) and (C_d) .

Weak bisimilarity (\approx_w) is defined in terms of (C_τ) and (C_w) .

It can be checked that all the bisimilarities defined above are indeed equivalence relations. In [11] it is shown that \approx_s coincides with \approx_b , the inclusions $\approx_b \subseteq \approx_q \subseteq \approx_\eta \subseteq \approx_w$ and $\approx_b \subseteq \approx_q \subseteq \approx_d \subseteq \approx_w$ are strict, but \approx_η and \approx_d are incomparable.

It turns out that all the bisimilarities defined above are not congruences with respect to the operator $+$. The classical counterexample is that $\tau.a \approx_x a$ but $\tau.a + b \not\approx_x a + b$ for $x \in \{b, q, \eta, d, w\}$. A typical way of obtaining congruences from bisimilarities is to require that both processes make essential moves at the first step [7, 11]. For instance, we define quasi-branching congruence as follows.

Definition 2.2. P and Q are quasi-branching congruent, written $P \simeq_q Q$, if

1. whenever $P \xrightarrow{\alpha} P'$ then
 - (a) either $\alpha = \tau$ and there exists Q' such that $Q \xRightarrow{\tau} Q'$ and $P' \approx_q Q'$,
 - (b) or there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \approx_q Q'$;
2. symmetric to clause 1 by exchanging the roles of P and Q .

The other four congruences can be defined in a similar way.

The next two lemmas report simple properties that hold for all the five bisimilarities studied in this paper. We shall exploit them to prove Lemma 3.3.

Lemma 2.3. For $x \in \{b, q, \eta, d, w\}$, if $\tau.P + Q \approx_x P$ then $P + Q \approx_x P$.

Proof. We make use of “bisimulation up to” techniques to construct appropriate bisimulations. See [4] for a detailed proof. \square

Lemma 2.4. For $x \in \{b, q, \eta, d, w\}$, if $P \approx_x Q$ then one of the three cases holds:

1. there exists some P' such that $P \xrightarrow{\tau} P'$ and $P' \approx_x Q$;
2. there exists some Q' such that $Q \xrightarrow{\tau} Q'$ and $P \approx_x Q'$;
3. $P \approx_x Q$.

Proof. See [4]. \square

For \approx_d and \approx_w , we have the following result, where the part on \approx_w is known as the original Hennessy Lemma in CCS.

Lemma 2.5. *For $x \in \{d, w\}$, $P \approx_x Q$ iff ($\tau.P \simeq_x Q$ or $P \simeq_x Q$ or $P \simeq_x \tau.Q$).*

Proof. For \approx_w , a proof is given in [7]. It can be adapted for \approx_d easily. \square

Remark 2.6. *The above property does not hold for \approx_b , \approx_q and \approx_η . For a counterexample, consider the two processes $\tau.(a + b) + a$ and $a + b$. Let $x \in \{b, q, \eta\}$, it is true that $\tau.(a + b) + a \approx_x a + b$. However,*

$$\begin{aligned} \tau.(\tau.(a + b) + a) &\not\approx_x a + b & (i) \\ \tau.(a + b) + a &\not\approx_x a + b & (ii) \\ \tau.(a + b) + a &\not\approx_x \tau.(a + b) & (iii) \end{aligned}$$

In (i) an action b from the right hand side cannot be matched up by any action from the left hand side of the inequality. Similar for (ii). In (iii) an action a from the left hand side cannot be matched up by any action from the right hand side.

3 Axiomatisations

In this section we consider complete axiomatisations of branching congruence, quasi-branching congruence, η -congruence, delay congruence, and weak congruence. For all the axiomatisations, the soundness properties are quite easy to show, thus we omit them. So we focus on the completeness properties and provide a uniform but simple completeness proof that works for the five congruences.

All the axioms that we need are displayed in Figure 1. It is shown in [7] that **S1-4** form a complete axiom system for strong bisimilarity. By adding the three τ -laws **T1-3**, Milner has obtained a complete axiom system for weak congruence. In [11] van Glabbeek and Weijland have established a complete axiomatisation of branching congruence by adding **B** to **S1-4**. Two other congruences \approx_η and \approx_d are also axiomatised in [11], just by adding **{B,T3}** and **T1-2**, respectively, to **S1-4**. The axiom **T3'** is the special case of **T3** when $\alpha = \tau$, and it is derivable from **T2** and **S4**. We shall show that \approx_q can be axiomatised by adding **{B,T3'}** to **S1-4**. We use the following abbreviations for the axiom systems of the five congruences.

$$\begin{aligned} \mathcal{A}_b &= \{\mathbf{S1-4}, \mathbf{B}\} \\ \mathcal{A}_q &= \{\mathbf{S1-4}, \mathbf{B}, \mathbf{T3'}\} \\ \mathcal{A}_\eta &= \{\mathbf{S1-4}, \mathbf{B}, \mathbf{T3}\} \\ \mathcal{A}_d &= \{\mathbf{S1-4}, \mathbf{T1-2}\} \\ \mathcal{A}_w &= \{\mathbf{S1-4}, \mathbf{T1-3}\} \end{aligned}$$

S1	$P + \mathbf{0} = P$
S2	$P + Q = Q + P$
S3	$P + (Q + R) = (P + Q) + R$
S4	$P + P = P$
B	$\alpha.(\tau.(P + Q) + Q) = \alpha.(P + Q)$
T1	$\alpha.\tau.P = \alpha.P$
T2	$\tau.P + P = \tau.P$
T3	$\alpha.(P + \tau.Q) + \alpha.Q = \alpha.(P + \tau.Q)$
T3'	$\tau.(P + \tau.Q) + \tau.Q = \tau.(P + \tau.Q)$

Table 1: All the axioms

Note that **B** implies **T1** and is derivable from **T1-2**. So we can also use **T1** when doing equational reasoning in $\mathcal{A}_b, \mathcal{A}_q, \mathcal{A}_\eta$, and use **B** in $\mathcal{A}_d, \mathcal{A}_w$. We write $\mathcal{A} \vdash P = Q$ if $P = Q$ can be derived from the equations in \mathcal{A} .

The following saturation properties, clauses 1 and 4 in particular, are well-known in CCS. Here we also consider two special cases of the transition relation $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$: $\xrightarrow{\alpha} \Longrightarrow$ and $\Longrightarrow \xrightarrow{\alpha}$, in clauses 2 and 3, respectively.

Lemma 3.1 (Saturation). 1. if $P \xRightarrow{\tau} P'$ then $\{\mathbf{S1-4}, \mathbf{T3'}\} \vdash P = P + \tau.P'$;

2. if $P \xrightarrow{\alpha} \Longrightarrow P'$ then $\{\mathbf{S1-4}, \mathbf{T3}\} \vdash P = P + \alpha.P'$;

3. if $P \Longrightarrow \xrightarrow{\alpha} P'$ then $\{\mathbf{S1-4}, \mathbf{T2}\} \vdash P = P + \alpha.P'$;

4. if $P \Longrightarrow \xrightarrow{\alpha} \Longrightarrow P'$ then $\{\mathbf{S1-4}, \mathbf{T2-3}\} \vdash P = P + \alpha.P'$.

Proof. By transition induction. As an example, we show the second clause.

Basis step: $P \xrightarrow{\alpha} P'$. Then the conclusion holds by **S4**.

Induction step: $P \xrightarrow{\alpha} \Longrightarrow P'' \xrightarrow{\tau} P'$. Then we infer

$$\begin{aligned}
 \{\mathbf{S1-4}, \mathbf{T3}\} \vdash P &= P + \alpha.P'' && \text{by induction} \\
 &= P + \alpha.(P'' + \tau.P') && \text{by S4} \\
 &= P + \alpha.(P'' + \tau.P') + \alpha.P' && \text{by T3} \\
 &= P + \alpha.P'
 \end{aligned}$$

□

As usual we use the notion of *normal form*. P is in normal form if it is of the form $\sum_{i=1}^n \alpha_i.P_i$, where each P_i is also in normal form.

Lemma 3.2. *For each P , there is a normal form P' such that $\{\mathbf{S1-4}\} \vdash P = P'$.*

We now introduce the important promotion lemma. It relates operational semantics to equational rewriting. Its proof is achieved by induction on the sizes of processes. We define the size, $size(P)$, of process P as follows.

$$\begin{aligned} size(\mathbf{0}) &= 0 \\ size(\alpha.P) &= 1 + size(P) \\ size(P + Q) &= size(P) + size(Q) \end{aligned}$$

Lemma 3.3 (Promotion). 1. *If $P \approx_b Q$ then $\mathcal{A}_b \vdash \tau.P = \tau.Q$;*

2. *If $P \approx_q Q$ then $\mathcal{A}_q \vdash \tau.P = \tau.Q$;*

3. *If $P \approx_\eta Q$ then $\mathcal{A}_\eta \vdash \tau.P = \tau.Q$;*

4. *If $P \approx_d Q$ then $\mathcal{A}_d \vdash \tau.P = \tau.Q$;*

5. *If $P \approx_w Q$ then $\mathcal{A}_w \vdash \tau.P = \tau.Q$.*

Proof. By Lemma 3.2, we can assume that P and Q are in normal form.

1. We carry out the proof by induction on $size(P + Q)$.

Basis step If $size(P + Q) = 0$, then it is straightforward to see that $\mathcal{A}_b \vdash P = Q = \mathbf{0}$, thus $\mathcal{A}_b \vdash \tau.P = \tau.Q$.

Induction step Suppose $size(P + Q) > 0$. Since $P \approx_b Q$, by Lemma 2.4 we can distinguish three cases.

(a) There exists some P' such that $P \xrightarrow{\tau} P'$ and $P' \approx_b Q$. To have the strong transition $P \xrightarrow{\tau} P'$, P must be of the form $\tau.P' + R$ for some process R . Since $\tau.P' + R \approx_b Q \approx_b P'$, it follows from Lemma 2.3 that $P' + R \approx_b P' \approx_b Q$. Note that $size(P' + R + Q) < size(\tau.P' + R + Q)$ and $size(P' + Q) < size(\tau.P' + R + Q)$. By induction hypothesis, it can be inferred that

$$\mathcal{A}_b \vdash \tau.(P' + R) = \tau.Q = \tau.P'. \quad (1)$$

So we derive

$$\begin{aligned} \mathcal{A}_b \vdash \tau.P &= \tau.(\tau.P' + R) \\ &= \tau.(\tau.(P' + R) + R) && \text{by (1)} \\ &= \tau.(P' + R) && \text{by B} \\ &= \tau.Q && \text{by (1)} \end{aligned}$$

- (b) There exists some Q' such that $Q \xrightarrow{\tau} Q'$ and $P \approx_b Q'$. This case is symmetric to case 1 by exchanging the roles of P and Q .
- (c) $P \approx_b Q$. We aim to prove that each summand of P can be absorbed by Q . Let $\alpha.P'$ be a summand of P , which gives rise to a transition $P \xrightarrow{\alpha} P'$. Correspondingly, there exists some Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \approx_b Q'$. Clearly we can derive

$$\mathcal{A}_b \vdash Q = Q + \alpha.Q' \quad (*)$$

by the axiom **S4**. Note that $\text{size}(P' + Q') < \text{size}(P + Q)$. So by induction hypothesis we obtain

$$\mathcal{A}_b \vdash \tau.P' = \tau.Q'. \quad (2)$$

So we derive

$$\begin{aligned} \mathcal{A}_b \vdash Q + \alpha.P' &= Q + \alpha.\tau.P' && \text{by T1} \\ &= Q + \alpha.\tau.Q' && \text{by (2)} \\ &= Q + \alpha.Q' && \text{by T1} \\ &= Q && \text{by (*)} \end{aligned}$$

In summary, $\mathcal{A}_b \vdash Q + P = Q$ and symmetrically $\mathcal{A}_b \vdash P + Q = P$. Therefore $\mathcal{A}_b \vdash \tau.P = \tau.(P + Q) = \tau.Q$.

2. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_q and \approx_q , and replace the two underlined parts with “either $Q \xrightarrow{\alpha} Q'$ or $\alpha = \tau$ and $Q \xrightarrow{\tau} Q'$ ” and “either the axiom **S4** or Lemma 3.1(1)” respectively.
3. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_η and \approx_η , and replace the two underlined parts with $Q \xrightarrow{\alpha} Q'$ and Lemma 3.1(2) respectively.
4. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_d and \approx_d , and replace the two underlined parts with $Q \xRightarrow{\alpha} Q'$ and Lemma 3.1(3) respectively.
5. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_w and \approx_w , and replace the two underlined parts with $Q \xRightarrow{\alpha} Q'$ and Lemma 3.1(4) respectively.

□

Remark 3.4. In the induction step of the above proof, we have distinguished three independent cases by Lemma 2.4, and in the first two cases Lemma 2.3 plays an important role. Nevertheless, for behavioural equivalences that are insensitive to the branching structure of processes such as \approx_d and \approx_w , the proof of the above lemma can be simplified. For instance, in the case of \approx_w , one just needs to consider two possibilities: (i) $P \xrightarrow{\tau} P'$ with $P' \approx_w Q$, (ii) $P \xrightarrow{\alpha} P'$ with $Q \Rightarrow \xrightarrow{\alpha} Q'$ and $P' \approx_w Q'$. In the particular case of (ii), one can prove the property (*) by Lemma 3.1(4). This proof schema is adopted in [6] to show the promotion property of all the behavioural equivalences considered in that paper. It is also adapted to a probabilistic setting in [5] where probabilistic weak bisimilarity is investigated. However, the proof schema does not apply to weak behavioural equivalences that are to some extent sensitive to the branching structure of processes, such as \approx_b , \approx_q and \approx_η . The reason is that for these equivalences the τ -transitions before the α -transition in (ii) cannot be simply absorbed. Otherwise, the branching structure of processes would not be observable.

With the saturation and promotion properties we are now ready to establish the following completeness theorem.

Theorem 3.5 (Completeness). 1. If $P \approx_b Q$ then $\mathcal{A}_b \vdash P = Q$;

2. If $P \approx_q Q$ then $\mathcal{A}_q \vdash P = Q$;

3. If $P \approx_\eta Q$ then $\mathcal{A}_\eta \vdash P = Q$;

4. If $P \approx_d Q$ then $\mathcal{A}_d \vdash P = Q$;

5. If $P \approx_w Q$ then $\mathcal{A}_w \vdash P = Q$.

Proof. 1. Similar to the proof Lemma 3.3(1) we assume P, Q in normal form and proceed by induction on $\text{size}(P+Q)$. The basis step is trivial, so we only consider the induction step. Let $\alpha.P'$ be a summand of P . Then $P \xrightarrow{\alpha} P'$ must be matched up by $\underline{Q} \xrightarrow{\alpha} Q'$ for some Q' such that $P' \approx_b Q'$. Clearly we can derive

$$\mathcal{A}_b \vdash Q = Q + \alpha.Q' \quad (**)$$

by the axiom **S4**. By Promotion Lemma,

$$\mathcal{A}_b \vdash \tau.P' = \tau.Q'. \quad (3)$$

Therefore

$$\begin{aligned} \mathcal{A}_b \vdash Q + \alpha.P' &= Q + \alpha.Q' && \text{by T1 and (3)} \\ &= Q && \text{by (**)} \end{aligned}$$

Hence we have $\mathcal{A}_b \vdash Q + P = Q$. Symmetrically $\mathcal{A}_b \vdash P + Q = P$. Therefore $\mathcal{A}_b \vdash P = Q$.

2. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_q and \approx_q , and replace the two underlined parts with “either $Q \xrightarrow{\alpha} Q'$ or $\alpha = \tau$ and $Q \xRightarrow{\tau} Q'$ ” and “either the axiom **S4** or Lemma 3.1(1)” respectively.
3. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_η and \approx_η , and replace the two underlined parts with $Q \xrightarrow{\alpha} \Rightarrow Q'$ and Lemma 3.1(2) respectively.
4. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_d and \approx_d , and replace the two underlined parts with $Q \Rightarrow \xrightarrow{\alpha} Q'$ and Lemma 3.1(3) respectively.
5. The arguments are the same as in the proof of clause 1 except that we change all the notations \mathcal{A}_b and \approx_b into \mathcal{A}_w and \approx_w , and replace the two underlined parts with $Q \Rightarrow \xrightarrow{\alpha} \Rightarrow Q'$ and Lemma 3.1(4) respectively.

□

Remark 3.6. For $x \in \{d, w\}$, there exists a even simpler completeness proof that does not rely on the Promotion Lemma. The reason is that Lemma 2.5 helps to lift $P \approx_x Q$ to either $\tau.P \approx_x Q$ or $P \approx_x Q$ or $P \approx_x \tau.Q$, thus allowing the induction hypothesis to apply when proving (3) in the last proof. For instance, this is the method adopted in [7] for showing that \mathcal{A}_w constitutes a complete axiom system for \approx_w . For $x \in \{b, q, \eta\}$, however, this method cannot be used because the Hennessy Lemma fails for them (cf. Remark 2.6).

4 Related work

Van Glabbeek pointed out to us some related work, notably the paper [1] which uses an approach very close to ours. In that paper, Aceto *et al.* gave complete axiomatisations for branching, delay, weak, and η -congruences in the basic CCS augmented with prefix iteration. A careful comparison between [1] and this paper shows the following similarities and differences.

1. The completeness results in [1] are a bit stronger than ours because they are valid for open terms in the basic CCS with prefix iteration. However, to make our ideas as net as possible, in this paper we only consider closed terms in the basic CCS.
2. The main proof strategy of [1] is similar to that adopted in [11], i.e., one shows a completeness result for branching congruence, from which the

completeness results for delay, weak, and η -congruences are derived. However, since one of our aims is to obtain uniformity, we have adopted the strategy of setting up a general proof schema that gives us a direct completeness proof for all the congruences as well as quasi-branching congruence.

3. In [1] the completeness proof for branching congruence relies on a key result which states that (with a change of notations)

$$\text{If } P \approx_b Q \text{ then } \mathcal{A}_b \vdash \alpha.P = \alpha.Q \text{ for all } \alpha \in A_\tau. \quad (\dagger)$$

This is very similar to our Promotion Lemma for branching bisimilarity (Lemma 3.3(1)), though the statement looks different. Moreover, in view of the axiom **T1**, the two results are essentially equivalent. The difference is that our statement appears more concise, and more importantly we have extended the result to all other congruences, which allows us to give direct (and mutually independent) proofs of completeness for all weak behavioural equivalences. In contrast, the completeness proofs for delay, weak, and η -congruences in [1] are indirect because they depend on the proof for branching congruence.

4. We treat delay, weak, and η -bisimilarity in a way different from [1]. We do saturation by means of some “preprocessing” jobs (Lemma 3.1) before showing the quite general Promotion Lemma, whereas Aceto *et al.* do it by means of some “postprocessing” jobs ([1] Proposition 4.7) that reduce the completeness proofs for delay, weak, and η -congruences to the proof for branching congruence.
5. We show the Promotion Lemma in a very simple way, but we use the property Lemma 2.3 whose proof is not trivial. So in fact we have transferred some complexity in proving the Promotion Lemma to the proof of Lemma 2.3. Nevertheless, [1] does not need a property like Lemma 2.3 in showing (\dagger) , at the price of a less concise proof.

The notion of “branching bisimulation up to” first appears in [12], which axiomatises branching congruence over processes with finite-state behaviours (so the axiomatisation of branching congruence over finite processes is treated as a special case). The completeness proofs use, in the same style as [8], a unique solution theorem.

We may consider Theorem 3.7 in [11] as the semantic version of our Promotion Lemma. It states that $P \approx_x Q$ iff $\tau.P \simeq_x \tau.Q$ for $x \in \{b, \eta, d, w\}$. It is also remarked in [11] that the Hennessy Lemma holds for delay and weak bisimilarity, but not for branching and η -bisimilarity. Nevertheless, in [11] the usefulness of the above theorem in completeness proofs is not investigated.

5 Concluding remarks

This paper presents a simple but uniform completeness proof for the axiomatisations of branching congruence, quasi-branching congruence, η -congruence, delay congruence, and weak congruence over finite processes. Compared with the completeness proof of [11] that employs a graph rewriting technique, our proof by direct equational reasoning is more accessible to non-experts in process algebra. This paper also shows a setting where the Hennessy Lemma fails but the Promotion Lemma leads to completeness.

For convenience of presentation, we have focused on the basic CCS where processes are built from the operators of inaction, prefixing, and summation. Other operators, for example, restriction, relabelling, and parallel composition, can be incorporated while preserving the proof schema presented in this paper. It is also straightforward to extend our results from closed terms to open terms. However, since our completeness proof proceeds by induction on the size of processes, we are not able to handle recursion, which requires other machinery (e.g., the unique solution theorem investigated in [8]).

For future work, it is interesting to extend our results to the π -calculus [9, 10]. In that setting, one has the freedom to consider late vs. early style of weak behavioural equivalences. For example, we could define a *late branching bisimulation* and an *early branching bisimulation*. Here is a potential candidate:

Definition. A symmetric binary relation \mathcal{R} is a late branching bisimulation if whenever PRQ and $P \xrightarrow{\alpha} P'$ then

1. either $\alpha = \tau$ and $P'\mathcal{R}Q$,
2. or the following property holds:
 - (a) if α is not an input action then some Q', Q'' exist such that $Q \Longrightarrow Q'' \xrightarrow{\alpha} Q'$ with PRQ'' and $P'\mathcal{R}Q'$;
 - (b) if $\alpha = a(x)$ then some Q', Q'' exist such that $Q \Longrightarrow Q'' \xrightarrow{\alpha} Q'$ with PRQ'' and $P'\{y/x\} \mathcal{R} Q'\{y/x\}$ for every name y .

To define early branching bisimulation, we just need to change clause 2(b) in the above definition into the following one:

if $\alpha = a(x)$ then for every name y some Q', Q'' exist such that $Q \Longrightarrow Q'' \xrightarrow{\alpha} Q'$ with PRQ'' and $P'\{y/x\} \mathcal{R} Q'\{y/x\}$.

It is easy to check that the largest late/early branching bisimulation is an equivalence relation. For the corresponding congruences and their axiomatisations, the results in this paper and [6] would be useful.

Acknowledgements We thank Rob J. van Glabbeek for helpful comments on some related work. We also acknowledge the support of the National Nature Science Foundation of China (60703033).

References

- [1] L. Aceto, R. J. van Glabbeek, W. Fokkink, and A. Ingólfssdóttir. Axiomatizing prefix iteration with silent steps. *Information and Computation*, 127(1):26–40, 1996.
- [2] J. A. Bergstra and J. W. Klop. A complete inference system for regular processes with silent moves. In *Proceedings of Logic Colloquium 1986*, pages 21–81. North Holland, Amsterdam, 1988.
- [3] F. Chierief. *Contributions à la sémantique du parallélisme: Bisimulations pour le raffinement et le vrai parallélisme*. PhD thesis, Institut National Polytechnique de Grenoble, 1992.
- [4] Y. Deng. A simple completeness proof for the axiomatisations of weak behavioural equivalences, 2007. Full version of the current paper. Available at <http://basics.sjtu.edu.cn/~yuxin/publications/branch.ps>.
- [5] Y. Deng and C. Palamidessi. Axiomatizations for probabilistic finite-state behaviors. *Theoretical Computer Science*, 373(1-2):92–114, 2007.
- [6] Y. Fu and Z. Yang. Tau laws for pi calculus. *Theoretical Computer Science*, 308:55–130, 2003.
- [7] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
- [8] R. Milner. A complete axiomatisation for observational congruence of finite-state behaviours. *Information and Computation*, 81:227–247, 1989.
- [9] R. Milner. *Communicating and Mobile Systems: the π -Calculus*. Cambridge University Press, 1999.
- [10] D. Sangiorgi and D. Walker. *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
- [11] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.
- [12] R. J. van Glabbeek. A complete axiomatization for branching bisimulation congruence of finite-state behaviours. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, volume 711 of *Lecture Notes in Computer Science*, pages 473–484. Springer, 1993.

THE DOMINO PROBLEM OF THE HYPERBOLIC PLANE IS UNDECIDABLE

Maurice Margenstern,

Université Paul Verlaine – Metz, LITA, EA 3097,
IUT de Metz, Île du Saulcy, 57045 METZ Cédex, FRANCE,
e-mail: margens@univ-metz.fr

Abstract

In this paper, we prove that the general tiling problem of the hyperbolic plane is undecidable by proving a slightly stronger version using only a regular polygon as the basic shape of the tiles. The problem was raised by a paper of Raphael Robinson in 1971, in his famous simplified proof that the general tiling problem is undecidable for the Euclidean plane, initially proved by Robert Berger in 1966.

1 Introduction

The question, whether it is possible to tile the plane with copies of a fixed finite set of tiles was raised by Wang, [24] in the late 50's of the previous century. Wang solved the *origin-constrained* problem which consists in fixing an initial tile in the above finite set of tiles. Indeed, fixing one tile is enough to entail the undecidability of the problem. The general case, later called the **general tiling problem** in this paper, *GTP* in short, without condition, in particular with no fixed initial tile, was proved undecidable by Berger in 1966, [1]. Both Wang's and Berger's proofs deal with the problem in the Euclidean plane. In 1971, Robinson found an alternative, simpler proof of the undecidability of *GTP* in the Euclidean plane, see [22]. In this 1971 paper, he raises the question of *GTP* for the hyperbolic plane. Seven years later, in 1978, he proved that in the hyperbolic plane, the origin-constrained problem is undecidable, see [23]. Up to now, *GTP* remained open.

In this paper, we give a synthetic presentation of the techniques contained on several technical papers deposited on *arXiv*, see [13, 18], and on the web site of the author, in particular [15].

In the second section, we remind a few features of all proofs of this problem. Then, we turn to the hyperbolic case, assuming that the reader is familiar with

hyperbolic geometry, at least with its popular models: Poincaré's disc or half-plane. See [20] and [10] for preliminaries and other bibliographical references.

In the third section, we sketchily present the frame of the construction. In the fourth section, we present a needed interlude, a parenthesis on brackets: it is a basic ingredient of the proof. In the fifth section, the line construction is lifted up to a planar Euclidean one. In the sixth section, we show how to implement the Euclidean construction in the hyperbolic plane. In the seventh section, we complete the proof of the main result:

Theorem 1. *The domino problem of the hyperbolic plane is undecidable.*

As an immediate corollary, *GTP* is undecidable in the hyperbolic plane.

In the eighth section, several corollaries of the construction are given. We give an indication on a possible simplification of the construction. We try to infer what might be learned from the construction leading to theorem 1.

Note that an alternative proof of *GTP* is outlined by Jarkko Kari in [6, 7]. His proof is completely different from this one. Here, the presented proof has the following property: given a one-bit oracle that the finite set of tiles S has a solution, our construction gives an algorithm to tile the plane with copies of tiles in S in infinite time.

2 The general strategy

In the proofs of *GTP* in the Euclidean plane by Berger and Robinson, an assumption, most probably considered as obvious at that time, is implicit.

Consider a finite set S of **prototiles**. Call **solution** of the tiling of the plane by S a partition \mathcal{P} such that the closure of any element of \mathcal{P} is a copy of an element of S . We notice that this definition contains the traditional condition on matching signs in the case when the elements of S possess signs.

Note that *GTP* can be formalized as follows:

$$\forall S \quad (\exists \mathcal{P} \text{ sol}(\mathcal{P}, S)) \vee \neg(\exists \mathcal{P} \text{ sol}(\mathcal{P}, S)),$$

where \vee is interpreted as: there is an algorithm which, applied to S provides us with 'yes' if there is a solution and 'no' if there is none.

The origin-constrained problem can be formalized in a similar way by:

$$\forall (S, a) \quad (\exists \mathcal{P} \text{ sol}(\mathcal{P}, S, a)) \vee \neg(\exists \mathcal{P} \text{ sol}(\mathcal{P}, S, a)),$$

where $a \in S$, with the same algorithmic interpretation of \vee .

Now, note that if we have a solution of *GTP*, we also have a solution of the origin-constrained problem, with the facility that we may choose the first tile. However, to prove that *GTP* has no solution, we have to prove that, whatever the initial tile, the corresponding origin-constrained problem also has no solution.

Nevertheless, Berger's and Robinson's proofs consider that we start the construction with a special tile, called the **origin**. There is no contradiction: they force the tiling to have a dense subset of origins. These origins start the simulation of the space-time diagram of the computation of the same Turing machine M . The machine M can be assumed to start from an empty tape. The origins define infinitely many domains of computation of infinitely many sizes. If the machine does not halt, starting from an origin, it is possible to tile the plane. If the machine halts, whatever the initial tile, we nearby find an origin and, from this one, we shall eventually fall into a domain which contains the halting of M : at this point, it is easy to prevent the tiling to go on.

The present construction aims at the same goal.

3 The general frame: the tiling $\{7, 3\}$ and its mantilla

Our construction takes place in a particular tiling of the hyperbolic plane: the tessellation $\{7, 3\}$ which we call the **ternary heptagrid**, simply **heptagrid**, for short, see [2, 11, 20]. It is generated by the regular heptagon with vertex angle $\frac{2\pi}{3}$ by reflection in its sides and, recursively, by reflection of the images in their sides. The background of figure 3 gives an illustration of this tiling in the Poincaré disc.

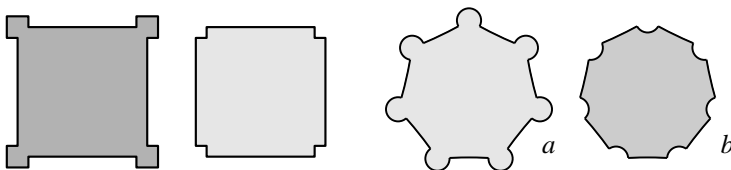


Figure 1. *On the left: Robinson's basic tiles for the undecidability of GTP in the Euclidean case. On the right: the tiles a and b are a 'literal' translation of Robinson's basic tiles to the situation of the heptagrid.*

We fix a special tiling based on the tessellation $\{7, 3\}$ which is motivated by the following consideration. In figure 1, the left-hand side indicates the basic shapes of the tiles devised by Robinson in the construction of the tiling used in his proof of the undecidability of GTP . The right-hand side of the figure gives the 'literal' translation of these tiles for the heptagrid. It is not difficult to see that it is not possible to tile the hyperbolic plane with the tiles a and b . However, a slight modification of the tile b , see the tile c of figure 2, leads to the solution. On the right-hand side of figure 2, we have the translation of the tiles a and c into genuine

à la Wang tiles. The pattern α corresponds to a and the pattern β corresponds to c .

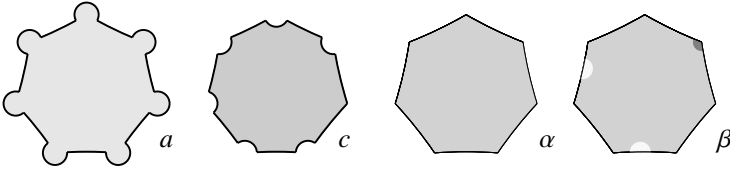


Figure 2. On the left: change in the tiles à la Robinson. On the right: their translation in pure Wang tiles.

3.1 The mantilla

In the ternary heptagrid, a **ball** of **radius** n around a tile T_0 is the set of tiles which are within distance n from T_0 which we call the **centre** of the ball. The **distance** of a tile T_0 to another T_1 is the number of tiles constituting a shortest path of adjacent tiles between T_0 and T_1 . Call **flower** a ball of radius 1.

Now, the tiles α and β of figure 2 can be assembled in flowers only: a tile α , which we call **centre**, requires to be surrounded by tiles β only, which we call **petals**. This is obtained by numbering the edges of the tiles α from 1 to 7, see [14, 15]. Now, a petal belongs to three flowers at the same time by the very definition of the implementation. And so, the flowers partially merge. The resulting tiling is called the **mantilla**.

There are several types of flowers, considering the number of red vertices for which the other end of an edge is a vertex of a centre. We refer the reader to [14] for the corresponding properties. Here, we simply take into consideration three basic patterns of flowers, which we call F -, G - and **8**-flowers respectively, see figure 3.

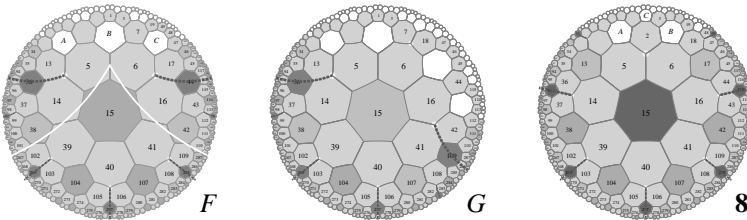


Figure 3. Splitting of the sectors defined by the flowers. From left to right: an F -sector, a G -sector and an **8**-sector.

The figure also represents the way which allows to algorithmically construct the mantilla. We split the **sectors** generated by each kind of flowers in sub-sectors

of the same kind and only them, which we call the **sons** of the flowers. This easily gives a way to recursively define a tiling. The construction is deterministic below the flower. It is non-deterministic when we proceed upwards. Later, we shall make the notions of top and bottom more precise. The exact description of the splitting can be found in [14]. We simply remark that such a splitting is an application of the general method described in [20, 10], for instance.

Based on these considerations, we have the following result which is thoroughly proved in [14].

Lemma 1. *There is a set of 4 tiles of type α and 17 tiles of type β which allows to tile the hyperbolic plane as a mantilla. Moreover, there is an algorithm to perform such a construction.*

3.2 Trees of the mantilla

The leftmost flower of figure 3, which represents an F -sector, also indicates a region delimited by continuous white lines. These lines are **mid-point** lines, which pass through the mid-points of consecutive edges of heptagons of the heptagrid. As shown in [2, 11], they delimit a Fibonacci tree. Let us remind that a Fibonacci tree has two kinds of nodes: black ones and white ones. A black node has two sons, a black and a white one. A white node has three sons, a black and two white ones. In both cases, the black son is the leftmost son. The root of a Fibonacci tree is a white node. The tiles inside the tree which are cut by these mid-point rays are called the **borders** of the tree, while the set of tiles spanned by the Fibonacci tree is called the **area** of the tree.

The F -son of a G -flower is called a **seed**. The tree, rooted at a seed, is called a **tree of the mantilla**. The seeds are the candidates for the construction of a computing region. From figure 3, we can easily define the **border** of a sector which is a ray crossing **8**-centres. See [14] for exact definitions.

Lemma 2. *(see [14]) The borders of a tree of the mantilla never meet the border of a sector. Consequently, the borders of two different trees of the mantilla never meet. Either their areas are disjoint or the area of one of them contains the area of the other.*

From this, we can order the trees of the mantilla by inclusion of their areas. It is only a partial order. We are interested in the maximal elements of this order. We call them **threads**, see [14] for an exact definition. Threads are indexed by N or Z . We call **ultra-threads** the threads which are indexed by Z . When there are ultra-threads, two of them coincide, starting from a certain index. The union of the areas of the trees which belong to an ultra-thread is the hyperbolic plane. Some realizations of the mantilla have ultra-threads, others have none.

3.3 Isoclines

In [15], we have a new ingredient. Define the status of a tile as **black** or **white**, applying the usual rules of such nodes in a Fibonacci tree. Then, we have:

Lemma 3. *It is possible to require that 8-centres are always black tiles. When this is the case, a seed is always a black tile.*

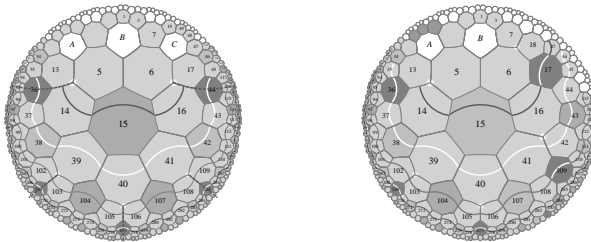


Figure 4. *The black tile property and the levels: On the left-hand side, a black F-centre; on the right-hand side, a black G_t -centre. We can see the case of an 8-centre on both figures.*

Define arcs as follows, see [15]: in a white tile, the arc joins the mid-point of the sides which have a common vertex with the side to the father. In a black tile, the arc joins the mid-point of the sides separated by the side to the father and the side to the uncle, on the left-hand side of the father. Joining arcs, we get paths. The maximal paths are called **isoclines**. They are illustrated on figure 4. An isocline is infinite and it splits the hyperbolic plane into two infinite parts. The isoclines from the different trees match, even when the areas are disjoint.

Lemma 4. *Let the root of a tree of the mantilla T be on the isocline 0. Then, there is a seed in the area of T on the isocline 5. If an 8-centre A is on the isocline 0, starting from the isocline 4, there are seeds on all the levels. From the isocline 10 there are seeds at a distance at most 20 from A .*

We number the isocline from 0 to 19 and repeat this, periodically. This defines **upwards** and **downwards** in the hyperbolic plane.

4 A parenthesis on brackets

We refer the reader to [15] for an exact definition. However, figure 5, below, illustrates the construction which now, we sketchily describe.

The generation 0 consists of points on a line which are regularly spaced. Label the points R, M, B, M , in this order. Repeat it periodically. An interval containing

R, M, B exactly is called **active**. An interval containing B, M, R exactly is called **silent**. The generation 0 is **blue**.

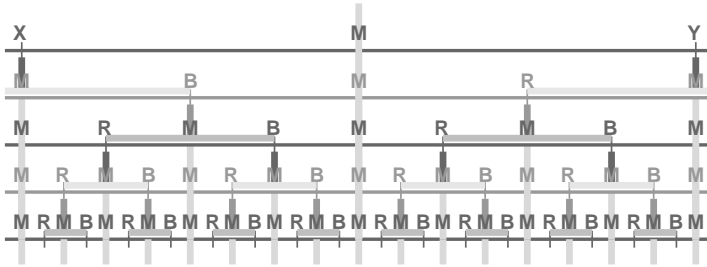


Figure 5. The silent and active intervals with respect to mid-point lines. The light vertical signals send the mid-point of the concerned interval to the next generation. The grays are chosen to be easily replaced by red or blue in an opposite way. The ends X and Y indicate that the figure can be used to study both active and silent intervals.

Blue and red are said **opposite**. Assume that the generation n is defined. For the generation $n+1$, the points which we take into consideration are the points which are still labelled M when the generation n is completed. Then, take at random an M which is the mid-point of an active interval of the generation n , and label it, either R or B . Next, define the active and silent intervals in the same way as for the generation 0. The colour of the active and silent intervals of the generation $n+1$ is opposite to that of the generation n .

When the process is achieved, we get an **infinite model**.

Deep results on the space of all these realizations are given by an accurate analysis to be found in [8]. The interested reader should have a look at this paper.

For our purpose, infinite models have interesting properties, see [15]. We shall mention some of them in the Euclidean implementation with triangles.

Cut an infinite model at some letter and remove all active intervals which contain this letter. What remains on the right-hand side of the letter is called a **semi-infinite model**.

It can be proved that in a semi-infinite model, any letter y is contained in at most finitely many active intervals, see [15].

5 The interwoven triangles

Now, we lift up the active intervals as *triangles* in the Euclidean plane. The triangles are isoceses and their heights are supported by the same line, called the **axis**, see figure 6. We also lift up silent intervals of the infinite model up to again isoceses triangles with their heights on the axis. To distinguish them from the others,

we call them **phantoms**. We shall speak of **trilaterals** for properties shared by both triangles and phantoms.

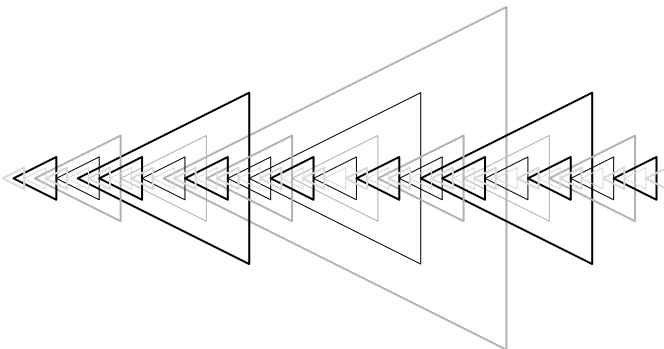


Figure 6. An illustration of the interwoven triangles.

We have very interesting properties for our purpose.

Lemma 5. *Triangles of the same colour do not meet no overlap: they are disjoint or embedded. Phantoms can be split into **towers** of embedded phantoms with the same mid-point and with alternating colours. Trilaterals can meet by a basis cutting the halves of the legs which contains the vertex.*

From the construction of the abstract brackets, we get a simple algorithm to construct the inteerwoven triangles.

The generation 0 is fixed by alternating triangles and phantoms. The mid-distance lines of the phantoms of generation 0 grow a horizontal green signal which crosses the legs of the phantom which they meet.

When the generation n is completed, a vertex of a triangle or a phantom is put on the intersection of the axis with the mid-distance line of a **triangle** of generation n . The legs grow until they meet a green signal, traveling on a horizontal line. The legs stop the green signal if the trilateral is a triangle, otherwise, they do not stop it. In both cases, the legs go on, until they meet a basis of their colour. They stop it and constitute a trilateral of the generation $n+1$. Now, the intersection of the basis of the trilateral with the axis requires a vertex: of a triangle, on a basis of phantom, of a phantom, on a basis of triangle. The process is repeated endlessly.

The detection of the green signal and of the correct basis are facilitated by the following mechanism: the legs of red triangles emit horizontal signals which have a laterality. Right-, left-hand side legs emit right-, left-hand side signals respectively. We do not provide a tile for the meeting of such signals inside a

triangle. This allows to detect what corresponds to the free letters: call them the **free rows** of a red triangle.

Lemma 6. *The interwoven triangles can be obtained by a tiling of the Euclidean plane which can be forced by a set of 190 tiles.*

In [15], we display the corresponding tiles which are in a square format, and we also describe them with the help of formulas taking into account the properties of lemma 5.

6 Implementing the interwoven triangles in the hyperbolic plane

The idea of the implementation in the hyperbolic plane is based on the following observation.

From lemma 4, we define the isoclines 0, 5, 10 and 15 to play the rôle of the rows in the Euclidean implementation. The trilaterals are constructed on trees of the mantilla. A vertex is a seed, and the legs are supported by the borders of the tree. The basis is defined by an isocline which cuts both borders of the tree.

As there are 6 seeds on the isocline 5 inside the tree defined by a seed at the isocline 0, there are 6 trilaterals of the generation 1 raised by a triangle of the generation 0. And so, contrarily to what happens in the Euclidean construction, we have several trilaterals of the same generation for the same set of isoclines crossed by the legs of these trilaterals.

Call **latitude** of a trilateral the set of isoclines which are crossed by its legs, vertex and basis being included.

It is not difficult to see that there will be infinitely many trilaterals within a given latitude. This requires to **synchronize** the choice of triangle or phantom when turning from one generation to the next one. Also, we can imagine that horizontal signals coming from different triangles of the same latitude and with different lateralities will meet. This will require to tune many details in order to maintain the guidelines of the algorithm, described in section 5.

The idea will be to synchronize the bases, the vertices and the signals on the mid-distance lines. We shall also have to see how the axis is implemented.

From what we have already seen, there is no more one axis, but a lot of them. In fact, what we called an axis can be materialized by a thread. As most threads are indexed by N only, we have always the implementation of a semi-infinite model. We shall manage the implementation in such a way that the semi-infinite models are simply different cuts of the same infinite model. The possibility of the realization of the infinite model in the case of an ultra-thread brings in no harm: it can be viewed as a cut at infinity.

6.1 The scent

By definition, we decide that all seeds which are on an isocline 0 are **active**. It means that they actually grow legs of a triangle of the generation 0. This is enough to guarantee that the set of active seeds is dense in the hyperbolic plane. Next, an active seed diffuses a **scent** inside its trilateral until the fifth isocline, starting from this seed, is reached. Seeds which receive the scent, and only them, become active. An active seed triggers the green signal when it reaches an isocline 5 or an isocline 15. By construction, the generation 0 is not determined by the meeting of a green signal. But the others are.

We can see that the scent process constructs a tree. The branches of the tree materialize the thread which implements the considered semi-infinite models. Note that the above synchronization mechanism fixes things for spaces between triangles but also inside them.

6.2 Horizontals signals with a laterality

Due to the occurrence of several trilaterals within a given latitude, now we have to require that all triangles, both red and blue and blue 0, emit a signal along their legs. The signal will be called **upper** when it is emitted by the legs, but there is an exception: the vertices do not emit any horizontal signal. There is another exception: the corners of a phantom, as well as those of a triangle, also emit an upper signal. The colour and the laterality of this signal are those of the corner.



Figure 7. *The pattern of a join-tile.*

In between two contiguous triangles of the same latitude, horizontal signals of the same colour but with a different laterality will meet. Such a meeting must be allowed: it is performed by an appropriate tile, called a **join** tile. There is a join-tile for red and blue signals, as well as for the orange signals, further defined. The pattern represented by figure 7 illustrates such a junction. On the left-, right-hand side, we have the right-, left- hand side signals respectively. We also require that an upper horizontal signal of a given laterality may cross a leg of a trilateral of the same colour, only if it has the same laterality as the leg. Note that the opposite junction cannot be obtained, as turning tiles is made impossible by the existence of the isoclines and their numbering.

Lemma 7. *An upper horizontal signal with a laterality cannot join two legs of the same tilateral.*

The proof easily follows from the rule about the meeting of legs with an upper horizontal signal. Accordingly, the legs of a trilateral may only be joined by a horizontal signal which has no laterality.

6.3 Synchronization: the mechanisms

The first mechanism which we introduce to force the synchronization of different constructions is that all bases on a given isocline merge. This changes the tile for the coner, but this does not affect the algorithm of section 5.

This principle forces the presence of various signals on the same isocline. Contradictions might follow, which would ruin the construction. We have to look at this very carefully.

As a first point, note that the upper signals allow to differentiate the various parts of a basis. If the upper horizontal signal is of the same colour as the basis it accompanies, we are outside any trilateral whose basis lies on the same isocline. Then, the basis is called **covered**. If not, we are inside a trilateral of this generation, and the basis is called **open**.

This distinction is important. When a leg meets a basis: if it is the first half of a leg, *i.e.* between the vertex and the mid-point of the leg, it meets the basis without changing it. When the second half of a leg meets a basis, it crosses it if the colour is different. If it is the same colour, it crosses it only if the basis is covered. If it is open, the leg has met the basis with which it forms a corner and it does not cross it. Indeed, from lemma 7, an upper horizontal signal cannot go from one leg of a trilateral to the other: there must be a triangle of the same colour in between.

Now, we have to look at the consequences of the synchronization on other signals: on the green signals and on the horizontal blue and red signals.

The problem is the following. Consider two triangles A and B of the same latitude. They belong to the same generation n and they have the same colour. We may assume that A is on the left-hand side of B . Let ι be the isocline of the mid-distance line of A and B . From the study of the abstract brackets, we know that there is a tower of phantoms inside A and inside B , such that ι is the mid-distance line of the phantoms of the tower. We also know that the same tower is repeated along ι between A and B , possibly several times. Accordingly, there is a green signal on ι inside A and inside B , and also between A and B .

A similar problem occurs with the horizontal signals which are emitted by the right-hand side leg of A , the vertex being excepted. Symmetrically, the left-hand side leg of B emits a horizontal signal of the same colour and on the same

isoclines of the latitude of A . The join-tile of figure 7, replicated in appropriate colours, allows to connect together horizontal signals of opposite lateralities travelling between A and B in the appropriate directions and on the same isocline. Now, the structure of inner trilaterals inside a phantom is the same as inside a triangle of the same generation. Accordingly, the notion of a free row also applies to phantoms. From lemma 7, if a horizontal signal enters a phantom of its colour on a free row, we get into trouble for the meeting with the other leg of the phantom.

Both problems can be solved in a similar way.

The idea is to **avoid** the phantoms as it is not possible to cross them. The advantage is that the deviated signal does not disturb the construction inside the phantom. Also, if during the detour, the signal keeps its laterality, it behaves as if the phantom were not present. In particular, the join-tile can be used for the connection of the signal with the opposite one coming in front of it, from the other triangle.

Technically, the solution is the following. The legs of a triangle stop the green signal which meet them on the mid-distance line, as required by the algorithm of section 5. Now, at the mid-point of the leg, on the corresponding isocline ι , the leg triggers an **orange** signal σ of its laterality, outside the triangle, say A . When σ meets the first phantom P on its way, it does not cross it: on the other side of the leg of P , there is a green signal. Instead of this, σ climbs up over the left-hand side leg of P until it reaches the vertex. Then, it goes down along the right-hand side leg of P until it reaches the isocline σ . This is easy to recognize: it is the single tile such that there is a green signal on the other side of the leg. Also, during this travel along the first half of the legs of P , σ does not change its laterality. From this, lemma 7 also applies to σ , which rules out the occurrence of an orange signal on ι inside P . And so, σ does not disturb the construction inside P . By lemma 7, we can see that σ matches its opposite signal on ι with the join-tile.

It is not difficult to see that the solution applies to a horizontal blue signal. For a blue-0 or blue phantom, there is a single free row: the mid-distance line. Accordingly, a horizontal blue signal meeting the leg of a blue-0 or blue phantom P on an isocline which is not the mid-distance line of P also meets the opposite signal coming from a blue-0 or blue triangle inside the phantom. The join-tile solves the problem if needed. For the blue signal which travels on the isocline ι of the mid-distance line of P , its behaviour is exactly that of an orange signal, which solves the problem.

We remain with the case of a red triangle. This time, ι still denotes the mid-distance line of the triangles A and B , and the right-hand side leg of A emits right-hand side horizontal red signals on all the isoclines of the latitude of A , the line of the vertex of A being excepted. We have the same situation as with a blue signal if the signal arrives on an isocline which does not correspond to a free row of the red phantom P which is first met on the way. If the signal arrives on an isocline

of a free row of P , the idea is to collect all the red signals travelling within the latitude of P on the isocline of a free row in a single signal, as there are a lot of free rows in a red trilateral. This single signal has the form of a red signal with a laterality: that of all the red signals arriving to P . It climbs up over the legs of P , this time from the vertex to the basis and conversely. Also, the laterality of the signal is not changed and, when it goes down along the other leg of P , the signal sends a copy of itself, outside P , on each isocline of a free row. In this way, the red signals which arrive to P and which depart from it on an isocline of a free row of P constitute a comb: on one side, the comb gathers the signals, and, on the other side, it dispatches them. In this way, the avoiding is obtained without perturbing what happens inside P , and also without disturbing what must be outside P . The signal passes as if P were not present. We just remark that, as the laterality is not changed, lemma 7 also applies here. As a consequence, the same phenomenon may happen inside P , but only within the latitudes of the triangles which P contains. As the mid-distance line of these triangles are different from that of P , the just described phenomenon occurs for the inner phantoms inside P whose mid-distance line is that of P . This is conformal with the requirement that what happens inside P must not be disturbed.

Now, we can conclude that the tiling forces the construction of trilaterals generation after generation, as indicated by the algorithm of section 5.

7 Completing the proof

7.1 The computing areas

Now, we ignore the blue-0 and the blue triangles, the phantoms of any colour as well as the parts of the bases of red triangles which are covered. Accordingly, we focus our attention on the red triangles only.

We have already mentioned that lemma 7 applied to horizontal red signals allows us to detect the free rows inside the red triangles. We shall agree that a special signal, the yellow one, without laterality, will mark the free rows of the red triangles. The set of tiles of lemma 6, see [13, 15], also implements this detection and the installation of the yellow signal on the free rows.

The free rows of the red triangles constitute the horizontals of the grid which we construct in order to simulate the space-time diagram of a Turing machine.

Now, we have to define the verticals of the grid to complete the simulation. They consist of rays which cross $\mathbf{8}$ -centres. Figure 8 illustrates the various ways of their connection with the tile of a border of the tree being on a free row.

The computing signal starts from a the seed. It travels on the free rows. Each time a vertical is met, which contains a symbol of the tape, the required instruction

is performed. If the direction is not changed and the corresponding border is not met, the signal goes on, on the same row. Otherwise, it goes down along the vertical until it meets the next free row. There, it looks at the expected vertical, going in the appropriate direction. Further details are dealt with in [15].

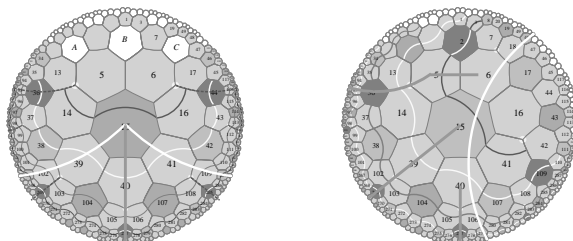


Figure 8. The perpendicular starting from a point of the border of a triangle which represents a square of the Turing tape.

On the left-hand side: the case of the vertex. On the right-hand side, the three other cases for the right-hand side border are displayed on the same figure.

Note that in the case of the butterfly model, see [13, 15], the mechanism of the orange signal, in particular for crossing legs of trilaterals, forces the green signal to run over the whole isocline which is the mid-point of the latitude which contains no triangle.

7.2 The tiles

Within the frame of this paper, it is not possible to exhaustively describe the tiles needed for the construction which we described. In the reports [15] and in [13], we give a precise account of the tiles.

Here, we just indicate how to construct the tiles, at the same time giving a way to describe all the tiles and to count them.

The finite set of tiles which we need to prove theorem 1, consists of two parts. First, we have the set of **prototiles** which forces the construction of the mantilla as well as the isoclines, the activation of seeds through the isoclines 0 and the spreading of the scent, and then the construction of the interwoven triangles, including the detection of the free rows in the red triangles.

The second set consists of **meta-tiles** which, in fact, are *variables* for tiles, as the meta-tiles convey the signals directly connected with the simulation of a given Turing machine. In the actual construction, the meta-tiles replace a part of the prototiles: they replace all the prototiles which are placed on an element of the computation: either the tiles which convey the computing signal, or those which convey the evolution of each square of the Turing tape. But the replacement is

not systematic: the precise interval of a free row where a computing signal occurs depends on the simulated machine.

In each set, the tiles are constituted of a tile α or β on which we superpose several signals. There are horizontal and vertical signals. The horizontal signals can be viewed as **channels** which run along the path of the isocline inside the tile. We know that each tile has a top part and a bottom one. By definition, the top is determined by the side to the father to which we assign the number 1. By numbering the other sides from 2 up to 7 by counter-clockwise turning around the tile, we define a **local numbering**. In this numbering, the isocline goes from the side 2 to the side 7 in a white tile and it goes from the side 3 to the side 7 in a black one. For upper signals, the channel is above the isoclines. A basis runs below the isocline. The green and orange signals run on the same channel which is above the isocline and below the channel for a horizontal red or blue signal.

We have two main kinds of verticals: the legs of a trilateral and the verticals for the computing grid. The signals of the legs always cross black tiles. Right-hand side legs go from the side 2 to the side 6 and left-hand side ones go from the side 1 to the side 4. From this, we notice that a tile of a leg always knows its laterality and, also, which of its sides are inside the trilateral and which are outside it. The vertical signals for the grid go through **8-centres**. In an **8-centre**, the signal goes from the side 2 to the side 5 in terms of the local numbering. In the next petal, it goes from the side 1 to the side 4. In the following tile, it goes from the side 1 to the side 6, from which it again enters an **8-centre**, through its side 2.

We have three colours for the trilaterals: blue-0 for the generation 0, blue for the even generations and red for the odd ones. Legs of triangles are represented by a **thick** signal, those of phantoms by a **thin** one. First halves of legs are dark and second ones are light in blue-0 and blue trilaterals. In red trilaterals, we use the opposite convention: the first half is light and the second one is dark. The tile of figure 7 defines the general patterns to indicate the laterality of a signal. These patterns are given the colour of the corresponding signal.

Then, the most intricate task comes: the superposition of all these kinds of signals. This can be precisely described and counted, as performed in [15, 13].

With this, we completed the proof of theorem 1.

8 A few corollaries

The construction leading to the proof of theorem 1 allows to get a few results in the same line of problems.

As indicated in [3, 4], there is a connection between *GTP* and the **Heesch number** of a tiling. This number is defined as the maximum number of **coronas** of a disc which can be formed with the tiles of a given set of tiles, see [9] for more

information. As indicated in [4], and as our construction fits in the case of domino tilings, we have the following corollary of theorem 1.

Theorem 2. *There is no computable function which bounds the Heesch number for the tilings of the hyperbolic plane.*

The construction of [12, 14] gives the following result, see [16, 19].

Theorem 3. *The finite tiling problem is undecidable for the hyperbolic plane.*

Combined with the construction proving theorem 3 and a result of [20], the construction of the present paper allows us to establish another result, see [17].

Theorem 4. *The periodic tiling problem is undecidable for the hyperbolic plane, also in its domino version.*

In this statement, **periodic** means that there is a shift which leaves the tiling globally invariant.

At last, in another direction, we may apply the arguments of Hanf and Myers, see [5, 21], and prove the following result, see [15].

Theorem 5. *There is a finite set of tiles such that it generates only non-recursive tilings of the hyperbolic plane.*

9 Conclusion

According to the estimations of [15, 18], the number of tiles is huge. Taking into account the changes introduced in [18], a new counting indicates that we need 18,856 prototiles and 4,006 meta-tiles. This will precisely be presented in a forthcoming paper.

Of course, we may wonder whether the number of tiles can be reduced. This might be possible by a small tuning of the present signals. For instance, we could forbid yellow rows on the mid-distance line of a red triangle. But the advantage would not be that important. Moreover, an attentive look at the tiles indicates that the reason of the big number of tiles lies in lemma 4. A consequence of the lemma is the important number of passive tiles connected with the isoclines which do not bear the construction signals. And so, a significant reduction of the number of tiles could be reached by a new, simpler setting. Professor Goodman-Strauss advised me to do so. In fact, recently, I realized that this is possible. I have not the room, here, to explain the idea. It is interesting to notice that this simplification allows to implement a similar implementation of the interwoven triangles in both the ternary heptagrid and in the pentagrid. But more than one third of the prototiles is needed for conveying and controlling the synchronization signals.

Another consequence could be derived from the construction. At first glance, lifting the abstract brackets to the interwoven triangles seems to be a purely Euclidean construction. However, it seems to me that the possibility to transfer the construction to the hyperbolic plane has an important meaning. From my humble point of view, this construction, which looks like purely Euclidean, is indeed within the scope of absolute geometry. It mainly requires Archimedes' axiom. Absolute geometry itself has no pure model: a realization is necessarily either Euclidean or hyperbolic. We suggest to conclude that, probably, the extent of absolute geometry is somehow under-estimated.

Acknowledgment

I am very pleased to acknowledge the interest of several colleagues and friends to the main result of this paper. Let me especially thank André Barbé, Jean-Paul Delahaye, Chaim Goodman-Strauss, Serge Grigorieff, Tero Harju, Oscar Ibarra, Hermann Maurer, Gheorghe Păun, Grzegorz Rozenberg, Vladimiro Sassone and Klaus Sutner.

References

- [1] Berger R., The undecidability of the domino problem, *Memoirs of the American Mathematical Society*, **66**, (1966), 1-72.
- [2] Chelghoum K., Margenstern M., Martin B., Pecci I., Cellular automata in the hyperbolic plane: proposal for a new environment, *Lecture Notes in Computer Sciences*, **3305**, (2004), 678-687, proceedings of ACRI'2004, Amsterdam, October, 25-27, 2004.
- [3] Goodman-Strauss, Ch., A strongly aperiodic set of tiles in the hyperbolic plane, *Inventiones Mathematicae*, **159**(1), (2005), 119-132.
- [4] Goodman-Strauss, Ch., Growth, aperiodicity and undecidability, *invited address at the AMS meeting at Davidson, NC*, March, 3-4, (2007).
- [5] Hanf W., Nonrecursive tilings of the plane. I. *Journal of Symbolic Logic*, **39**, (1974), 283-285.
- [6] Kari J., A new undecidability proof of the tiling problem: The tiling problem is undecidable both in the Euclidean and in the hyperbolic plane, *AMS meeting at Davidson, NC, Special Session on Computational and Combinatorial Aspects of Tilings and Substitutions*, March, 3-4, (2007).
- [7] Kari J., The Tiling Problem Revisited, *Lecture Notes in Computer Science*, **4664**, (2007). 72-79.

- [8] Levin L.A., Aperiodic Tilings: Breaking Translational Symmetry, *The Computer Journal*, **48**(6), (2005), 642-645.
- [9] Mann C., Heesch's tiling problem, *American Mathematical Monthly*, **111**(6), (2004), 509-517.
- [10] Margenstern M., Cellular Automata and Combinatoric Tilings in Hyperbolic Spaces, a survey, *Lecture Notes in Computer Sciences*, **2731**, (2003), 48-72.
- [11] Margenstern M., A new way to implement cellular automata on the penta- and heptagrids, *Journal of Cellular Automata* **1**, N° 1, (2006), 1-24.
- [12] Margenstern M., About the domino problem in the hyperbolic plane from an algorithmic point of view, *arXiv:cs.CG/0603093 v1*, (2006), 11p.
- [13] Margenstern M., About the domino problem in the hyperbolic plane, a new solution, *arXiv:cs.CG/0701096*, (2007), January, 60p.
- [14] Margenstern M., About the domino problem in the hyperbolic plane from an algorithmic point of view, *Technical report*, 2006-101, *LITA, Université Paul Verlaine – Metz*, (2006), 100p., available at:
http://www.lita.sciences.univ-metz.fr/~margens/hyp_dominoes.ps.gz
- [15] Margenstern M., About the domino problem in the hyperbolic plane, a new solution, *Technical report*, 2007-102, *LITA, Université Paul Verlaine – Metz*, (2007), 106p., available at:
http://www.lita.sciences.univ-metz.fr/~margens/new_hyp_dominoes.ps.gz
- [16] Margenstern M., The finite tiling problem is undecidable in the hyperbolic plane, *arxiv:cs.CG/0703147v1*, (2007), March, 8p.
- [17] Margenstern M., The periodic domino problem is undecidable in the hyperbolic plane, *arxiv:cs.CG/0703153v1*, (2007), March, 12p.
- [18] Margenstern M., About the domino problem in the hyperbolic plane, a new solution: complement, *arxiv:0705.0086v4*, (2007), May, 20p.
- [19] Margenstern M., The Finite Tiling Problem Is Undecidable in the Hyperbolic Plane, *Workshop on Reachability Problems*, **RP07**, July 2007, Turku, Finland.
- [20] Margenstern M., Cellular Automata in Hyperbolic Spaces, Volume 1, Theory, *OCP*, Philadelphia, (2007), 422p.
- [21] Myers D., Nonrecursive tilings of the plane. II. *Journal of Symbolic Logic*, **39**, (1974), 286-294.
- [22] Robinson R.M. Undecidability and nonperiodicity for tilings of the plane, *Inventiones Mathematicae*, **12**, (1971), 177-209.
- [23] Robinson R.M. Undecidable tiling problems in the hyperbolic plane. *Inventiones Mathematicae*, **44**, (1978), 259-264.
- [24] Wang H. Proving theorems by pattern recognition, *Bell System Tech. J.* vol. **40** (1961), 1-41.

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.

81 Earth rotation

How long is an earth rotation ? Give the answer with a 1 minute precision.

Note : 24 hours is not the good answer !

82 Faces, edges of the n -cube

A 3-cube has $X_0^3 = 8$ vertices, $X_1^3 = 12$ edges, $X_2^3 = 6$ faces and $X_3^3 = 1$ volume. A 2-cube, that is a square, has $X_0^2 = 4$ vertices, $X_1^2 = 4$ edges and $X_2^2 = 1$ face.

What is the formula hidden behind these numbers ?

That is, generalize X_p^n for every $0 \leq p \leq n$ and find a formula for it.

SOLUTIONS TO PREVIOUS PUZZLES

80 A square grid to fill

Put the numbers from 1 to n^2 in an $n \times n$ grid. Consider M the maximum of the differences of the values of next to each other slots (one above the other slots, or one on the left of the other slots). It is very easy to manage to get $M = n$. Is it possible to get $M < n$?

Solution

To begin with, it is easy to see that M must be larger than about $n/2$: Number 1 must be somewhere in the grid, as well as number n^2 . There is a path from 1 to n^2 of length at most $2n$. If the value change is always less than $n/2$, the path is too short to perform the change from 1 to n^2 .

To prove that M must be larger or equal to n , consider the three sets $A = [1, (n^2 - n)/2 + 1]$, $B = [(n^2 - n)/2 + 2, (n^2 + n)/2]$ and $C = [(n^2 + n)/2 + 1, n^2]$. Observe that if $M < n$, there cannot be any contact between A and C , so that the “frontier” of A is made of elements in B . Now, A is large and B is not. Play a little with it, try to put all the elements in A in a surface with a frontier as small as cardinal of B . You will get quickly convinced it is impossible (basically because B is too small to go from one side of the grid to the opposite side. The best way to put $n - 1$ elements to delimit a large surface is to put them just above a diagonal). To prove it properly is technical and boring, I won't do it.

REPORTS FROM CONFERENCES



REPORT ON ICALP 2007 / LICS 2007

34th Intl Colloquium on Algorithms, Languages and Programming and

22nd Annual IEEE Symposium on Logic in Computer Science

8–15 July, 2007, Wrocław, Poland

Manfred Kudlek

ICALP 2007, the 34th in this series of conferences on theoretical computer science, took place from July 9-13, 2007, together with the workshops from July 9-15, 2007, at Wrocław, the first time in Poland. It was co-located with LICS 2007, held from July 10-14, 2007, PPDP 2007 (9th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming), and LC 2007 (Logic Colloquium) which took place from July 14-19, 2007. There were also 9 joint ICALP/LICS satellite workshops. Conference site was the just 2 weeks before finished *Instytut Informatyki Uniwersytetu Wrocławskiego (Institute of Computer Science, University of Wrocław)*. All conferences and workshops took place there.

WROCLAW (VRATISLAVA, BRESLAU) is an old town with many historical buildings many of them having been damaged at the end of the second World War, but carefully reconstructed. Among them is the Town Hall (RATUSZ) and the University Main building (UNIVERSITAS LEOPOLDINA VRATISLAVIENSIS) in Baroque style. It was inaugurated in 1702 in a former Jesuit College.

The conferences were organized by Institute of Computer Science, University of Wrocław. The Organizing Committee consisted of JERZY MARCINKOWSKI (chair), MARCIN BIEŃKOWSKI (WORKSHOPS), ARTUR JEŻ, TOMASZ JURDZIŃSKI, EMANUEL KIEROŃSKI (LICS), PIOTR KOWALSKI (LC), KRZYSZTOF LORYŚ, ALEKSANDER MĄDRY, PAWEŁ RYCHLIKOWSKI (PPDP), PIOTR WIECZOREK (LOCAL ARRANGEMENT), and ALEKSANDRA CZERNECKA, MICHAŁ FALEŃSKI, AGNIESZKA FUSS, AGNIESZKA KUJZA, PAWEŁ LASKOŚ-GRABOWSKI, MARTA STAŃSKA, and KATARZYNA JARMOŁKOWICZ.

ICALP 2007, LICS 2007 were sponsored by UNIVERSITAS VRATISLAVIENSIS, CITY OF WROCLAW, ICALP 2007 also by EATCS, and LICS 2007 also by IEEE Computer Society.

ICALP and LICS together were attended by 255 participants, including also 7 local organizers, together with the workshops by 320. Details are given in the following table where (I,IL,L) means participants for (ICALP, ICALP and LICS, LICS) or for (ICALP, ICALP and LICS, LICS) and workshops, respectively, W for workshops only, and $IT=I+IL$, $LT=L+IL$.

C	I	IL	L	W	IW	LW	C	I	IL	L	W	IT	LT
AU			3			3	IL	6	1	1		7	2
BE	5	1	1		6	2	IN				1		
CA	6	2	4	2	8	6	IT	5	3	4	3	8	7
CH	3	1	1		4	2	JP	5	2	2	5	7	4
CL		1	1			2	NL	1	2	3	1	3	5
CN	1				1		NO	2				2	
CZ		1	2	1		3	PL	25	19	5		44	24
DE	13	4	4	13	17	8	RO	2			1	2	
DK	3				3		SE			2	1		2
EE	1				1		SK	1			1	1	
ES	1	1		2	2	1	TR				1		
FI		1	1	1	1	2	TW	2				2	
FR	8	8	7	14	16	15	UK	6	12	16	13	18	28
GR	4				4		US	26	6	6	5	32	12
HK	1				1		Σ	127	65	63	65	192	128

The scientific program of ICALP 2007 consisted of 6 invited lectures (2 joint with LICS), 1 special talk, and 76 contributions, selected from 240 submissions. coming from 37 countries. 4 papers were withdrawn, 1 more in track A after acceptance. Details on number of authors and distribution by countries for all tracks are given below. The program was divided into 27 sessions (15 in track A, 8 in B, 4 in C), sometimes 3 in parallel, 6 plenary sessions (2 joint with LICS), as well as 1 special session. The entire program of all conferences and the workshops can be found at <http://icalp07.ii.uni.wroc.pl/>.

The following tables give the statistics by number of authors (S submitted, A accepted, W withdrawn) in all tracks and total (1 paper in track A was withdrawn after acceptance), and by countries (C country, I invited, (A,B,C)S submitted, (A,B,C)A accepted:

	A		B		C		Σ	
	S	A	S	A	S	A	S	A
1	33	3	18	4	2		53	7
2	51	17	19	11	17	5	87	32
3	40	11	15	5	11	4	66	20
4	20	9	6	3	3	1	29	13
5	3	1			1		4	1
6	1	1					1	1
	148	42	58	24	34	11	240	77
W	1	1	2		1		4	1

C	I	AS	AA	BS	BA	CS	CA	ΣS	ΣA
AU		$\frac{11}{12}$	$\frac{1}{8}$	$1\frac{1}{2}$		1	$\frac{1}{2}$	$3\frac{5}{12}$	$\frac{5}{8}$
BE				$2\frac{1}{4}$	$1\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{3}$	$2\frac{7}{12}$	$1\frac{7}{12}$
CA		$10\frac{11}{12}$	$2\frac{3}{4}$	$\frac{1}{3}$	$\frac{1}{3}$	2	1	$13\frac{1}{4}$	$4\frac{1}{12}$
CH		3	$\frac{2}{3}$	$2\frac{1}{4}$	$\frac{1}{4}$	$1\frac{1}{2}$	$\frac{1}{2}$	$6\frac{3}{4}$	$1\frac{5}{12}$
CL				$\frac{2}{3}$	$\frac{2}{3}$			$\frac{2}{3}$	$\frac{2}{3}$
CN		$3\frac{3}{4}$	$\frac{1}{2}$	5		3		$11\frac{3}{4}$	$\frac{1}{2}$
CY		$\frac{1}{3}$						$\frac{1}{3}$	
CZ		$\frac{1}{2}$		$\frac{1}{4}$	$\frac{1}{4}$			$\frac{3}{4}$	$\frac{1}{4}$
DE		$21\frac{1}{2}$	$7\frac{11}{12}$	$7\frac{7}{12}$	$4\frac{3}{4}$			$29\frac{1}{12}$	$12\frac{2}{3}$
DK	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{4}$		1		$1\frac{7}{12}$	$\frac{1}{3}$
EE				$\frac{1}{3}$	$\frac{1}{3}$			$\frac{1}{3}$	$\frac{1}{3}$
ES		$1\frac{5}{6}$	$\frac{2}{3}$	$\frac{7}{12}$	$\frac{7}{12}$			$2\frac{5}{12}$	$1\frac{1}{4}$
FI		1						1	
FR		$6\frac{3}{4}$	$2\frac{1}{4}$	$8\frac{1}{12}$	$4\frac{1}{4}$	2		$16\frac{5}{6}$	$6\frac{1}{2}$
GR	$\frac{1}{3}$	$4\frac{2}{3}$	$1\frac{3}{4}$					$4\frac{2}{3}$	$1\frac{3}{4}$
HK		$1\frac{1}{6}$	$\frac{1}{2}$			$1\frac{1}{2}$		$2\frac{2}{3}$	$\frac{1}{2}$
IE		1						1	
IL	$\frac{1}{6}$	$19\frac{9}{10}$	$6\frac{1}{15}$	$\frac{1}{3}$		$1\frac{1}{4}$	$1\frac{1}{4}$	$21\frac{29}{60}$	$7\frac{19}{60}$
IN		$3\frac{1}{5}$	$\frac{1}{10}$	$\frac{1}{3}$		1		$4\frac{8}{15}$	$\frac{1}{10}$
IS		$\frac{1}{4}$						$\frac{1}{4}$	
IT		$2\frac{3}{4}$	$1\frac{1}{6}$	$3\frac{7}{12}$	$1\frac{1}{2}$	1		$7\frac{1}{3}$	$2\frac{2}{3}$
JP		6	1	2	1	1	1	9	3
KR		4		1				5	
MX		$\frac{1}{2}$						$\frac{1}{2}$	
MY						1		1	
NL		$\frac{5}{6}$	$\frac{1}{4}$	$2\frac{2}{3}$	$\frac{2}{3}$			$3\frac{1}{2}$	$\frac{11}{12}$
NO	$\frac{2}{3}$	$1\frac{8}{15}$	$\frac{3}{10}$					$1\frac{8}{15}$	$\frac{3}{10}$
NZ				$\frac{1}{2}$				$\frac{1}{2}$	
PL		$3\frac{1}{4}$		$1\frac{1}{2}$	$\frac{1}{2}$	1		$5\frac{3}{4}$	$\frac{1}{2}$
PT		$\frac{3}{4}$						$\frac{3}{4}$	
RU		1						1	
SE		$1\frac{1}{3}$		$\frac{3}{4}$				$2\frac{1}{12}$	
TH						$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
TR		7						7	
TW		$1\frac{1}{4}$	$1\frac{1}{4}$	1				$2\frac{1}{4}$	$1\frac{1}{4}$
UK	1	$4\frac{19}{20}$	$1\frac{23}{40}$	$4\frac{11}{12}$	$4\frac{11}{12}$	1		$10\frac{13}{15}$	$6\frac{59}{120}$
US	$2\frac{5}{6}$	$31\frac{5}{6}$	$12\frac{5}{6}$	$10\frac{1}{3}$	$2\frac{3}{4}$	$14\frac{1}{12}$	$6\frac{1}{12}$	$56\frac{1}{4}$	$21\frac{2}{3}$
Σ	6	148	42	58	24	34	11	240	77

There were 117 submissions for LICS, from which 39 were accepted, plus 5 short presentations.

C	I	S	A	P	C	I	S	A	P	C	I	S	A	P
AT		$\frac{1}{4}$			DE		9	$\frac{5}{6}$		PL		$3\frac{8}{15}$	$1\frac{8}{15}$	1
AU		$5\frac{4}{5}$	$\frac{4}{5}$	1	DK		$1\frac{1}{2}$			RO		$1\frac{1}{4}$		
BD		1			ES		$2\frac{1}{2}$	$\frac{1}{2}$		SE		$2\frac{2}{3}$		
BE		$\frac{5}{6}$			FI		$\frac{5}{6}$	$\frac{1}{3}$		SG		$1\frac{1}{2}$		
CA		$5\frac{11}{15}$	$4\frac{2}{5}$		FR	$\frac{2}{3}$	$21\frac{1}{12}$	$6\frac{5}{6}$		UA		1		
CH		$\frac{1}{3}$	$\frac{1}{3}$		IL	$\frac{1}{6}$	3			UK	$5\frac{1}{3}$	$21\frac{1}{5}$	$6\frac{5}{6}$	1
CL		$\frac{1}{3}$	$\frac{1}{3}$		IT		$12\frac{1}{3}$	$3\frac{2}{3}$	1	US	$2\frac{2}{6}$	$11\frac{1}{5}$	$2\frac{37}{60}$	1
CN		$3\frac{1}{2}$			JP		$1\frac{2}{3}$	$\frac{2}{3}$						
CZ		$\frac{5}{6}$			NL	1	$3\frac{11}{12}$	$1\frac{5}{12}$		Σ	10	117	39	5

The scientific program of LICS consisted of 10 invited lectures (2 joint with ICALP, 6 joint with LC), 2 special talks, 39 contributions selected from 117 submissions, and 5 short papers, presented in 13 sessions of full and 2 of short papers. The invited talks were in plenary sessions, either for LICS or joint with ICALP or LC.

The scientific program of PPDP 2007 consisted of 1 invited lecture and 13 contributions.

All talks in ICALP and LICS were presented, although some not by their authors.

There were 9 joint ICALP/LICS satellite workshops, attended by 112 participants:

FCS/ARSPA 2007 - 2nd Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis

WCAN 2007 - 3rd Workshop on Cryptography for Ad-hoc Networks

GOCP 2007 - 1st International Workshop on Group-Oriented Cryptographic Protocols

PAuL 2007 - 2nd International Workshop on Probabilistic Automata and Logic

SOS 2007 - 4th Workshop on Structural Operational Semantics

TRSH 2007 - Workshop on Theory of Randomized Search Heuristics

LCC 2007 - 9th International Workshop on Logic and Computational Complexity

DCM 2007 - 3rd International Workshop on Development of Computational Models

TMCNAA 2007 - Workshop on Traced Monoidal Categories, Network Algebras, and Applications

Date, number of invited talks (I), contributions (C), and participants (P) are given in the following table.

WS	7.	I	C	P	WS	7.	I	C	W
FCS/ARSPA	8	1	9	16	TRSH	14-15		9	12
WCAN	8		6	5	LCC	15	4	5	21
GOCP	9	2	7	11	DCM	15	1	10	13
PAUL	9	1	4	14	DCM/TMCNAA			1	
PAUL/SOS			1		TMCNAA	15	7	2	11
SOS	9	1	8	15					

ICALP 2007 was opened on Monday morning by JERZY MARCINKOWSKI, showing a GoogleEarth picture of the conference site with old barracks (the new building was finished only two weeks before), and welcoming the participants.

The first invited ICALP talk by BERNARD CHAZELLE on ‘*Ushering in a New Era of Algorithm Design*’ was a very good overview on that topic. He started with a question from King Kong ‘*If Google is a religion, what is God?*’ and ‘*Every lecture should make only one point, and keep in time. I am out of both.*’ He talked on low energy, noises (showing a monkey at typewriter or ‘cryptographer at work’), and science of the formula as in physics, chemistry, astronomy or in genomics, neuroscience, and networkonomics, finishing with *Dziękuję*. Unfortunately there is only a short abstract in the proceedings.

IVAN DAMGÅRD, in the second ICALP one on ‘*A “Proof Reading” of Some Issues in Cryptography*’ or ‘*Has Provable “Security” Proved to be Good for Anything?*’, gave an excellent overview on security problems, ending with ‘*How to proceed in research and practice?*’ and the conclusion ‘*Provable security proved useful*’.

THOMAS C. HALES, with the first invited LICS lecture ‘*The Kepler Conjecture and the Flyspeck Project*’, or ‘*A Collection of Problems in Geometry*’, presented a very good introduction into the history and solution of the conjecture, finally published in 2006.

FRED B. SCHNEIDER presented a nice third invited ICALP talk with ‘*Credentials-Based Authorization: Evaluation and Implementation*’ on problems of authorization in the context of the Nexus OS, and ways to solve them as with credentials, certain logics and proof methods, as well as the open question of completeness. Unfortunately the proceedings contain only an abstract.

FEDOR V. FOMIN (co-authors FREDERIC DORN, DIMITRIOS M. THILIKOS) gave a fast fourth invited ICALP lecture ‘*Subexponential Parameterized Algorithms*’, starting with the problem of placing fire brigade stations in Bergen, talking on frameworks for parallel algorithms as combinatorial bounds and dynamic programming, and algorithms on planar graphs.

The second invited LICS talk '*Reflections on Finite Model Theory*' by PHOKION G. KOLAITIS was an excellent overview on the history and importance for LICS of that field. He started with '*What is FMT?*' and finished with '*FMT is a very good prison for me*'.

GORDON D. PLOTKIN with the fifth ICALP/third LICS invited lecture '*The Algebraic Theory of Effects*' gave an excellent presentation on functional programming, relations between type theory, proof theory and categorical logic, on non-pure computation, interactive IO, deconstructors, algebras and co-algebras, logic for programming languages, logic for effects, Scott/Milner LCF, dynamic logic. He started with '*Some important ideas have never been published*' and finished with '*To do many things as operational semantics, Hoare logic, modularity in concurrency*'. Unfortunately, his paper is neither contained in the ICALP nor LICS proceedings.

With the sixth ICALP/fourth LICS invited talk '*Highly Efficient Secrecy-preserving Proof Correctness of Computations, and Applications*' MICHAEL O. RABIN gave an excellent presentation on zero-knowledge proofs, multiparty computation, and on a highly efficient method to prove correctness of computations preserving secrecy of input ('*Wiping out traces in the sand*'), verifying value consistency ('*OK? Elementary, Dr. Watson!*'), done in an evaluator model. He also presented an application to auctions, giving shocking experimental results for 100 bidders, finishing with '*Who is first bidder for question?*'.

There were other 6 joint LICS/LC invited lectures

MARTIN HYLAND (co-authors RUSS HARMER, PAUL-ANDRÉ MELLIÈS), gave an excellent talk with '*Combinatorics of Proofs*' or '*Categorical Combinatorics for Innocent Strategies*', referring to a talk at LC 31 years ago, reflecting on *extreme* abstractions as category theory, game semantics, and linear logic, and finishing with '*I hope this style of approach will extend*'.

ROSALIE IEMHOFF, with '*Skolemization in Constructive Theories*', had a nice presentation on an alternative to Skolemization implying an analogue to Herbrand's theorem.

Another good presentation was given by ALEX SIMPSON with '*Non-well-founded Proofs*' or '*Formal Borel Sets (A Proof-theoretic Approach)*' on domain theory, topological models, and probabilistic models, concentrating on proof theory and locality of random sequences.

An excellent presentation '*Higher-order Matching, Games and Automata*' with colourful slides was given by COLIN STIRLING on model checking of higher order schemes, application of tree automata, higher order unification, and relation between model and satisfiability checking.

CRISTIANO CALCAGNO (with co-authors JOSH BERDINE, DINO DISTEFANO, PETER O'HEARN, MATTHEW PARKINSON, VIKTOR VAFEIADIS, HONGSEOK YANG) gave a nice

talk with ‘*Can Logic Tame Systems Programs?*’ on designing and implementing tools for automatic reasoning about safety of systems programs.

A good presentation was given by MARTÍN ESCARDÓ with ‘*Infinite Sets that Admit Exhaustive Search I*’, referring to part II as most highly rated paper in LICS 2007, on relations between topology and computability as continuous, open, clopen, discrete, Hausdorff, compact vs. computable, semi-decidable, decidable, semi-decidable equality, semi-decidable compactness, exhaustively searchable, respectively. He finished with ‘*This conjecture which is in the paper is probably wrong*’.

Abstracts of these 6 talks can be found in the abstract book of LC.

Also to mention are three special award lectures. The EATCS DISTINGUISHED AWARD, after a short introduction by WOLFGANG THOMAS on the scientific work, was given on Monday afternoon to DANA STEWART SCOTT. The awarded then presented an excellent lecture ‘*Looking to the Future*’ in which he started with a list of previous winners, all good friends of him, and particularly congratulating MICHAEL RABIN. He continued with *very many funny things*, what is going to be the next big thing (solving $P=NP?$, building a quantum computer?), and what are not next big things (aesthetic, natural, pervasive computing, stream computing theory). The next big thing (‘*That’s it!*’) is multicore computing having no tools yet. He then gave a short history of computers, its main frames until web computing and the return of supercomputers for parallel computing, finishing with ‘*It’s time for EATCS to move into the Real World!*’.

TWO LICS TEST-OF-TIME AWARDS (for papers 20 years ago) were distributed on Tuesday afternoon. One was presented to SAMSON ABRAMSKY for his paper ‘*Domain Theory in Logical Form*’, who then gave a very nice lecture on ‘*Domain Theory in Logical Form: 20 Years Later*’, talking on precursors and successors in the field and ending with personal reflections like ‘*Now it can be solved! A quirky fact*’ (not having PhD thesis at hand), that a PhD thesis is not a life sentence, and the importance of LICS.

The other one was presented to GORDON D. PLOTKIN for the paper ‘*A Framework for Defining Logics*’ (co-authors ROBERT HARPER, FURIO HONSELL). The awarded then gave a very good talk on ‘*Local Frameworks, Initial Ideas, and Further Development*’, as on logical frameworks, Martin Löf theory, type systems, and what happened next to Edinburgh, at other places, and what is done now.

The winners of the best ICALP papers were JIN-YI CAI, PINYAN LU for ‘*Holographic Algorithms: The Power of Dimensionality Resolved*’ in track A, BERNARD BOIGELOT, JULIEN BRUSTEN for ‘*A Generalization of Cobham’s Theorem to Automata over Real Numbers*’ in track B, and TAL MORAN, MONI NAOR, GIL SEGEV for ‘*Deterministic History-independent Strategies for Storing Information on Write-once Memories*’ in track C. Unfortunately, those from track C could not be present,

their paper being well presented by TAL MALKIN, but JIN-YI CAI and BERNARD BOIGELOT gave excellent presentations of their papers.

The winner of the KLEENE Award for the best student paper in LICS, selected from 5 candidates, was NIKOS TZEVELEKOS for '*Full Abstraction for Nominal General Reference*'.

Among the very many good other contributions only few, personally interested, can be mentioned here. Thus there were very good and interesting presentations by PETER BÜRGISSEER on new complexity classes over the reals induced by exotic quantifiers, by MANUEL BODIRSKY on computational complexity of constraint problems over infinite domains, by LAURENT BIENVENU on relations between data compression and Kolmogorov complexity, and by THOMAS COLCOMBET on deterministic Ramseyan factorization and its application to monadic logic and infinite structures.

LUTZ SCHRÖDER, introduced a *nirwana state* in his talk on modal logic of Segala system, and CHUNG-KIL HUR caused DANA SCOTT to give the advice '*Don't show with laser to audience*'.

Also to mention is the excellent invited lecture '*Copy-cat Strategies and Information Flow in Physics, Geometry, Logic and Computation*' by SAMSON ABRAMSKY in DCM, talking on emergence, cloning and relations between linear and classic logic, quantum and classical physics, linear time and computational universality.

In TCMNAA 7 good and interesting invited lectures were presented: MASAHIITO HASEGAWA with '*On Traced Monoidal Closed Categories*', PAULO OLIVA with '*Hoare Logic in the Abstract*', PRAKASH PANANGADEN with '*Traces, Kahn Networks and the Geometry of Interaction*', GHEORGHE ȘTEFĂNESCU with '*Streams, Network Algebras, and Interactive Systems*', Gordon Plotkin with '*Completeness of TMC's wrt. FDVec*', DUŠKO PAVLOVIĆ with '*Traces of Intruders II: From Copy-cat, through Man-in-the-middle, to Automated Turing Test*', and MARTIN HYLAND with '*Categorical Aspects of the Trace*'.

A highlight was the ceremony in the Baroque AULA LEOPOLDINA of the Main University Building on Thursday afternoon to award the degree of *Doctor Honoris Causa* to MICHAEL OSER RABIN who was born in BRESLAU in 1931. It started with the entry of university professors guiding MICHAEL RABIN, all in traditional Academic dresses. LESZEK PACHOLSKI, the university rector, welcomed the awarded, his wife RUTH and sister MIRIAM BEN-PERETZ, and referred to a ceremony on October 14, 1913, granting the PhD in literature to ELS HESS, Michael Rabin's mother. TOMASZ ROLSKI, Dean of Faculty of Mathematics and Computer Science, presented Michael Rabin's Curriculum Vitae. After that, LESZEK PACHOLSKI, in Latin and English, presented the Dr. H.c. to the awarded. Michael Rabin then expressed his personal feelings on science, talked about the start of his scientific life, mentioned Kleene's book *Metamathematics*, on a dialogue when asked by a

biology professor (*'What do you want to study? - Foundations of Computing - What are they usefuk for? - Use in Biology - That will never happen!'*), that his early results in computer science have been absolutely disregarded by mathematicians, the influence by ALFRED TARSKI and ANDRZEJ MOSTOWSKI, randomizing for primality test, and his marriage and daughter. He finished with *'Not study for prices but learning of its own'*. The ceremony was accompanied by an Academic student choir, singing in Polish, English and Latin.

The proceedings of ICALP 2007, edited by LARS ARGE, CHRISTIAN CACHIN, TOMASZ JURDZIŃSKI, and ANDRZEJ TARLECKI, have been published as Springer LNCS 4596. They contain the 2 invited papers by DAMGÅRD, FOMIN, and abstracts of those by CHAZELLE, SCHNEIDER, and all other contributions. The proceedings of LICS 2007, edited by LUKE ONG, have been published by IEEE Computer Society. They contain the invited papers by HALES, RABIN, KOLAITIS, STIRLING, HYLAND, and all other normal contributions.

The EATCS General Assembly was held on Tuesday late afternoon. On it there is a separate report. THOMAS HENZINGER got an EATCS button by the author of this report for having reached more than 5 full papers on ICALP's. Each of the four editors of the proceedings received a button, too. The current state of busy contributors to ICALP's is given in the table below.

The social program started on Tuesday evening with a welcome reception in the Baroque ORATORIUM MARIANUM in the Main University Building, with cold and warm buffet, and drinks. LESZEK PACHOLSKI, the rector of the university, announced a speech of only 45 minutes on the university, its history and buildings, but it lasted only a few minutes, including the welcome for the participants of ICALP and LICS. GIORGIO AUSIELLO thanked the organizers. It ended by 22 h.

Because of some rain on Wednesday the excursion to the southwest of Wrocław was slightly changed, cancelling the visit of ŚLĘŻA mountain where a picnic and a 2 hour hiking tour had been planned. We first visited the Peace Church at ŚWIDNICA commemorating the 30 years war peace treaty in 1648. It is constructed entirely from wood and was finished within one year in 1657. Later we had a guided tour through a part of a huge tunnel system in the SOWIE mountains near OSÓWKA, built by prisoners of the Nazi regime in the 1940's, of whom only few survived the hard conditions. The last stop was at PARK WOJSŁAWICE near NIEMCZA, with an Arboretum. There also a BBQ party was offered, with typical Polish dishes as PIEROGY, BIGOS and draft Polish beer. It was about 23 h when we returned to Wrocław.

On Thursday evening a cocktail party was given in the Town Hall of Wrocław, with cold, warm buffet and drinks, and Polish dishes again. The Lord Mayor RAFAŁ DUTKIEWICZ, with PhD in Logics, having prepared a speech of 23 pages, but didn't use it and welcomed us and in particular MICHAEL RABIN, talking about

Kurt Mehlhorn	11	ICALP Contributors	
Jean-Eric Pin	$10\frac{1}{2}$		
Juhani Karhumäki	$9\frac{7}{60}$	Arnold Schönhage	5
Zvi Galil	8	Thomas Henzinger	5
Amir Pnueli	$7\frac{1}{2}$		
Mihalis Yannakakis	$7\frac{5}{12}$	Dominique Perrin	$4\frac{5}{6}$
Philippe Flajolet	$7\frac{1}{4}$	Zohar Manna	$4\frac{5}{6}$
Grzegorz Rozenberg	7	Juraj Hromkovič	$4\frac{7}{10}$
Paul Vitányi	$6\frac{11}{12}$	Denis Thérien	$4\frac{7}{12}$
Claus-Peter Schnorr	$6\frac{1}{2}$	Moshe Vardi	$4\frac{7}{12}$
Torben Hagerup	$6\frac{1}{2}$	Manfred Droste	$4\frac{1}{2}$
Géraud Sénizergues	$6\frac{1}{2}$	Robin Milner	$4\frac{1}{2}$
Karel Čulík II	6	David Peleg	$4\frac{9}{20}$
Walter Vogler	6	Ming Li	$4\frac{5}{12}$
Christos Papadimitriou	$5\frac{5}{6}$	Maurice Nivat	$4\frac{1}{4}$
John Reif	$5\frac{3}{4}$	Moti Yung	$4\frac{1}{4}$
Joost Engelfriet	$5\frac{1}{2}$	Volker Diekert	$4\frac{1}{6}$
Matthew Hennessy	$5\frac{1}{2}$	Piotr Berman	$4\frac{1}{6}$
Arto Salomaa	$5\frac{1}{2}$	Marek Karpieński	$4\frac{1}{6}$
Juris Hartmanis	$5\frac{1}{3}$	Thomas Wilke	$4\frac{1}{6}$
Andrzej Lingas	$5\frac{1}{3}$	Christophe Reutenauer	4
Burkhard Monien	$5\frac{19}{60}$	Marcel Paul Schützenberger	4
Michael Rabin	$5\frac{1}{3}$	Davide Sangiorgi	4
Ronald Book	$5\frac{1}{4}$	Leslie Valiant	4
Christian Choffrut	5	Colin Stirling	4

the town. MICHAEL RABIN, born in BRESLAU (the German name of the town) in 1931, accompanied by his wife RUTH and his sister MIRIAM BEN-PERETZ, briefly talked about changing names. During the party the group CZTERY PORY ROKU (4 seasons) was playing classical music. It ended around 22 h.

Following the Honorary PhD ceremony for Michael Rabin another reception was given in Oratorium Marianum on Friday evening.

On Sunday evening a reception of ASL took place in the Botanical Garden, also with cold and warm buffet, and drinks.

There were 40 PC's for access to Internet. Again, SPRINGER and ELSEVIER were present with book exhibitions, as well as GOOGLE.

Lunch was served in the restaurant of the Informatics Institute.

Most participants stayed in hotels HP PARK PLAZA, JOHN PAUL 2 PILGRIM'S HOUSE, TUMSKI, PATIO, DUET, ZAŁĘK, CAMPANILE, and the student residence OŁÓWEK.

The temperatures the day before ICALP started were around 35° but went down considerably to less the 20°, with rain. Only on Friday noon it became summer again with more than 30°.

Altogether, ICALP, LICS, PPD, LC, and the workshops were successful, of high quality, well organized, with quite a number of highlights. The atmosphere was relaxed. Next ICALP will be held in REYKJAVÍK on Iceland, from July 6-13, 2008.

DO WIDZENIA WROCŁAW AND VELKOMINN Á REYKJAVÍK.

Pictures from LICS&ICALP 07 (by M. Kudlek)



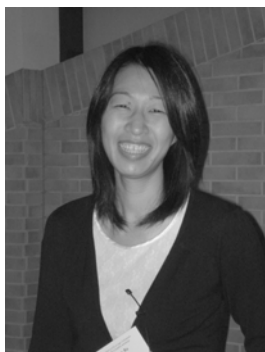
Jerzy Marcinkowski



Bernard Chazelle



Lars Arge



Ying Xu



Stephanie Naewe



Rudy Raymond



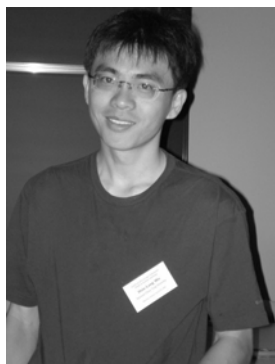
Ashley Montanaro



Matei David



Ming-Yang Kao



Hsin-Lung Wu



Karl Wimmer



Peter B rgisser



Ivan Damg rd



Christian Cachin



Tal Malkin



Payman Mohassel



Aggelos Kiayias



Lior Malka



Thomas C. Hales



Martin Abadi



Magdalena Grüber



Fred B. Schneider



Lutz Schröder



Richard Min



Manuel Bodirsky



Anuj Dawar



Nicole Schweikardt



Fedor V. Fomin



Chung-Kil Hur



Tarmo Uustalu



Pinyan Lu (track
A best paper)



Laurent Bienvenu



Gary Miller



Sylvain Schmitz



Takashi Suto



Phokion Kolaitis



Balder ten Cate



Bernard Boigelot
(track B best paper)



Vinayak Prabhu



Marcin Jurdziński



Wiesław Zielonka



Gordon Plotkin



Henrik Björklund



Wong Karianto



Pablo Barcelo



Thomas Colcombet



Michael Rabin



Salvatore La Torre



Gennaro Perlatto



Steffen Lempp,
opening of LC



Leszek Pacholski,
opening LC



Andrew Pitts



Martin Hyland



Rosalie Iemhoff



Alex Simpson



Colin Stirling



Cristiano Calcagno



Martín Escardó



Samson Abramsky



Ian Mackie



Ellie d'Hondt



Simon Perdrix



Elham Kashefi



Pablo Arrighi



Philip Scott



Prakash Panangaden



Gheorghe Ștefănescu



Masahito Hasegawa



Pablo Oliva



Gordon Plotkin



Peter Hines



Duško Pavlović

Awards and Prices



Peter Hyland



Giorgio Ausiello



Christian Cachin



Jin-Yi Cai, Pinyan Lu



Andrzej Tarlecki,
Giorgio Ausiello,
Bernard Boigelot,
Julien Brusten



Dana Scott,
Giorgio Ausiello,
Wolfgang Thomas



Wolfgang Thomas,
Catuscia Palamidessi,
Prakash Panangaden



Martin Abadi,
Gordon Plotkin



Samson Abramsky,
Gordon Plotkin



Wolfgang Thomas



Dana Scott



Luke Ong



Samson Abramsky



Gordon Plotkin



Mohammad Reza Mousavi

Receptions and Excursions



Paul Spirakis



Lord Mayor Rafał
Dutkiewicz



Jean-Pierre
Jouannaud and wife



Leszek Pacholski,
Giorgio Ausiello



Manfred Kudlek



Phokion Kolaitis,
Lauri Hella



Michael Rabin,
Leszek Pacholski



Michael Rabin,
Rafał Dutkiewicz,
Leszek Pacholski



Miriam Ben-Peretz,
Ruth Rabin



Michael Rabin, Dana
Scott, Ruth Rabin



Paul Spirakis,
Giorgio Ausiello



Leszek Pacholski,
Michael Rabin



Prakash Panangaden,
Astrid Kiehn



Mariangiola Dezani,
Pierpaolo Degano



Branislav Rován,
Jean-Pierre Jouannaud



Quartet Cztery Pory Roku



Aula Leopoldina



Students Choir



Michael Rabin
receiving Hon. PhD



Ruth and Michael Rabin,
Leszek Pacholski



Michael Rabin



Prakash Panangaden,
Samson Abramsky



Gordon Plotkin, Dana
Scott, Samson Abramsky



Bakh Khoussainov,
Petr Hájek



Gheorghe Ștefănescu,
Samson Abramsky



Chairlady Anna
Lysyanskaya



Anuj Dawar, Martin
Hyland, Leszek Pacholski

REPORT ON AGT 2007

Algorithmic Game Theory workshop, Warwick, 2007

Mary Cryan, University of Edinburgh



The University of Warwick, which has a long-standing history in Algorithms & Complexity, was recently awarded an EPSRC/HEFCE grant to establish a new Centre for Discrete Mathematics and its Applications (DIMAP). The support given has enabled the establishment of one Professorship in Computer Science, three permanent lectureships (in Computer Science, Mathematics, and Operations Research), and the appointment of several postdoctoral research fellows and PhD students. DIMAP also has plans to hold workshops, and host visitors, in the area of Discrete Algorithms. More details can be found on the DIMAP website, at <http://www.dcs.warwick.ac.uk/dimap/>.

This “conference report” is reporting on the first DIMAP event, the international workshop on Algorithmic Game Theory (AGT), which was held during March 25 - 28th, 2007. The AGT workshop took place in the (relatively) new Mathematics and Computer Science buildings on the University of Warwick campus, located in the West Midlands region of the UK. The five keynote speakers for the workshop were Noam Nisan, Christos Papadimitriou, Tom Roughgarden, Eva Tardos and Vijay Vazirani. There were 23 other invited speakers, including many well-known figures from the world of Algorithmic Game Theory. Added to this

were many students, faculty and researchers from the UK, Europe, and the rest of the world, making a total of about 75 enthusiastic participants.

The workshop started early on the morning of Sunday 25th, with Uri Zwick giving the opening talk on “Simple Stochastic Games, Mean Payoff Games and Parity Games”, standing in for keynote speaker Vijay Vazirani, whose plane had been delayed. Vijay eventually arrived after lunch, and in the afternoon presented his talk about algorithms for computing equilibria in classical and modern markets, including some applications for Google’s AdWords. The day’s program ended with two talks on unrelated machine scheduling, by Yossi Azar and Elias Koutsoupias. The evening meal took place in Radcliffe House on campus.

The first talk on Monday was given by Éva Tardos, who presented game theoretic results for price-setting in Networks in a perfect-information setting. The main theme on Monday was Nash Equilibria, and in particular different notions of approximation for Nash Equilibria. Kousha Etessami discussed the complexity of finding an ϵ -close approximation to a Nash equilibrium vector, and related this problem to the Sqrt-Sum problem. After lunch Christos Papadimitriou presented work on a different notion of approximation: the ϵ -approximate Nash Equilibrium (where no player may improve by more than ϵ by changing strategy). He discussed this problem in the context of pure and mixed strategies. For mixed strategies, there is no PTAS for this problem unless $\text{PPAD}=\text{P}$; however a 0.38-approximation is known to exist. In the afternoon, Paul Spirakis continued the subject of ϵ -approximate NE, in particular well-supported ϵ -approximate NE. On Monday evening there was a short reception to mark the official opening of DIMAP, followed by a panel discussion on the future of Algorithmic Game Theory. Members of the panel had differing opinions on the amount of remaining work to be done on Algorithmic Game Theory; however everyone was able to come up with open problems. These varied from specific questions such as resolving the approximation status of the ϵ -approximate NE problem, to Frank Kelly’s hopes for a game-theoretic solution to the problem of email spam.

Tim Roughgarden kicked off Tuesday’s program with a talk about Moulin Mechanisms with low social cost, giving a broad overview of what has been done in this area, as well as presenting recent results for the network design problem. Most of Tuesday’s talks continued on the theme of network design and congestion in networks. Talks ended early at 3pm for the workshop excursion to Kenilworth Castle, a large ruined castle about 3 miles from the University. It was a sunny day so we spent a couple of hours looking through the buildings and walking round the gardens. After that we took the scenic route (via a pub) to the Cross restaurant in Kenilworth town, where an excellent dinner was served.

The final set of talks took place on Wednesday morning. The last talk of the workshop was given by Piotr Krysta, who spoke about approximation algorithms and NP-hardness results for different multi-product pricing systems.

I have only mentioned a few of the talks in this report. The complete and final schedule of the workshop (with slides for each of the talks) can be seen at <http://www.dcs.warwick.ac.uk/dimap/EVENTS/AGT-2007/schedule.html>.

The AGT workshop organisers were Artur Czumaj, Marcin Jurdzinski, Mike Paterson, and Rahul Savani, all from the University of Warwick. They did an excellent job of taking care of registration, email facilities, and showing us around Kenilworth on Tuesday. Many participants were captured on camera by Artur Czumaj, Mike Paterson and Michal Rutkowski. The photos appear online at <http://www.dcs.warwick.ac.uk/~agt2007/GALLERY/>

REPORT ON WG 2007

The 33rd International Workshop on Graph Theoretic Concepts in Computer Science

Fedor V. Fomin

From June 21 – 23, 2007, the **33rd International Workshop on Graph-Theoretic Concepts in Computer Science**, WG 2007, took place in a nice castle at Dornburg (which is near Jena), Germany.

The program consisted from 30 regular (out of 99 submissions) and 2 invited lectures. The invited talks were given by Ming-Yang Kao on *Algorithmic DNA Assembly* and by Klaus Jansen on *Approximation algorithms for geometric intersection graphs*. There were 73 participants.

Each of the 30 accepted papers was presented at the meeting. The talks showed a variety of topics concerning graphs, with many of their aspects in relation to computer science. The overall quality of the presented results and the presentations was high. These talks and the two invited lectures made an excellent scientific program.

The social program was highly cultural. It consisted of a nice organ concert in an old church in Dornburg and an excursion to Weimar, die Stadt von Goethe und Schiller. The conference dinner was in a restaurant on the top of the tower, 120 m high over Jena. Nice location together with Thuringian Cuisine created an unforgettable experience.

The excellent organization, the very interesting and dense scientific program, nice local beer, and the pleasant atmosphere among the participants made this a very enjoyable meeting.

WG 2008 will be held in Durham, UK, and this is the first time the workshop is leaving continental Europe. We are looking forward to it!

ABSTRACTS OF PHD THESES



Abstract of PhD Thesis

Author: Sylvain Schmitz
Title: Approximating Context-Free Grammars
for Parsing and Verification
Language: English
Supervisor: Jacques Farré
Institute: Université de Nice - Sophia Antipolis, France
Date: 24 September 2007

Abstract

Programming languages developers are blessed with the availability of efficient, powerful tools for parser generation. But even with automated help, the implementation of a parser remains often overly complex.

Although programs convey an unambiguous meaning, the parsers produced for their syntax, specified by a context-free grammar, are most often nondeterministic, and even ambiguous. Facing the limitations of traditional deterministic parsers, developers have embraced general parsing techniques, capable of exploring every nondeterministic choice, but at the expense of any unambiguity warranty. The real challenge in a parser development lies then in the proper identification and treatment of ambiguities—but these issues are undecidable.

In this thesis, we show two ways to better deal with nondeterminism while keeping the guarantee of unambiguity: the use of noncanonical parsers, through the examples of the Noncanonical LALR(1) and Shift-Resolve constructions, and a conservative ambiguity detection algorithm. Both can be seen as the result of a static analysis on the grammar. Guided by the intuition that most parsing techniques operate a form of left-to-right depth-first walk in the derivation trees of the grammar, we define the *position graph* of a grammar as the set of all these walks, and a *position automaton* as a quotient of a position graph by an equivalence relation between positions in derivation trees. We obtain various levels of approximation when we choose refined or coarser equivalences, and thus position automata provide a fairly general approximation framework, in which several classical parser generation techniques can be expressed. In particular, the states of a position automaton generalize the notion of *items* usually employed to denote a position in a grammar. Besides the generation of noncanonical parsers and the detection of ambiguity, we also apply position automata to two small problems, namely the recognition of derivation trees by means of a finite state automaton,

and the generation of canonical bottom-up parsers for $LR(k)$ and strict deterministic grammars.

The two noncanonical parsing techniques that we study, called Noncanonical LALR(1) and Shift-Resolve parsing, contribute to the range of practical noncanonical parsing methods, until now mostly reduced to Noncanonical SLR(1) parsing. Noncanonical parsers rely on the remaining text to help choose between parsing actions; they are able to perform parsing actions in this right context, and pull back in order to use the information gathered there to help with resolving earlier conflicts. They enforce both the unambiguity of the grammar and a linear time parsing bound. Our Noncanonical LALR(1) construction is more than just a step from simple lookaheads to contextual ones, as it can be seen as a generic construction of a noncanonical parser from a canonical one. The Shift-Resolve parsing construction further exploits position automata to compute the reduced lookahead lengths it needs for each parsing action, but keeps these lengths finite in order to preserve the linear parsing time complexity. It solves a major drawback of noncanonical parsers, which are either limited to a reduced lookahead window of fixed length, or in contrary not limited at all but with a quadratic parsing time complexity in the worst case.

Our conservative algorithm for ambiguity detection follows some of the principles we designed in noncanonical parser constructions. The algorithm is designed to work on any position automaton, allowing for various degrees of precision in the ambiguity report, depending on the chosen position equivalence. Our method is conservative in that it cannot return false negatives, and it is therefore safe to employ a grammar it reports as unambiguous, for whatever level of approximation we might choose. Thanks to the generality of the position automata framework, other means for ambiguity detection can be compared formally to our technique, and we prove that it generalizes in particular the LR-Regular tests. The practical experiments conducted so far support the adequacy of the algorithm, and suggest several developments that should be undertaken in the future.

Table of Contents

1 Introduction 1

2 General Background 5

 2.1 Topics 5

 2.2 Scope 13

3 Advanced Parsers and Their Issues 25

 3.1 Case Studies 26

3.2 Advanced Deterministic Parsing	34
3.3 General Parsing	39
4 Grammar Approximations	47
4.1 Derivation Trees	47
4.2 Position Automata	53
4.3 Recognizing Derivation Trees	65
4.4 Related Models	76
5 Parser Generation	79
5.1 Parser Generation from Position Automata	80
5.2 Noncanonical LALR(1) Parsers	86
5.3 Shift-Resolve Parsers	112
6 Ambiguity Detection	123
6.1 Regular Unambiguity	124
6.2 Noncanonical Unambiguity	134
6.3 Practical Results	144
7 Conclusion	157

Author's correspondence address Sylvain Schmitz
LORIA
Campus Scientifique
BP 239
54506 Vandœuvre-lès-Nancy Cedex
France

Abstract of PhD Thesis

Author: Josep Silva
Title: Debugging Techniques for Declarative Languages:
Profiling, Program Slicing, and Algorithmic Debugging
Language: English
Supervisor: Germán Vidal
Institute: Technical University of Valencia, Spain
Date: June 29 2007

Abstract

The task of debugging can be arduous. A bug can be evident with a single glance, or it can be hidden in the deepest lurking place of our program. Nevertheless, surprisingly, debugging is one of the software processes that has been mostly abandoned by the scientific community, and the same debugging techniques used twenty years ago are still being used today.

The situation is not different regarding declarative languages. Or it is indeed worst; because declarative languages can pose additional difficulties due, for instance, to the lazy evaluation mechanism.

In this thesis, we revise the current debugging methods for declarative languages and we develop some new methods and techniques which are based on profiling, program slicing, and algorithmic debugging. In short, the main contributions of the thesis are:

- The definition of a profiling scheme for functional logic programs which is based on the use of *cost centers* and that allows us to measure different kinds of symbolic costs.
- The formulation of a new dynamic slicing technique based on *redex trails*, its application to debugging and its adaptation for the specialization of modern multi-paradigm declarative programs.
- The introduction of a new algorithmic debugging scheme which combines conventional algorithmic debugging with program slicing.
- The definition of three new strategies for algorithmic debugging.
- The development of a comparative study and a subsequent classification of program slicing techniques and algorithmic debugging strategies.

Table of Contents

1	Introduction	1
1.1	Debugging Declarative Languages	11
1.2	Debugging Techniques	13
1.3	Structure of the Thesis	13
2	Preliminaries	11
2.1	Signature and Terms	11
2.2	Substitutions	13
2.3	Term Rewriting Systems	13
2.4	Declarative Multi-Paradigm Languages	13
3	Time Equations for Lazy Functional (Logic) Languages	17
3.1	Introduction	17
3.2	Cost Augmented Semantics	19
3.3	The Program Transformation	22
3.4	The Transformation in Practice	70
3.5	Related Work	70
4	Runtime Profiling of Functional Logic Programs	75
4.1	Introduction	75
4.2	A Runtime Profiling Scheme	79
4.3	Cost Semantics	85
4.4	Cost Instrumentation	96
4.5	Implementation	96
4.6	Related Work and Conclusions	96
5	Dynamic Slicing Based on Redex Trails	103
5.1	Introduction	103
5.2	Combining Tracing and Slicing	107
5.3	Building the Extended Trail	107
5.4	Computing the Slice	107

5.5 Program Specialization Based on Dynamic Slicing	107
5.6 Implementation Issues	107
5.7 Related Work	107
6 Combining Program Slicing and Algorithmic Debugging	115
6.1 Introduction	115
6.2 The Idea: Improved Algorithmic Debugging	125
6.3 Computation Graphs	125
6.4 The Augmented Redex Trails	125
6.5 Slicing the ART	125
6.6 Combining Program Slicing and Algorithmic Debugging	125
6.7 The Technique in Practice	125
6.8 Related Work	125
7 A Classification of Program Slicing and Algorithmic Debugging Techniques	115
7.1 Introduction	115
7.2 Classifying Slicing Techniques	125
7.3 Three New Algorithmic Debugging Techniques	125
7.4 Classifying Algorithmic Debugging Strategies	125
8 Conclusions and Future Work	115
8.1 Conclusions	115
8.2 Future Work	125
9 Acknowledgements	115

Author's correspondence address

Josep Francesc Silva Galiana
 Universidad Politécnica de Valencia
 Dept. Sistemas Informáticos y Computación
 Camino de Vera, s/n
 46022 - Valencia
 Spain

Abstract of PhD Thesis

Author: Simon Kramer
Title: Logical Concepts in Cryptography
Language: English
Supervisor: Thomas A. Henzinger and Uwe Nestmann
Institute: Ecole Polytechnique Fédérale de Lausanne
Switzerland
Date: July 4, 2007

Abstract

The thesis is about a breadth-first exploration of logical concepts in cryptography and their linguistic abstraction and model-theoretic combination in a comprehensive logical system, called CPL (for *Cryptographic Protocol Logic*). We focus on two fundamental aspects of cryptography. Namely, the security of *communication* (as opposed to security of *storage*) and cryptographic *protocols* (as opposed to cryptographic *operators*). The logical concepts explored are the following. PRIMARY CONCEPTS: the *modal* concepts of belief, knowledge, norms, provability, space, and time. SECONDARY CONCEPTS: belief with error control, individual and propositional knowledge, confidentiality norms, truth-functional and relevant (in particular, intuitionistic) implication, multiple and complex truth values, and program types. The distinguishing feature of CPL is that it unifies and refines a variety of existing approaches. This feature is the result of our *wholistic conception* of property-based (modal logics) and model-based (process algebra) formalisms. We illustrate the expressiveness of CPL on representative *requirements engineering* case studies. Further, we extend (core) CPL (qualitative time) with *rational-valued time*, i.e., time stamps, timed keys, and potentially drifting local clocks, to tCPL (quantitative time). Our extension is conservative and provides further evidence for Lamport's claim that adding real time to an untimed formalism is really simple. Furthermore, we sketch an extension of (core) CPL with a notion of *probabilistic polynomial-time* (PP) computation. We illustrate the expressiveness of this extended logic (ppCPL) on tentative formalisation case studies of fundamental and applied concepts. *Fundamental concepts*: (1) one-way function, (2) hard-core predicate, (3) computational indistinguishability, (4) (*n*-party) interactive proof, and (5) (*n*-prover) zero-knowledge. *Applied concepts*: (1) security of encryption schemes, (2) unforgeability of signature schemes, (3) attacks on encryption schemes, (4) attacks on signature schemes, and (5) breaks of signature

schemes. In the light of logic, adding PP to a formalism for cryptographic protocols is perhaps also simple and can be achieved with an Ockham's razor extension of an existing core logic, namely CPL.

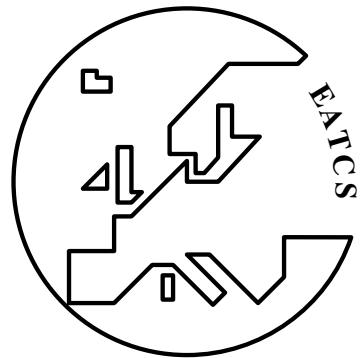
Moreover, we define: (1) *message meaning*; (2) *message information content*; (3) *protocol meaning*; and, based on all that, (4) *protocol information content*. From the meaning of a cryptographic message, we obtain (1) an equational definition of its *context-sensitivity*, and (2) a *formalisation* of the first of Abadi and Needham's principles for prudent engineering practice for cryptographic protocols. From the meaning of a cryptographic protocol, we obtain natural definitions of the concepts of (1) a *protocol invariant*, (2) *protocol safety*, and (3) *protocol refinement*. Last but not least, we show that *protocol agents* can be conceived as evolving *Scott information systems*.

Keywords applied formal logic, information security.

URL <http://library.epfl.ch/en/theses/?nr=3845>

Contact simon.kramer@a3.epfl.ch
<http://mtc.epfl.ch/~skramer/>

European
Association for
Theoretical
Computer
Science



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, and a Treasurer. 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;"
- Award of research and academic careers prizes, including the "EATCS Award," the "Gödel Prize" (with SIGACT) and best papers awards at several top conferences;
- Active involvement in publications generally within 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: ETAPS (The European Joint Conferences on Theory and Practice of Software), 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, data security, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

The Bulletin of the EATCS

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
- Lisbon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008

(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) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the *EATCS Texts* and the *EATCS Monographs* published by Springer-Verlag and launched during ICALP in 1984. The Springer series include *monographs* covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as *texts* intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the series are W. Brauer (Munich), J. Hromkovic (Aachen), G. Rozenberg (Leiden), and A. Salomaa (Turku). Potential authors should 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.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. 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. The Editors-in-Chief of the journal currently are G. Ausiello (Rome), D. Sannella (Edinburgh), G. Rozenberg (Leiden), and M.W. Mislove (Tulane).

ADDITIONAL EATCS INFORMATION

For further information please visit <http://www.eatcs.org>, or contact the President of EATCS:

*Prof. Dr. Giorgio Ausiello, Dipartimento di Informatica e Sistemistica
Universita di Roma "La Sapienza", Via Salaria 113, 00198 Rome, ITALY
Email: president@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, Jules Moretuslei 229, B-2610 Wilrijk, Belgium
Account number: 220-0596350-30-01130
IBAN code: BE 15 2200 5963 5030, SWIFT code: GEBABE BB 18A*