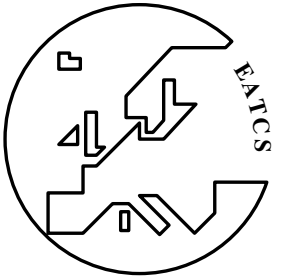
ISSN 0252–9742

Bulletin

of the

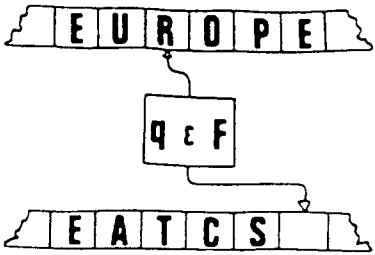
European Association for Theoretical Computer Science

EATCS



Number 87 October 2005

Council of the European Association for Theoretical Computer Science



Board:

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

Other Council Members:

Pierpaolo Degano	Italy	Juhani Karhumäki	Finland
M. Dezani-Ciancaglini	Italy	David Peleg	Israel
Josep Díaz	Spain	Jiří Sgall	Czech Republic
Zoltán Ésik	Hungary	Andrzej Tarlecki	Poland
Javier Esparza	Germany	Wolfgang Thomas	Germany
Hal Gabow	USA	Dorothea Wagner	Germany
Alan Gibbons	UK	Emo Welzl	Switzerland
Kazuo Iwama	Japan	Gerhard Wöeginger	The Netherlands
Jean-Pierre Jouannaud	France	Uri Zwick	Israel

EATCS Monographs and TCS:

Monographs Editors:	Wilfried Brauer	Germany
	Grzegorz Rozenberg	The Netherlands
	Arto Salomaa	Finland
TCS Editors:	Giorgio Ausiello	Italy
	Michael Mislove	USA
	Don Sannella	United Kingdom

Past Presidents:

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

EATCS Council Members

EMAIL ADDRESSES

Giorgio Ausiello . . . . . ausiello@dis.uniroma1.it
Wilfried Brauer . . . . . brauer@informatik.tu-muenchen.de
Pierpaolo Degano . . . . . degano@di.unipi.it
Mariangiola Dezani-Ciancaglini . . . . . dezani@di.unito.it
Josep Díaz . . . . . diaz@lsi.upc.es
Zoltán Ésik . . . . . ze@inf.u-szeged.hu
Javier Esparza . . . . . esparza@informatik.uni-stuttgart.de
Hal Gabow . . . . . hal@research.cs.colorado.edu
Alan Gibbons . . . . . amg@dcs.kcl.ac.uk
Kazuo Iwama . . . . . iwama@kuis.kyoto-u.ac.jp
Dirk Janssens . . . . . Dirk.Janssens@ua.ac.be
Jean-Pierre Jouannaud . . . . . jouannaud@lix.polytechnique.fr
Juhani Karhumäki . . . . . karhumak@cs.utu.fi
Jan van Leeuwen . . . . . jan@cs.uu.nl
Michael Mislove . . . . . mwm@math.tulane.edu
Mogens Nielsen . . . . . mn@brics.dk
David Peleg . . . . . peleg@wisdom.weizmann.ac.il
Jiří Sgall . . . . . sgall@math.cas.cz
Branislav Rovan . . . . . rovan@fmph.uniba.sk
Grzegorz Rozenberg . . . . . rozenber@liacs.nl
Arto Salomaa . . . . . asalomaa@utu.fi
Don Sannella . . . . . dts@dcs.ed.ac.uk
Vladimiro Sassone . . . . . vs@susx.ac.uk
Paul Spirakis . . . . . spirakis@cti.gr
Andrzej Tarlecki . . . . . tarlecki@mimuw.edu.pl
Wolfgang Thomas . . . . . thomas@informatik.rwth-aachen.de
Dorothea Wagner . . . . . Dorothea.Wagner@uni-konstanz.de
Emo Welzl . . . . . emo@inf.ethz.ch
Gerhard Wöeginger . . . . . g.j.woeginger@math.utwente.nl
Uri Zwick . . . . . zwick@post.tau.ac.il

Bulletin Editor: Vladimiro Sassone, Sussex, BN1 9QH, United Kingdom
Cartoons: DADARA, Amsterdam, The Netherlands

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

All contributions are to be sent electronically to

bulletin@eatcs.org

and must be prepared in LaTeX 2_ε using the class `beatcs.cls` (a version of the standard LaTeX 2_ε `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.

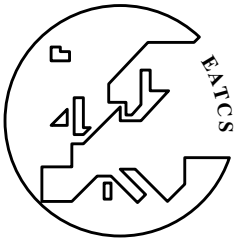
The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

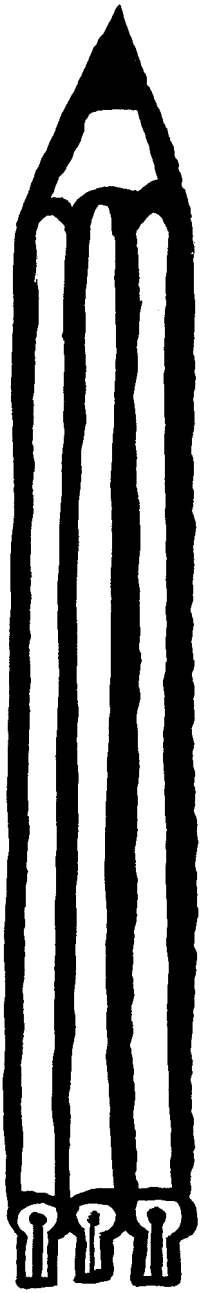
Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

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.

Table of Contents

EATCS MATTERS





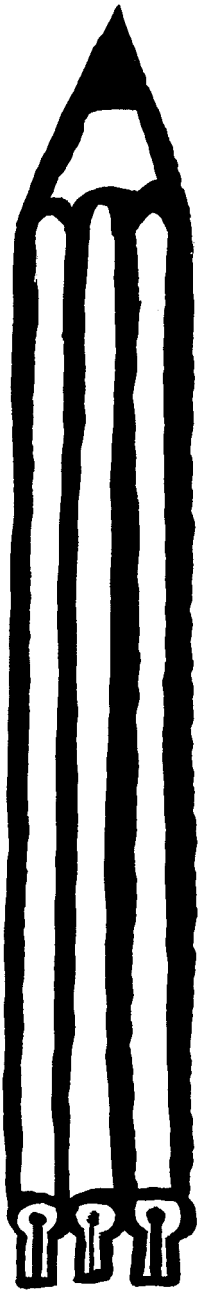
Letter from the President

Dear EATCS members,

Following my letter in the previous Bulletin, things are now developing rapidly towards shaping the European Union's 7th Framework, FP7, and again I would like to encourage all EATCS members to take an active part in the national as well as international discussions. The current proposal from the European Commission has just been released, and can be found from www.cordis.lu/fp7. In particular, the developments towards a European Research Council to support basic, frontier research can be followed from the so-called European Basic Research Website, to which you may find a link on www.cordis.lu/fp7/ideas.htm.

At the time of writing, the election of 10 members for EATCS Council is still running. On behalf of EATCS, I would like to thank all the candidates for accepting their nominations for the election, and to welcome the new members of the Council. Also, I would like to thank the previous members now leaving the Council for their work through the years. Their contributions have been highly appreciated, and I hope that the association may rely on their assistance also in the future.

As many of you will know, we had a very successful 32nd ICALP in Lisbon this year. We are grateful to Luis Monteiro, Luis Caires and their colleagues in Lisbon for an excellent organization. ICALP was co-located with 7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming PPDP'05 as well as a number of workshops and other events, including the presentation of the



EATCS Award 2005. You may find many more details in this issue of the Bulletin.

Let me also draw your attention to the calls for papers and workshops for ICALP 2006 in Venice, organized by Michele Bugliesi. As you will see, we continue in 2006 with the successful format of three track introduced in 2005. You may find more details in this Bulletin, where you will also find calls for nominations for the Gödel prize 2006 and the EATCS Award 2006.

At the General Assembly in Lisbon, it was unanimously decided to have ICALP 2007 in Poland, more precisely in Wroclaw, organized by Leszek Pacholski. Also, the General Assembly approved a proposal to modernize the EATCS statutes, a proposal which will subsequently be sent for final approval by all EATCS members. More on this in the next issue of the Bulletin.

Mogens Nielsen, Aarhus
September 2005

Letter from the Bulletin Editor

Dear Reader,

*Welcome to the October 2005 issue of the Bulletin of the EATCS. Like all Autumn issues, this one includes typical 'post-ICALP' items, such as the Secretary's report from the EATCS General Assembly and Manfred Kudlek's punctual reportage from Lisbon of the 32nd ICALP conference. ICALP 2005 witnessed the presentation of the EATCS Award for lifetime distinguished achievements to Robin Milner, whose reflections after his Award Lecture are collected here in the paper "*Software Science: From Virtual to Reality.*" Remarkable too is Cristian Calude's interview to Solomon Marcus, member of the Romanian Academy, reported in Cristian's column, "*News from New Zealand.*"*

*This instalment of the Bulletin contains the last of Josep Díaz's "*Algorithmic Column*"s. Josep has edited the column for four years, summing up to 12 issues, and I wish to thank him very warmly indeed for the fantastic job he has done. The new editor of the "*Algorithmic Column*," Gerhard Woeginger, will take charge from issue 88 in February.*

The set of regular columns and contributed papers is as rich and interesting as ever, and I am glad to submit it to your attention. As usual the volume is closed by a set of reports from international conferences and meetings, and by abstract of PhD dissertations.

Enjoy

Vladimiro Sassone, Sussex
September 2005

ICALP 2005

Report on the EATCS General Assembly 2005

The 2005 General Assembly of EATCS took place on Tuesday, July 12^{th}, 2005, at the Gulbenkian Foundation in Lisbon, the site of the ICALP. President Mogens Nielsen opened the General Assembly (GA) at 18:07. He started out by announcing that due to organisational constraints the GA had been allocated one hour strictly, and hence apologised that the agenda was slightly shorter than usual. The agenda consisted of the following items.

Report of the EATCS President. Mogens Nielsen reported briefly on the EATCS activities between ICALP 2004 and ICALP 2005. He referred to the more detailed report posted a couple of weeks before the GA on the EATCS web page at `www.eatcs.org`. Mogens Nielsen explicitly mentioned and emphasised several items.

The number of members of EATCS had continued to increase, due to a range of activities, including a new web based membership registration system. Mogens Nielsen encouraged all members to update their membership information regularly (from `www.eatcs.org`).

Also, the financial situation of EATCS had continued to improve, mainly due to efforts of the editor of the Bulletin of the EATCS, Vladimiro Sassone, resulting in a significant reduction of the production cost, as well as last years decision to increase the annual EATCS membership fee to €30. Mogens Nielsen concluded that the improved financial situation would give room for some new EATCS initiatives currently under discussion in the EATCS Council, and encouraged all members to contribute to this discussion by contacting Council members.

The president reported on the new composition of the award committees. The Gödel Prize committee 2006 consists of P.-L. Curien (chair), P. Vitanyi, and V. Diekert representing EATCS, and C. Papadimitriou, J. Reif, and J. Ullman representing ACM SIGACT. The EATCS Award 2006 committee consists of M. Dezani-Ciancaglini (chair), D. Peleg, and W. Thomas.

Mogens Nielsen also reported on a Council decision to keep the successful structure of ICALP 2005 with the three tracks A (*Algorithms, Automata, Complexity and Games*), B (*Logic, Semantics and Theory of Programming*), and C (*Security and Cryptography Foundations*), also for ICALP 2006.

In the reporting period a total of 21 events were under the auspices of EATCS, and EATCS sponsored a number of prizes for the best papers or best student papers at conferences (ICALP, ETAPS, ESA ICGT, and MFCS), sponsor conferences and acknowledges activity of its chapters. More details in the report on the web.

Mogens Nielsen also included brief reports from the EATCS associated publications, again referring to the annual report for details.

In the reporting period, three volumes of the Bulletin of the EATCS has been published, and special thanks and appreciation were given to the editor, V. Sassone, for his efforts in continuously improving the quality of the Bulletin, and at the same time reducing the cost. A number of recent Bulletin issues are now available electronically for EATCS members. On behalf of the editor he also thanked the Column editors, News editors, and everybody else contributing to the success of the Bulletin.

In the EATCS Texts and Monographs series, a total of 7 Texts had been published in the reporting period. Special thanks were given to the editors A. Salomaa, W. Brauer, and G. Rozenberg and to Springer Verlag, and the new Springer editor for the series, Ronan Nugent, was given a warm welcome and introduced to the audience.

In the journal Theoretical Computer Science, TCS, volumes 322 to 340 had been published. Special thanks were given to the editors in chief G. Ausiello, D. Sannella, and M. Mislove (ENTCS editor). Also, the special celebration of the 30th anniversary of TCS during ICALP 2005 was mentioned, and the new Elsevier TCS editor, Christopher Leonard, was given a warm welcome and introduced to the audience.

Ratification of Council Election. According to the EATCS Statutes, 10 members of the EATCS Council are elected in every odd year for a four year period. A call for nominations had been posted in the Bulletin of the EATCS as well as on the EATCS website. Following this a total of 23 candidates had been nominated by EATCS members. The names of these were presented to the GA, and the list was approved with no additions. (Subsequently one of the candidates withdrew the candidacy, and hence 22 candidates ran in the electronic election taking place in September 2005.)

Proposal of Revised EATCS Statutes. Following the announcement at last years GA, a concrete proposal for revisions of the current EATCS Statutes was presented to the GA. The purpose of the revision is mainly to modernise the formation of the Council (by removing references to explicit publications and the notion of a Board), to clarify some ambiguities (e.g., the formulation of the nationality constraint in the formation of the Council), to remove some unfortunate restrictions (e.g., the inflexibility of timing constraint on Council elections, which fall in the holiday season), and to correct some small inconsistencies.

The proposal was approved by the GA. Furthermore, the president announced that additional revisions of a legal nature could result from current consultations

with legal advisers. The GA gave the Council the authority to formulate a complete set of new Statutes following the proposal approved, to be presented to all members for final approval following the formulations in the current Statutes.

Report ICALP 2005. Luis Caires gave a report on the local arrangements for ICALP 2005, on behalf of himself and the other chair of the organising committee, Luis Monteiro. ICALP 2005 was co-located with the 7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming PPDP'05, and 8 pre/post-conference workshops.

Luis Caires provided some basic statistics on the organisation. A total of 255 participants from 29 countries registered for ICALP, and including the workshops there were 384 participants from 32 countries. Also, Luis Caires thanked explicitly all the sponsors of ICALP 2005.

It was for the first time ICALP was located in Portugal, and the GA expressed its appreciation for a superb organisation.

ICALP 2005 introduced for the first time three tracks with separate program committees. Besides the traditional tracks A (Algorithms, Automata, Complexity and Games) and B (Logic, Semantics and Theory of Programming), an additional track C on Security and Cryptography Foundations was introduced.

The three PC chairs Pino Italiano (track A), Catuscia Palamidessi (track B), and Moti Yung (track C) gave separate reports. There were a record number of 407 submissions for ICALP (258 for track A, 75 for track B, 74 for track C), out of which 113 were accepted for the conference. The three chairs provided many more statistical details of their work, some of which will appear in the usual ICALP report contributed to this volume by Manfred Kudlek. Again, the GA expressed its appreciation for their excellent work.

The President kept the tradition presenting the ICALP organisers and the PC chairs with small gifts, thanking all of them for their efforts.

Report ICALP 2006. On behalf of Michele Bugliesi, the ICALP 2006 organising chair, Vladimiro Sassone reported on the organisation of ICALP 2006 to be held in Venice on July 9–16, 2006. ICALP 2006 will follow the successful format from 2005 with the three tracks A (chaired by Ingo Wegener), B (chaired by Vladimiro Sassone), and C (chaired by Bart Preneel). The conference venue will be the island of San Servolo in Venice, and Vladimiro provided detailed information on the city of Venice, San Servolo, accommodation facilities etc.

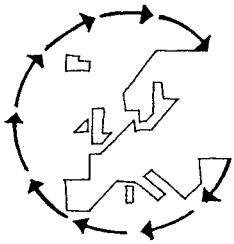
Venue for ICALP 2007. Mogens Nielsen announced that several sites had expressed an interest in hosting ICALP in 2007, but that in the end, he was only aware of one contender, University of Wroclaw, Poland, with Leszek Pacholski as

conference chair. When nobody from those present brought up another proposal, Leszek Pacholski presented his proposal of organising ICALP 2007 in the period July 9–13, including basic information about the University of Wroclaw, the city of Wroclaw, accommodation facilities, etc., Furthermore it was announced that ICALP 2007 would be co-located with other conferences, including the Annual IEEE Symposium on Logic in Computer Science, LICS 2007, following the successful co-location in Turku 2004.

The GA approved unanimously Wroclaw as the site for ICALP 2007.

SPECIALS. At this point, at 19:11, the President thanked all present and concluded the 2005 General Assembly of the EATCS by introducing Manfred Kudlek, presenting the statistics of the authors who published repeatedly at ICALP, and presenting the special EATCS badges to those having reached 5 or more full papers at ICALP. By tradition Manfred also presented the EATCS badges to the editors of the ICALP 2005 proceedings.

Branislav Rovan and Mogens Nielsen



The Distinguished Achievements Award

EATCS AWARD 2005

In a special afternoon ceremony on July 14th during ICALP 2005 at the Calouste Gulbenkian Foundation in Lisbon, the EATCS Distinguished Achievements Award 2005 was awarded to

Professor Robin Milner

emeritus professor at the University of Cambridge (UK), for his outstanding contributions to theoretical computer science. After a brief introduction by the EATCS president, Mogens Nielsen, the EATCS Award was presented to Robin by the chairman of the Award Committee 2005, Jan van Leeuwen. The other members of the Committee were: Mariangiola Dezani and Wolfgang Thomas.

Robin Milner has been a cornerstone of the mathematical theory of computation for many years. The first beginnings of his work date back to the 1960's when, in positions at the City University of London and the University College in Swansea, he became deeply interested in rigorous methods for the analysis of computer programs, using the tools of mathematical logic. During a very influential period at Stanford University from 1971 to 1973, Robin began his work on computer-assisted reasoning, initially with the further development of LCF, a deductive system for computable functions based on ideas of Dana Scott.

Following up on this in subsequent years, Robin embarked on his first major project, the design and definition of the language ML (*Meta-Language*), a now widely accepted programming language with polymorphic type inference and type-safe exception handling, and with a fully formal semantics. ML, or rather its later version *Standard ML*, is taught to students even at the undergraduate level now and it has found applications in industry.

Already in the 1970's Robin saw the classical paradigms of computer science shift towards concurrent and interactive processes. Aiming at a fully mathematical framework again, he developed CCS, the *Calculus of Communicating Systems*, one of the first algebraic calculi for analysing concurrent systems. He also formulated and substantially advanced *full abstraction*, which concerns the intimate relationships between operational and denotational semantics. In the 1990's

Robin developed CCS into the π-calculus, the highly influential process calculus which became a widely accepted basis for modeling and reasoning about concurrent systems and which has found applications even in the modeling of business processes. More recently Robin has invented new frameworks for the analysis of distributed systems, notably *action calculi* and *bigraphical structures*, now including both mobility and reactivity together.

The mathematical theory of computation is devoted to the formal understanding of computation, of programs and programming and, as Zohar Manna wrote many years ago, to making the verification of programs into a *science*. The notions of programs and programming have evolved greatly over the years. Our summary of Robin's work only sketches his very important role in the field and its development into a *mathematical theory of interaction*. Robin's outstanding influence is best illustrated by the fact that he is the fourth most cited author in computer science in the world (cf. the CiteSeer.IST index of most cited authors, May 2005).

A key feature of Robin's work has always been the balance between theory and practice. Whereas he has always strived for fully mathematical frameworks, he always ensured that his research contributions were both relevant and fundamental. His books (on CCS, on ML and Standard ML, and on Communicating and Mobile Systems) are landmarks in their field. Many people have enjoyed working with Robin, and in most cases the work we mentioned was done in long term projects with PhD students and with fellow researchers, in Edinburgh where he worked from 1973 till 1995, in Cambridge after that, or at other places like Aarhus where Robin held a guest professorship. Robin has also been a leading scientist in the movement in the UK to establish long-term research goals for computer science, including the definition of Grand Challenges for our field.

Robin Milner received several prizes for his work. In 1987 he received the British Computer Society Technical Award (for ML) and 1991 the prestigious Turing Award (for ML and CCS). In 2001 he received the ACM SIGPLAN Award for Achievement in Programming Languages. Robin received at least 7 honorary doctorates (including the one at Bologna during ICALP'97). He is a Fellow of ACM, of the Royal Society and of the Royal Society of Edinburgh. In 2004 he received the Royal Medal, the highest award of the Royal Society of Edinburgh. Robin served on many committees in the UK and in other countries, and on several editorial boards, including e.g. TCS, the Computer Journal and Mathematical Structures in Computer Science. Last but not least, he also has been active for EATCS: he was a member of the EATCS Council for many years from 1973 onward, he was one of the organizers of ICALP'76 (Edinburgh), and a program committee member of ICALP'80 (Noordwijkerhout).

Jan van Leeuwen

SOFTWARE SCIENCE: FROM VIRTUAL TO REALITY
EATCS AWARD LECTURE

Robin Milner

FROM VIRTUAL...

I have spent a career working in what has been called "programming languages" and "semantics." It may seem ungrateful to question the way we use these terms, and whether they properly describe a branch of computation theory. But there are occasions which grant a licence to be provocative; I assume that this is such an occasion.

Modern computing, both hardware and software, is firmly rooted in the Church-Turing hypothesis, and in the physical possibility of a universal machine in each of the computational models that the hypothesis asserts to be equivalent. The most famous such model is what Turing called his 'paper machines;' another, developed by Minsky and others, is based upon register machines. In the first phase of modern computing there was no real issue about language; the early assembly codes, the autocodes and even FORTRAN were rather thinly disguised register machines; there was little doubt about the meaning of a program. This was the Garden of Eden, where people did not expect or feel the need for advanced forms of abstraction to describe a computation in more succinct or modular fashion. Somewhere John Backus expressed his satisfaction that the definition of FORTRAN occupied a mere 51 pages; most of them were presumably about grammar.

The departure from the Garden of Eden was gradual but relentless. The increasing distance from the Garden can be measured by the increasing gap between, on the one hand languages (even, in the early days, ALGOL 60), and on the other hand the mathematically powerful abstractions defining entities that programs should denote. True, there has been much mutual influence between language design and abstractions; the dialectic has borne great fruit for both practice and theory. But we have never re-entered the Garden. To regain entry we should have achieved a mathematical model of computation, perhaps highly abstract in contrast with the concrete nature of paper and register machines, but such that programming languages are merely executable fragments of the theory and require no independent invention. The gate back to the Garden seems inaccessible. In teach-

ing programming languages to advanced undergraduates I always said that every language design went one step (or more) beyond the theories available to explain it. (I know this well, as a language designer.) Of course, this fuelled many healthy strands of semantic research.

But is software science only about language and semantics, or is there something else?

. . . TO REALITY

A new strand entered, gradually, some forty years ago. As well as seeking abstractions to express the virtual structures of programs, we began to seek them also to model the real structures of communication and interaction. The word "real" is justified as soon as we accept that these phenomena exist independently of computers and programs. Equally, the word "model" is justified, in preference to "semantics." The latter term tends to presuppose a language to be explained; "model," on the other hand, presupposes no such thing. We began to model informatic phenomena that are independent of language. Petri, one of the first to do so, was explicitly thinking of the processes underlying office systems, as well as processes even less dependent on intelligent agents. About twenty years ago a leading theorist remarked that, with the study of concurrent informatic processes, computer science moved for the first time beyond what logicians might have expected; it created its own subject. Of course, many still prefer to talk of semantics rather than modelling; I caused a grumbling stir at a semantics conference fifteen years ago when I suggested we should emulate "natural" science by introducing the term "infodynamics" for what we do. . .

ARTIFICIAL OR NATURAL?

At any rate, "informatics" (which we can think of as the study of the activity of communicating, or informing), is an accurate term to describe computer science in this newer and broader sense. It is a science of real as well as virtual phenomena. To what extent are these real phenomena artificial? – i.e. to what extent is this new computing a "science of the artificial," in Herbert Simon's terms?

This question is not a matter for idle curiosity. As Simon points out, there is almost a contradiction in the notion of a science of artefacts: the very understanding we gain from the study is likely to affect the next generation of artefacts, and in turn the science itself. This surely happens in computing. We began our theories of interactive systems partly in response to early experience in computer networking. Understanding thus gained surely influenced the emergence of new

phenomena such as the Worldwide Web. We begin now to consider populations of autonomous entities, whether they be the "agents" of agent technologies or physical entities such as sensors; and theories of mobile processes become important because these entities, each in their own way, move in a virtual or physical space. The way we manage pervasive computing, and eventually manage the populations of "nanobots" that will emerge from nanotechnology, will be influenced by these theories.

CONVERGENCE

Or will they? This progression of technology and the theoretical response to it forces us to admit that informatics lacks the simple dignity of a "natural" science; this dignity, in physics or in biology, comes from the fact that – at least hitherto – the subject matter of the science remains fixed as an object of study. Having admitted this lack, do we then consider informatics as defective? A possible response is much more positive: the fact that informatic science moulds its own subject matter – for example the artefacts of pervasive computing – gives it a unique status among sciences. But this status is only justified if the science and its technologies find a way to develop in a tight feedback loop.

Of course, this is exactly what has not happened in conventional computing. Current software engineering and computing theories are too far apart; the phenomenon of inscrutable legacy software bears witness to their failure to maintain connection. Why, and how, can we succeed to maintain connection between software technologies and theories for the far more complex world of pervasive computing?

Part of the answer to this question must be that the penalties for failure are dramatically higher; the more pervasive a system, the more intimately its failures will affect the lives of people, and the more difficult it becomes to mend, adapt or replace the system in situ. In explaining this prognosis to society, we shall surely meet with the demand for a far higher level of scientific ratification of any proposed design. The other part of the answer will be to develop the terms of our science so that this ratification can be expressed in a form that is accepted by our society. This acceptance already exists for many engineering technologies (e.g. building); not that each member of society understands each technical argument, but that the scientific terms in which those arguments are couched are well rooted in our culture.

Science, and its well-rooting in our culture, is notably absent for the kind of pervasive software architectures that lie within our reach. To achieve it seems to require a fresh approach to the development of theories. We can already start from an impressive platform of calculi, logics and associated analytical tools; these

have mostly been built with Occam's razor in mind, and their very economy makes them tractable. But large pervasive applications will only be properly understood in terms of theories that deal with a much richer variety of concepts, including trust, reflectivity, mobility, knowledge, belief, purpose, ... (all as attributes of entity populations or the entities themselves). What is needed is a tower of theories of these concepts; the higher theories in the tower are concerned with the more complex concepts, which are explicitly "realised" or "implemented" in terms of those in the lower theories.

To achieve this tower, in turn, requires theorists and designers to collaborate in experimental projects that distil these concepts one-by-one. Such projects would have a dual purpose; for example, a project to experiment with an architecture for driver-less traffic management would aim not only to experiment with a specific design but also to express the properties of that design in theoretical terms that permit analysis. This kind of project is ambitious, but instances of it already exist. For example, platforms for conducting business processes, and languages for expressing the platforms, are already under study. And in the world of natural science, biologists and computer scientists are collaborating in building models drawing upon – and developing – existing informatic theories.

What is software?

If this trend is continued and amplified, success can be measured in terms of a new conception of software. It will come to mean, not just programs in a language defined by a design committee, but something much broader: the behavioural description of a system in terms of a stable informatic science, using concepts at an appropriate level of its theory-tower. This includes specification (at many conceptual levels) as well as implementation; the term "program" will be applied to levels of description low enough to be realised by machine. And the step from specification to program, or from high-level to low-level program, becomes just one instance of realisation of a higher-level theory by a lower-level one.

Is this a new meaning for "software"? Not entirely; after all, logic programming was a step in this direction. And it can been seen as re-entry to the Garden of Eden: "semantics" need longer be retrofitted to programming languages because they are already a part of an understood theory. But it is much more than that. For although artefacts and theories will continue to evolve, as befits an artificial discipline, their evolution will be stabilised by tight feedback from one to the other.

EATCS AWARD 2006

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 recognised contributions to theoretical computer science over a life long scientific career.

For the EATCS Award 2006, 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, 2005**.

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

Professor Mariangiola Dezani-Ciancaglini
Dipartimento di Informatica
Universita' di Torino
c. Svizzera 185, 10149 Torino (Italy)

Email: dezani@di.unito.it

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)

The next award is to be presented during ICALP'2006 in Venice.

Gödel Prize 2006

Call for Nominations

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 fourteenth presentation will take place during ICALP 2006, Venezia, July 10–14, 2006. 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 $5000 (US).

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 2006 Chair being an EATCS representative. The 2006 Award Committee consists of Pierre-Louis Curien (chair, CNRS, Université Paris 7), Volker Diekert (Universität Stuttgart), Christos Papadimitriou (UC Berkeley) John Reif (Duke University), Jeff Ullman (Stanford University), and Paul Vitanyi (CWI, Amsterdam).

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 1993. 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 $n = 2006$) then the following constraints should be respected:

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

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:

Pierre-Louis Curien
PPS case 7014, Université Paris 7
2 Place Jussieu, 75251 Paris Cedex 05, France
email: curien@pps.jussieu.fr
tel: (+33) 144277095 fax: (+33) 144278654

To be considered, nominations for the 2006 prize must be received by January 31, 2006. 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.

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 "Goedel06."

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:

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.

1994: Johan Håstad, "Almost optimal lower bounds for small depth circuits," *Advances in Computing Research* **5** (1989), 143–170.

1995: Neil Immerman, "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.

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.

1997: Joseph Halpern and Yoram Moses, "Knowledge and common knowledge in a distributed environment," *Journal of the ACM* **37** (1990), 549–587.

1998: Seinosuke Toda, "PP is as hard as the polynomial-time hierarchy," *SIAM Journal on Computing* **20** (1991), 865–877.

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.

2000: Moshe Y. Vardi and Pierre Wolper, "Reasoning about infinite computations," *Information and Computation* **115** (1994), 1–37.

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.

2002: Géraud Sénizergues, "$L(A) = L(B)$? Decidability results from complete formal systems," *Theoretical Computer Science* **251** (2001), 1–166.

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.

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.

2005: Noga Alon, Yossi Matias and Mario Szegedy, "The space complexity of approximating the frequency moments," *Journal of Computer and System Sciences,* **58** (1999), 137–147.

October 1, 2006
Comments to curien@pps.jussieu.fr

Report from the Japanese Chapter

K. Makino (Univ. of Tokyo)

EATCS-JP/LA Workshop on TCS

The *fourth EATCS/LA Workshop on Theoretical Computer Science* will be held at Research Institute of Mathematical Sciences, Kyoto Univ., January 30th ~ February 1st, 2005. 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 listeing 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 !

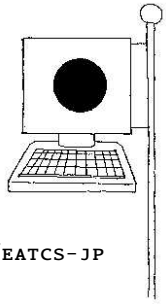
On TCS Related Activities in Japan:

TGCOMP Meetings, January ~ June, 2005

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 2005, *TGCOMP* organized 5 meetings and about 54 papers (including one tutorial) were presented there. Topics presented are, very roughly, classified as follows.

Algorithm: On Graphs (12)	Computational Learning (1)
Algorithm: On Strings (2)	Cryptography (2)
Algorithm: On-line Algorithms (4)	Distributed Computing (3)
Algorithm: On Other Objects (10)	Formal Languages and Automata (6)
Combinatorics / Probabilistic Analysis (4)	Quantum Computing (2)
Computational Complexity (7)	Semantics and Term Rewriting System (1)

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



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

Institutional Sponsors

BRICS, Basic Research in Computer Science,
Aarhus, Denmark

Elsevier Science
Amsterdam, The Netherlands

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

Microsoft Research,
Cambridge, United Kingdom

PWS, Publishing Company,
Boston, USA

TUCS, Turku Center for Computer Science,
Turku, Finland

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

http://www.win.tue.nl/ipa

INSTITUTE FOR PROGRAMMING RESEARCH AND ALGORITHMICS

Coming events

IPA **Herfstdagen** on **Security**
November 21-25, 2005, Hotel Zwartewater, Zwartsluis, The Netherlands.
The IPA Herfstdagen are an annual five day event, dedicated to one of IPA's current main application areas: Networked Embedded Systems, Security, Intelligent Algorithms, and Compositional Programming Methods.

In 2001, IPA dedicated an event to Security, when during the Lentedagen all researchers interested in Security came together for the first time. Since that time, research in the area has grown substantially both within IPA and in the Netherlands in general. The goal of the Herfstdagen is to provide an overview of the current stage of security research in and around IPA. Besides talks on current work in computer science on security of data, software, networks, and communication, attention will be paid to organisational and societal aspects of security. More information will become available through the Herfstdagen webpage.
See: `www.win.tue.nl/ipa/archive/falldays2005/`

IPA sponsors **FMCO 2005**
November 1 - 4, 2005, CWI, Amsterdam, The Netherlands.
The objective of this fourth international symposium on Formal Methods for Components and Objects is to bring together top researchers in the area of software engineering to discuss the state-of-the-art and future applications of formal methods in the development of large component-based and object oriented software systems. To encourage the participation of Ph.D. students, tutorials on central topics in the area are included in the program. Key-note speakers are Michael Barnett (Microsoft), Luis Caires (Lisbon), Patrick Cousot (ENS), Dennis Dams (Bell Labs), Wan Fokkink (VU Amsterdam), Orna Grumberg (Technion), Joost-Pieter Katoen (RWTH Aachen), Kung-Kiu Lau (Manchester), Peter O Hearn (Queen Mary, London), Arnd Poetzsch-Heffer (Kaiserslautern), John Reynolds (Carnegie

Mellon), and Davide Sangiorgi (Bologna).
See: fmco.liacs.nl/fmco05.htm

Recent events

IPA **Lentedagen** on **Software Architecture**
March 30 - April 1, 2005, hotel De Korenbeurs, Made, The Netherlands.
Software architecture is a topic that has figured in a low-key but consistent way throughout IPA events in the past years. After having looked at component-based architectures (Herfstdagen 1999), object-oriented architectures (Lentedagen on UML, 2000), and peer-to-peer systems (Lentedagen on Middelware, 2002), this year's Lentedagen had software architecture as the main topic, with the aim of presenting an overview of recent developments in the field that are of interest to the IPA community. The overall theme was architecture and change: dealing with change during design, the design of systems that change, and analysis of the architecture of existing systems (with the aim of changing them). The program contained sessions on variability, model driven and service oriented architectures, and the reconstruction and assessment of architectures. Abstracts, hand-outs and papers are available through the Lentedagen webpage.
See: www.win.tue.nl/ipa/activities/springdays2005/

IPA supports the **Security PhD Association Netherlands**
May 27, 2005, Delft University of Technology, The Netherlands.
After organising a number of meetings under the wings of the SAFE-NL platform (Security: Applications, Formal aspects and Environments in the NetherLands), a group of Ph.D. students from different Dutch universities started SPAN (the Security Ph.D. Association Netherlands) an open platform for all Ph.D. students in the Netherlands that are interested in Security. IPA supports this initiative, and sponsored the first national SPAN meeting in May.
See: www.win.tue.nl/span/

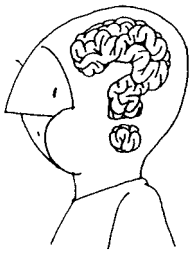
Addresses

Visiting address
Technische Universiteit Eindhoven
Main Building HG 7.22
Den Dolech 2
5612 AZ Eindhoven
The Netherlands

Postal address
IPA, Fac. of Math. and Comp. Sci.
Technische Universiteit Eindhoven
P.O. Box 513
5600 MB Eindhoven
The Netherlands

tel (+31)-40-2474124 (IPA Secretariat)
fax (+31)-40-2475361
e-mail ipa@tue.nl, url http://www.win.tue.nl/ipa

EATCS News



News from Australia

BY

C.J. Fidge

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

In February's column I summarised some of the financial woes facing Computer Science and Information Technology research in Australia. This included last year's dramatic shutdown of *Motorola Research Labs Australia* in Sydney and the gradual closure of the University of Queensland's *Software Verification Research Centre*. Also, *Ericsson Research Labs* in Melbourne laid off 300 engineers in 2003. Unfortunately, there is more bad news to report this time.

In that earlier column I mentioned the uncertain future facing the *Distributed Systems Technology Centre* (DSTC), one of the country's major Information and Communications Technology research organisations. Indeed, it was confirmed recently that the centre is 'winding up' its operations. Although the DSTC's web site puts the best possible spin on the situation (`http://www.dstc.edu.au/news1.html`), there is no denying that this is another severe blow for computer science research in Australia.

Sadly, despite figures showing that job vacancies in the ICT sector are growing, this has not yet changed academia's fortunes. Throughout the country falling enrolments mean that Computer Science and Information Technology departments in universities have been shedding academic staff. Newspaper reports cite declines in enrolments of between 40 and 50 percent.

Two especially telling examples are major redundancies of computing academics at both *Monash University* in Melbourne and *The University of New South Wales* in Sydney. Both of these universities have long been leaders in Australian

computer science teaching and research. Despite this, Monash University was recently forced to let 22 academics go. Similarly dramatic staff reductions have occurred at universities throughout Australia. (My own Queensland University of Technology has not been immune!)

Most shocking of all was the announcement in July by the privately-owned *Bond University* that it was going to close its Faculty of Information Technology entirely. The official story is that IT is to be merged with the Faculty of Business to create a single 'super faculty' but this still means the loss of academic staff from the IT side of the deal. What made this decision particularly startling was that Information Technology was one of the cornerstones of the university when it was first established not much more than a decade ago.

On top of this market downturn is the conservative federal government's dogmatic determination to make universities run like commercial businesses, with an emphasis on churning out graduates quickly and only doing research that produces short-term financial benefits. Not surprisingly, this situation does not help the cause of Theoretical Computer Science research in this country.

The bad news can't continue indefinitely, however. As Information Technology Deans around the country keep pointing out, IT is not going away and it needs to be supported by quality teaching and research. However, it is difficult to escape the impression that the landscape for computing research in Australia, especially on the theoretical side, is changing irrevocably.

News from India

BY

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

In this edition of News from India, we carry a report on the *Formal Methods Update 05* workshop held at IIT Bombay in July 2005 and look ahead to forthcoming events.

Formal Methods Update 05. *Formal Methods Update 05* was a three day workshop held at IIT Bombay on July 18–20, 2005. The workshop was organized by the Centre for Formal Design and Verification of Software (CFDVS) with support from the Indian Association for Research in Computing Science (IARCS) and the Department of Science and Technology, Government of India (DST). This was followed by a two day meeting of the working group on *Logic in Computer Science* of the Centre for Discrete Mathematics and it Applications (CARDMATH), recently established by DST.

This was the fourth in a series of "update meetings" that have been organized over the past 3–4 years. These meetings are intended as a forum for Indian researchers and students working in formal methods and theoretical topics in computer science to update themselves on current trends and to explore new research areas. In keeping with this goal, most presentations are technical surveys and advanced tutorials. The previous three meetings were held in IMSc, Chennai with the themes *Models for Programs, Timed Systems* and *Automata and Verification*.

The theme of this fourth update meeting was broadly *Advances in Concurrency, Logic and Verification*. There was a special focus on "Advanced Formal Methods" to explore the interplay of theoretical results from Automata, Concurrency and Logic with the advanced techniques for Specification and Verification of Programs. The talks presented covered, among other topics, verification techniques for hybrid systems, formalization computer security policies, verification of message-passing systems and theories of software testing.

For copies of some of the presentations and other details about the meeting, look up http://www.cfdvs.iitb.ac.in/~meeting05.

Forthcoming workshops. Two more workshops are being organized in October.

- A two day workshop on *Formal Methods for Design and Analysis of Software* is being jointly organized by Microsoft Research India, Indian Institute of Science and IARCS at Bangalore on October 7–8, 2005. Details are available at http://research.microsoft.com/~sriram/RSE/workshop.html
- A four day workshop on *Approximation Algorithms* is being organized at IIT Delhi on October 8–11, 2005 by Naveen Garg and Amit Kumar. Details are available at the URL http://www.cse.iitd.ernet.in/~naveen/approxwshp.htm.

FSTTCS 2005. FSTTCS 2005 will take place from December 15–18, 2005 at the International Institute of Information Technology, Hyderabad. The Programme Committee is chaired by R Ramanujam (IMSc, Chennai) and Sandeep Sen (IIT, Kharagpur). The list of selected papers is available at http://www.fsttcs.org.

This will be the 25th edition of the conference. To mark the silver jubilee, the conference has been scheduled to run over four days instead of the usual three, with seven invited talks that will survey advances in different fields over the last 25 years. The invited speakers this year are Manindra Agrawal, Tom Henzinger, Russell Impagliazzo, Raimund Seidel, Natarajan Shankar, Joel Spencer and Igor Walukiewicz. In addition, there will be a special session to commemorate the pioneers who helped establish this conference in its formative years.

In addition to invited talks and contributed papers, the FSTTCS 2005 programme will have two pre-conference workshops during December 12–14.

- *Software verification*, December 12–13, organized by P Madhusudan and Sriram Rajamani.
- *Algorithms for Networks*, December 14,organized by Amit Kumar and Aravind Srinivasan.

Other winter conferences. December is conference season in India. Here is a list of some of the other meetings being held around the country in December.

- 11th Asiacrypt Conference, Chennai, December 4–8, 2005.
  http://www.cs.iitm.ernet.in/~ac05
- 6th Indocrypt Conference, Bangalore, December 10–12, 2005.
  http://www.isical.ac.in/~indocrypt
- 12th HiPC Conference, Goa, December 18–21, 2005.
  http://www.hipc.org
- 2nd IICAI Conference, Pune, December 20–22, 2005.
  http://www.iiconference.org
- 12th COMAD Conference, Hyderabad, December 20–22, 2005.
  http://www.iiit.ac.in/comad2005
- 2nd ICDCIT Conference, Bhubaneswar, December 22–24, 2005.
  http://www.cse.iitk.ac.in/users/pmitra/ICDCIT05
- 7th IWDC Workshop, Kharagpur, December 27–30, 2005.
  http://www.iitkgp.ac.in/iwdc2005

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

News from Ireland

BY

Anthony K. Seda

Department of Mathematics, National University of Ireland
Cork, Ireland
a.seda@ucc.ie

A number of developments have occurred in Ireland recently which have expanded considerably the range of activity in TCS in this country. I want to report on these in this column and also on forthcoming events scheduled to take place in Ireland in the near and relatively near future.

A workshop on Complexity has been organized by the Hamilton Institute, Maynooth for September 16, 2005, and more details can be found at the website http://www.hamilton.ie/complexity/. The coverage of topics will be wide, and ranges from computational complexity in computer science, to the theory of complex dynamical systems. In addition, applications of various different approaches to complexity in biology, physics, engineering, communications systems, computer science, economics, social sciences and mathematics will be discussed. This is the first such meeting in the country and represents a coming together of the complexity theorists in Ireland.

Another new initiative and coming together of individuals also centres on a workshop to be held at the Hamilton Institute, NUI Maynooth, this time on September 27th and 28th, 2005. In this case, the subject is Congestion Control in Networks, and specific topics for discussion will include buffer sizing, AQM and design of congestion avoidance techniques. More details can be found at the website http://www.hamilton.ie/dwmalone/cc_workshop.html/.

As usual, there will be the annual meeting of the Artificial Intelligence Association of Ireland. This year, the 16th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'05) will take place from 7th September to the 9th September, 2005 in Ballycastle, Northern Ireland. Topics to be addressed include

reasoning under uncertainty, image processing, genetic programming, neural networks, and machine learning.

Finally, I want to give advance notice of next year's MFCSIT conference. The Fourth Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT'06) will be collocated with the Fourth International Conference on Information (Info'06). The combined conference will take place in the National University of Ireland, Cork, from the 1st August until the 5th August, 2006. The three previous MFCSIT conferences took place in NUI Cork in 2000, in NUI Galway in 2002, and in TCD Dublin in 2004, see http://www.cs.tcd.ie/MFCSIT2004/. The three previous Information conferences took place in Fukuoka, Japan in 2000, in Beijing in 2002, and in Tokyo in 2004, see http://www.information-iii.org/conf/info2004.html. It is expected that the proceedings of this conference will again appear as a volume in Elsevier's series Electronic Notes in Theoretical Computer Science (ENTCS), see http://math.tulane.edu/~entcs/ and in the Information Journal, see http://www.information-iii.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

0. Waikato University's Computer Science Machine Learning Group has won the 2005 SIGKDD Data Mining and Knowledge Discovery Service Award for their landmark product Weka (Waikato Environment for Knowledge Analysis), `http://www.cs.waikato.ac.nz/~ml/weka/index.html`. The award, the highest service prize in the field of data mining and knowledge discovery, is given to an individual or group who has performed significant service to the data mining and knowledge discovery field, including professional volunteer services in disseminating technical information to the field, education, and research funding. The group was recognised for their development of the freely-available Weka Data Mining Software, and the accompanying book by Ian H. Witten, Eibe Frank *Data Mining: Practical Machine Learning Tools and Techniques* (now in second edition), which has become one of the most popular textbooks for data mining and machine learning. The award was presented at the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining held in Chicago on August 21st.

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

262. M.C. Wilson. Asymptotics for Generalized Riordan Arrays. 04/2005

263. R. Pemantle and M.C. Wilson. Twenty Combinatorial Examples of Asymptotics Derived From Multivariate Generating Functions. 04/2005

264. L. Staiger. Infinite Iterated Function Systems in Cantor Space and the Hausdorff Measure of omega-power Languages. 04/2005

265. M. Stay. Very Simple Chaitin Machines for Concrete AIT. 05/2005

266. R. Eimann, U. Speidel, N. Brownlee and J. Yang. Network Event Detection with T-Entropy. 05/2005

267. M.R. Titchener, U. Speidel and J. Yang. A Comparison of Practical Information Measures. 05/2005

268 D.M. Greenberger and K. Svozil. Quantum Theory Looks at Time Travel. 06/2005

269. K. Svozil. Characterization of Quantum Computable Decision Problems by State Discrimination. 06/2005

270. S. Comoroşan. Computing with Molecules: A New Type of Quantum Molecular Computation. 07/2005

271. M.A. Stay. Truth and Light: Physical Algorithmic Randomness. 08/2005

2 Interview with Academician Solomon Marcus (I)

The biography of Academician Marcus presented at the Université Interdisciplinaire de Paris, http://www.uip.edu/gpss/Solomon_Marcus.html reads: "Dr. Solomon Marcus is a member of the Mathematical Section of the Romanian Academy and Emeritus Professor of the University of Bucharest. His publications have been in the field of mathematical analysis, mathematical and computational linguistics, computer science, poetics, linguistics, semiotics, philosophy and history of science, education, relations between science, humanities, philosophy and religion. He has published 40 books, 400 research papers and several hundreds of papers on various cultural topics of general interest."

Cristian Calude: Professor Marcus, congratulations on your 80th anniversary. This is an wonderful opportunity to talk about your activity. How did you start your career as mathematician?

Solomon Marcus: You said ‘mathematician’. I am very cautious to use this word as far as I am concerned. Norbert Wiener wrote “I am a mathematician", but Paul Halmos, an excellent scholar, who, however, cannot compete with Wiener, used a different title: “I want to be a mathematician".

CC: Then, let us refer to your relation with mathematics.

SM: My first attraction to mathematics appeared during the summer of the year 1944, when I was 19. I read something about non-Euclidean geometry. Its coexistence with the Euclidean one, each of them perfectly consistent, but contradicting the other, impressed me first of all from a logical viewpoint. Then, in October 1944, when I had to make a choice, my option for the study of mathematics was mainly of a negative nature. I did not like subjects like natural sciences or history, requiring a huge memory effort (at least this was their image in the high school textbooks of the time); mathematics seemed to me less demanding in this respect. Visiting the Faculty of Science (Math Section), of the University of Bucharest, I got new arguments for the same choice: titles of courses such as ‘Infinitesimal analysis’, ‘Transfinite cardinals and ordinals’, ‘Probability theory’, although not very clear for me at that time, made me very curious; I had a vague feeling that I will like them.

CC: If these are the only motivations which brought you to mathematics, they don’t seem to be very serious.

SM: You are right. In November 1944 my strategy was to check for one year my interest and capacity to assimilate Mathematics, and then to decide later whether I continue this adventure or I move to a different place. It happened that the first alternative proved to be valid. Moreover, the experience of the first year gave me more substantial reasons to continue my mathematical activity.

As a matter of fact, before paying attention to mathematics, I was attracted, as a teenager, by poetry. I read poetry much beyond what was required in school. I used to transcribe in special notebooks pieces of poetry I liked the most, from Poe and Baudelaire to Rilke and Esenin, but giving priority to Romanian poets such as Eminescu, Arghezi, Blaga and Barbu. I was also attracted by theatre: I was fascinated by Ibsen, Wilde, Shaw and Caragiale. So, you may wonder why I did not choose the Faculty of Letters (Arts). There are two reasons. First of all, as a survivor of the second world war, I got a sense of the gravity of life and of the need to have a sure profession; mathematics, as a basic support for engineering, gave me a feeling of stability, I was convinced that, in the unforeseeable world opened by the second word war, any further evolution will need teachers of mathematics. Second argument: literature seemed to me a topic I could know without university studies, while a similar possibility for mathematics appeared less plausible. So, I

chose mathematics.

CC: Wouldn't have been better to choose engineering, more practical and including mathematics too?

SM: I have never liked technical activities, although later I discovered that the distinction science/engineering is not so sharp as I supposed. It seems to me that it is very risky to choose a profession for which you don't have a strong attraction.

Many of those who made the same choice left mathematics after the first year of studies, either because they moved to engineering or because they were not able to face the rigour of examinations. As I said, it was not my case.

CC: As a student in mathematics, what new arguments did you get for your option?

SM: I enjoyed university mathematics, but my way to approach it was strongly influenced by my (existing) passion for poetry and theatre. Against the common opinion, according to which poetry and mathematics are rather opposite than similar, for me university mathematics was from the very beginning something in a natural, synergetic relation with poetry. With poetry I experienced for the first time the feeling of infinity, of paradox, of frustrated expectation, while the epsilon-delta approach to calculus lead me to similar situations, revealing me the world of processes with infinitely many steps and, as a consequence of this infinity, the results conflicting with our intuitive expectations. When you learn that the square has the same number of points as the cube and that the Riemann alternate series $1 - (1/2) + (1/3) - (1/4) + \ldots$ may have the sum equal to any arbitrarily given real number, as soon as you modify conveniently the order of the terms, then you decide that mathematics is both amazing and funny.

CC: Sometimes you wrote about the role of analogy in mathematics; is it also a common pattern with poetry?

SM: Obviously. The power of analogy in both poetry and mathematics is explained by the crucial role metaphors have in both subjects. All generalisation processes, so important in mathematics, are based on cognitive and creative metaphors. It is not by chance that the International Congress of Mathematicians in 1990 selected "The Mathematical Metaphor" as a topic of an invited address. I also enjoyed projective geometry, with its duality theorems, but I learned only some years later that this chapter of geometry was the mathematical product of some great artists of Renaissance, from Leonardo to Dürer, in connection with the phenomenon called 'perspective'.

CC: Did you intend from the beginning to become a researcher in mathematics?

SM: School mathematics did not give me the impression that something is left to be discovered in this field. On the other hand, I was ignorant about the existence of a profession called 'mathematics researcher', with the same status as engineering, medical doctor, teacher, accountant, etc. In the horizon of the late forties of the past century, my only aim was to become a school teacher and I was happy that I found a possible profession giving me an intellectual satisfaction.

CC: Your Erdös number is one, with a paper dating from 1957. Please tell us about it. Where did you meet Erdös?

SM: I met him for the first time at the Fourth Congress of Romanian Mathematicians, in Bucharest (1956).

CC: Were you already familiar with his mathematical work?

SM: I knew some of his papers very near to my interests at that time, specifically combinatorial set theory, the Borelian hierarchy and measure theory. At that moment I was involved in a problem proposed by Émile Borel in 1946: 'Is there a decomposition of the real line in a finite number > 1 or in a countable infinity of homogeneous sets?'

CC: What does it mean 'homogeneous set'?

SM: According to Borel, a set E in the n-dimensional Euclidean space is said to be homogeneous if, for two arbitrary points X and Y in E, the translation XY transforms E into itself. It was introduced in his paper "Les ensembles homogènes" (C.R. Acad. Sci. Paris, 222, 1946, 617–618).

CC: I guess you told this problem to Erdös ...

SM: One day of the Congress was dedicated to a trip to the Carpathian Romanian mountains. I observed that Erdös did not pay much attention to the beauty of nature. I approached him, being curious to experience a dialogue with this mathematician already known for his excentricities. I introduced myself and his reaction was: Do you have an interesting problem? I told him about my approach to Borel's problem. He showed much interest, we have continued to dialogue about it during the respective trip, then by mail. The result has been a joint paper "Sur la decomposition de l'espace euclidien en ensembles homogènes" (Acta Math. Acad. Sci. Hungaricae 8, 1957, 3/4, 443–452).

CC: So, what is the answer to Borel's problem?

SM: In our paper, we solved Borel's problem completely, but we solved also the corresponding problem in any polydimensional Euclidean space. The answer

we got was negative in the case of a finite > 1 decomposition and positive in the case of a decomposition in *m* homogeneous sets, where *m* is a transfinite cardinal smaller than the continuum.

CC: Why did you publish it in French? Did Erdös know French?

SM: I am afraid that Erdös did not speak French and perhaps he did not understand it. At that time, my English was more broken than it is now, so I wrote it in French. As far as I know, Erdös always left to his partners the care to write the final variant, valid for publication, of his joint papers.

CC: Did you continue to work on similar problems?

SM: In 1960, in a paper with another Hungarian mathematician, A. Csaszar (Colloq. Math.7, 1960, 2) we proved, in connection with a result by J. Mycielski, that there exists no decomposition of the interval (0, 1) in $n > 1$ disjoint sets, pairwise superposable by translation; the same holds true for the interval [0, 1].

CC: Do you know how many researchers have got the Erdös number 2 as a result of having a joint paper with you?

SM: I have joint papers or books with mathematicians, computer scientists, linguists, literary researchers, semioticians, political scientists, chemists and a medical doctor. At a first glance, the only one of them having an Erdös number equal to 2, not only due to his collaboration with me, but also to his collaboration with another author, seems to be the chemist A. T. Balaban, who, besides his four joint papers with me, is also author of some joint papers with Frank Harary, whose Erdös number is surely one. But who knows? There are about two hundred authors having Erdós number equal to one, this makes extremely large the number of authors having Erdös number equal to 2 (among them is Einstein too!) and we are still ignorant whether Gauss has or not a finite Erdös number!

CC: Trying to connect your work in mathematical analysis, set theory and topology, on the one hand, and your work in formal language theory, on the other hand, I observe a key word belonging to their common denominator: *symmetry*. This word appears in the title of many papers you have published from the fifties to the nineties of the past century and they culminate with a paper published in the 2000, in 'Real Analysis Exchange', whose title refers to a synthesis of your whole work referring to symmetry. Please give us some details.

SM: After working in fields that were by excellence of continuous mathematics (mathematical analysis and general topology), I moved to fields predominantly of a discrete nature. The continuous-discrete distinction is not so old in mathematics, it became important only in the second half of the past century, under

the stimulus of the emergence of the information paradigm. Working in discrete fields, I tried to take advantage of my experience in continuous fields. I was also curious to find out to what extent notions and results from the later have their correspondent in the former ones. Discreteness and continuity are different faces of the same coin, so there are aesthetic reasons (symmetry reasons too) to expect a strong analogy between these two branches of science that need each other.

CC: One of your first works concerning symmetry refers to a problem of Hausdorff. What was it?

SM: Hausdorff's problem (Fundamenta Math. 25, 1935, 578) is the following: Let f be a real function of a real variable, symmetrically continuous for any real x (i.e., the difference $f(x+h) - f(x-h)$ tends to zero when h is approaching zero). a) Is it possible for f to have an uncountable set of points of discontinuity? b) Given the real set D of Borelian type F_σ (i.e., a countable union of closed sets), does there always exist a symmetrically continuous function having D as its set of points of discontinuity? We will leave aside here the question a). H. Fried proved in 1937 that for D of second Baire category the answer to b) is negative, so b) was replaced by b'), obtained from b) by replacing 'F_σ' with 'F_σ and of first Baire category'.

CC: This is perhaps the moment when you entered the scene?

SM: Right. It is to b') that we gave a negative answer by proving that the Cantor ternary set cannot play the role of D (Bull. Polish Acad. 4, 1956, 4, 201–205). Further improvements were obtained by other authors, but a characterisation of the set D is not yet available.

CC: But there is an earlier paper you published on symmetry.

SM: It happened in 1955 when I proved that any F_σ real set is the set of points where a convenient real function of a real variable is not symmetrically continuous; but the problem to characterise the set of points where an arbitrary real function of a real variable is not symmetrically continuous is still open. (It is known that the condition to be of F_σ type is both necessary and sufficient for a real set to be the set of points of discontinuity of a real function of a real variable.)

CC: Did you find a correspondent of these results in discrete mathematics, more specifically, in formal language theory?

SM: I did not try it, but notions of continuity of a function don't seem to be compatible with discreteness. The situation changes however when we are dealing with symmetry of sets. In this respect, I will refer to another problem, raised by H. Steinhaus (Fund. Math. 1, 1920, 93): What can be said of a real set A such

that, if x and y are in A, then $(x+y)/2 \notin A$? In this way, we are lead to the notion of antisymmetry of a real set A in a point a: if $x \in A$, then $2a - x \notin A$. This notion can be localised, by requiring the respective condition only for a suitable interval with its center in a. S. Ruziewicz (Fund. Math. 7, 1925, 141) proved that if A is locally antisymmetric in each point of A, then its interior Lebesgue measure is equal to zero. In this respect, we have shown that, generally speaking, local symmetry and local antisymmetry of a real set impose the same measure-theoretic and the same topological (Baire category) restrictions: any locally symmetric real set with void interior is of measure zero as soon as it is Lebesgue measurable; the same is true for any locally antisymmetric real set (its interior is obligatory empty). Any locally symmetric real set with empty interior having the Baire property is of first Baire category; the same is true for any locally antisymmetric real set (its interior is obligatory empty).

CC: Constructive analogues of Lebesgue measure and Baire category have been intensively studied in computability and complexity theory. What about formal language theory, did you find analogies?

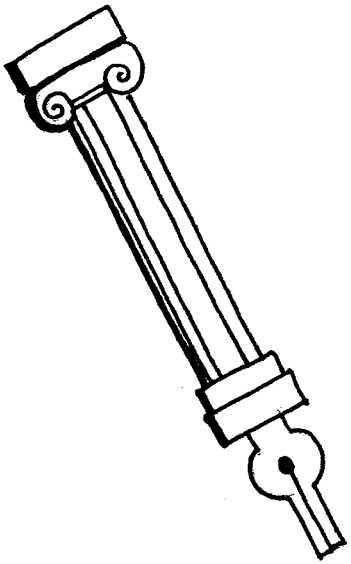
SM: There are, but the analogy with real sets is more subtle, I would say, it is surprising or at least unexpected. The discrete analogue of 'measure zero' and of 'first Baire category' is the property of a language to be regular, while the discrete analogue of measurability and of Baire property is the property to be in Chomsky's hierarchy (or, at least, to be context sensitive).

CC: Please give us some references for your results.

SM: A general account on symmetry of sets and functions was given in my 'Symmetry in the simplest case: the real line' (Computers and Math. Applications 17, 1989, 1/3,103–115), while the discrete analogue was considered in 'Symmetry in languages' (joint work with Gh. Păun, Intern. J. Computer Math. 52, 1994, 1/2, 1–15); 'Symmetry in strings, sequences and languages' (joint work with A. Mateescu, Gh. Păun, and A. Salomaa, Intern. J. Computer Math. 54, 1994, 1/2, 1–13).

CC: Thank you very much for this part of the interview.

The EATCS Columns



The Algorithmics Column

by

Josep Díaz

Department of Languages and Computer Systems
Polytechnical University of Catalunya
c/ Jodi Girona 1-3, 080304 Barcelona, Spain
diaz@lsi.upc.es

The present column deals with the "hot" topic of *exact algorithms*. Fomin, Grandoni and Kratsch give a very beautiful introduction to the existing techniques to design and analyze better exact (exponential) algorithms for solving hard problems. I strongly recommend its reading to anybody working in the field of theoretical computer science.
After four years as editor of the Algorithmics Column, it is time to have an editor with a different perspective. From next issue, Gerhard Woeginger will take over as editor of the Algorithmics Column. The 12 columns that so far have appeared were made possible by the effort of all the authors that have contributed to them. Thanks to all them!

Some new techniques in design and analysis of exact (exponential) algorithms

Fedor V. Fomin* Fabrizio Grandoni† Dieter Kratsch‡

*Department of Informatics, University of Bergen, N-5020 Bergen, Norway, fomin@ii.uib.no. Supported by Norges forskningsråd project 160778/V30. This work was done while the first author was at the Humboldt-Unoversität Berlin, supported by Alexander von Humboldt Foundation.

†Dipartimento di Informatica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy, grandoni@di.uniroma1.it. Supported by Web-Minds project of the Italian Ministry of University and Research, under the FIRB program.

‡LITA, Université de Metz, 57045 Metz Cedex 01, France, kratsch@univ-metz.fr

Abstract

This survey concerns techniques in design and analysis of algorithms that can be used to solve NP hard problems faster than exhaustive search algorithms (but still in exponential time). We discuss several of such techniques: Measure & Conquer, Exponential Lower Bounds, Bounded Treewidth, and Memorization. We also consider some extensions of the mentioned techniques to parameterized algorithms.

1 Introduction

In this survey we use the term *exact algorithms* for algorithms that find exact solutions of NP-hard problem (and thus run in exponential time).

The design of exact algorithms has a long history dating back to Held and Karp's paper [42] on the travelling salesman problem in the early sixties. The last years have seen an emerging interest in constructing exponential time algorithms for combinatorial problems like Coloring [9, 14], Max-Cut [64], 3-SAT [12, 22] (see also the survey of Iwama [43] devoted to exponential algorithms for 3-SAT), Minimum Dominating Set [32], Tree-width [34]. There are two nice surveys of Woeginger [65, 66] describing the main techniques that have been established in the field. We also recommend the paper of Schöning [61] for an introduction to exponential time algorithms.

In this paper we review four techniques for the design and analysis of exact algorithms which were not covered in the mentioned surveys. We also show how some of the techniques can be extended to parameterized algorithms.

The techniques are

- **Measure & Conquer.** For more than 30 years Davis-Putnam-style exponential time search tree algorithms have been the most common tools used for finding exact solutions of NP-hard problems. Despite of that, the way to analyze such recursive algorithms is still far from producing tight worst case running time bounds. The "Measure & Conquer" approach is one of the recent attempts to step beyond such limitations. It is based on the choice of the measure of the subproblems recursively generated by the algorithm considered; this measure is used to lower bound the progress made by the algorithm at each branching step. A good choice of the measure can lead to a significantly better worst case time analysis. We exemplify the approach by showing how to use it for solving the Minimum Dominating Set problem.

- **Lower bounds.** Since it is so difficult to obtain tight worst case running time bounds on exponential time search tree algorithms, the natural ques-

tion, we believe, that should be addressed is to find lower bounds for the worst case running time of such algorithms. [1]

- **Bounded tree-width and dynamic programming.** Dynamic programming is another common tool for exact algorithms. Here we discuss how structural properties of graphs and combinatorial bounds can be used to obtain fast exact algorithms on planar and sparse graphs. We also show how this technique can be used for parameterized algorithms.

- **Memorization.** This technique was introduced by Robson and is used to reduce the running time of exponential-time algorithms at the cost of space. We overview this technique and explain how to use it for parameterized algorithms.

2 Measure & Conquer

In this section we study the analysis of search tree algorithms. Search tree algorithms are also called branch-and-reduce algorithms, splitting algorithms, backtracking algorithms etc. Such an algorithm is recursively applied to a problem instance and uses two types of rules. *Reduction rules* are used to simplify the instance. *Branching rules* are used to solve the problem by recursively calling smaller instances of the problem. An execution of such an algorithm can best be analyzed by a search tree: assign the root node to the input of the problem; recursively assign a child to a node for each smaller instance reached by a branching rule at the instance of the node. Our goal is to analyze the running time of search tree algorithms, i.e. to upper bound the number of nodes of the search tree in the worst case.

In [65] among the major techniques to construct exponential-time algorithms Woeginger lists "pruning the search tree" and describes the classical method to analyze search tree algorithms for the problems INDEPENDENT SET, 3-SAT and BANDWIDTH. The analysis of such recursive algorithms is based on the bounded search tree technique: a measure of the size of an instance of the problem is defined; this measure is used to lower bound the progress made by the algorithm at each branching step. One obtains a linear recurrence or a collection of linear recurrences for each reduction and branching rule. Those linear recurrences can be solved using standard techniques. Finally the worst case is taken over all linear recurrences and a running time of the type $O(\alpha^p)$ is obtained, where p is some (natural) parameter for the size of the problem.

[1] Let us remark that we are interested in exponential lower bounds for a specific class of *algorithms*, so these type of results do not imply that P$\neq$ NP.

For the last 30 years the research on exact algorithms has been mainly focused on the design of more and more sophisticated algorithms. However, measures used in the analysis of search tree algorithms had been usually very simple, e.g. number of vertices for graphs and number of variables for satisfiability problems. Retrospective it is somewhat surprising that almost all analysis of search tree algorithms used standard measures for such a long period. Although a few papers used non-standard measures the general potential of a careful choice of the measure had not been discovered until very recently.

The idea behind Measure & Conquer is to focus on the choice of the (non-standard) measure, instead of creating algorithms with more and more rules. If the measure fulfils the following three conditions then the approach outlined above works.

- The measure of an instance of a subproblem obtained by a reduction or a branching rule must be smaller than the measure of the instance of the original problem.
- The measure of each instance is nonnegative.
- The measure of the input is upper bounded by some function of "natural parameters" of the input.

The last property is needed to retranslate the asymptotic upper bound in terms of the non-standard measure into an upper bound in terms of some natural parameters for the size of the input (such as the number of vertices in a graph or the number of variables in a formula). This way one is able to derive from different (and often complicated) measures, results that are easy to state and compare.

2.1 Eppstein's work

It seems that Eppstein was the first who observed the power of using non-standard measures for analyzing search tree algorithms. He used this type of analysis in several papers, among them [9, 30].

Eppstein's TSP algorithm [30]. There is a well-known dynamic programming $O(2^n \cdot n^2)$[2] algorithm for the TRAVELLING SALESMAN PROBLEM (TSP) by Held and Karp [42] and there is no improvement since 1962. Eppstein studied TSP for graphs of maximum degree three (for which the problem remains NP-hard) and obtained an $O(2^{n/3}n^{O(1)})$ algorithm [30]. More precisely, he studies the TRAVELLING

[2]Whether not specified differently, n and m denote the number of vertices and edges of the input graph, respectively.

SALESMAN PROBLEM WITH FORCED EDGES. The input is a (multi)graph G, a cost function $c : E(G) \to \mathbb{R}$ and a set of forced edges $F \subseteq E(G)$; the output is a minimum cost hamiltonian cycle of G containing all edges of F.

The search tree algorithm is simple. It consists of various reduction rules (step 1 in [30]), a unique branching rule (step 3) and it terminates in a leaf (step 2) if $G - F$ forms a collection of disjoint 4-cycles since in this case a minimum cost solution can be computed in polynomial time. In step 3 an edge xy is chosen and then the algorithm branches in the instances $G, F \cup \{xy\}$ (force xy) and $G - xy, F$ (discard xy). The analysis of the algorithm uses the following interesting non-standard measure: $s(G, F) = |V(G)| - |F| - |C|$, where C is the set of 4-cycles of G that form connected components of $G - F$. Note that despite the negative coefficients in the definition of the measure $0 \leq s(G, F) \leq n$ for all instances G, F. Using this measure the analysis gets fairly easy.

Beigel and Eppstein's 3-coloring algorithm [9]. The $O(2^{0.411n})$ time 3-coloring algorithm presented in [9] is the fastest one known. To a large extent the paper studies special CONSTRAINT SATISFACTION PROBLEMS (CSP). An instance of CSP consists of a collection of n variables, each with a list of possible colors, and a collection of m constraints consisting of a tuple of variables and a color for each variable. A solution assigns a possible color to each variable such that no constraint is satisfied, i.e. not every variable of the constraint is colored in the way specified by the constraint. For an instance of the problem (a, b)-CSP, each variable has at most a possible colors and each constraint involves at most b variables. Note that 3-SAT is equivalent to (2,3)-CSP. Furthermore 3-COLORING, 3-LIST-COLORING and 3-EDGE-COLORING can be translated to (3,2)-CSP.

An $O(2^{0.449n})$ time algorithm for (3,2)-CSP is the fundamental result of [9]. The algorithm also solves (4,2)-CSP and then its running time is $O(2^{0.854n})$. The basic idea is that any (4,2)-CSP instance can be transformed into a (3,2)-CSP instance by expanding each of its four-color variables to two three-color variables. Therefore the natural measure of a (4,2)-CSP instance would be $n = n_3 + 2n_4$, where n_i denotes the number of variables with i possible colors. Crucial for the analysis of the algorithm is the use of the non-standard measure $n = n_3 + (2 - \epsilon)n_4$ where the best choice of ϵ turns out to be $\epsilon \approx 0.095543$.

Eppstein's quasiconvex analysis [31]. Multivariate recurrences frequently arise in the analysis of the worst-case running time of search tree algorithms. Two examples are provided in the paper. One is a subroutine, used in a graph coloring algorithm [29], listing all maximal independent sets of size at most k. In fact when analyzing a search tree algorithm an instance is often characterized by more than one size parameter (variable), and thus it is convenient to establish multivariate recurrences (instead of linear recurrences based on a unique variable) for the reduction and branching rules. Those variables are part of the input or come

up during an execution of the algorithm in a natural way or might be chosen to improve the upper bound of the worst-case running time to be obtained. For example, the linear recurrences in terms of $s(G, F)$ obtained in the analysis of the TSP algorithm in [30] can easily be translated into multivariate recurrences in the variables $|V(G)|$, $|F|$ and $|C|$. Furthermore the linear recurrences in terms of the non-standard measure $n = n_4 + (2 - \epsilon)n_3$ obtained for the reduction and branching rules of the (4,2)-CSP algorithm in [9] can be translated easily into multivariate recurrences in the variables n_3 and n_4.

Given the multivariate recurrences we would like to obtain an upper bound on the running time of the algorithm. Eppstein showed that this multivariate system can be turned into an *equivalent* system of recurrences in a unique variable, where the new variable is a linear combination of the size parameters. It is sufficient to choose the coefficients (weights) which minimize the resulting running time. The optimal weight vector can be computed using quasiconvex programming.

Byskov and Eppstein's maximal bipartite subgraph listing algorithm [16]. The $O(2^{0.826n})$ time algorithm to list all maximal bipartite subgraphs of a graph is the fastest one known. The algorithm can also be found in Byskov's Ph.D. thesis [15] which contains a variety of exponential-time algorithms.

The key operations of the algorithm are: coloring a vertex black (resp. white), remove an edge and remove a vertex. The key idea is that all neighbors of a black (resp. white) vertex can either be white (resp. black) or have to be removed. To indicate this state they will be half-colored: half-white (resp. half-black). Thus an instance of the problem is a half-colored graph $G = (V, F, B, W, E)$ where F is the set of full vertices (i.e. uncolored yet), B is the set of half-black vertices and W is the set of half-white vertices.

The algorithm is based on a lengthy case analysis (p. 35–49 in [15]) generating reduction and branching rules. The analysis of the running time is based on Eppstein's technique: multivariate recurrences and quasiconvex programming. The recurrences depend on two variables: number of full variables and number of half-colored variables. Once provided the long list of two-variable recurrences they will be solved using Eppstein's quasiconvex programming based approach and one obtains the running time $O(2^{0.826n})$ of the algorithm.[3]

2.2 A set cover algorithm

A more careful choice of the measure can lead to a significantly better analysis of the worst case running time of simple search tree algorithms. To illustrate this let us consider the following simple exponential-time search tree algorithm for

[3]However to verify the stated running time without having a special program on hands is non-trivial.

the minimum set cover problem that has been presented in [32] by the authors of this survey. The analysis is based on a sophisticated choice of the measure. This algorithm is used to obtain the fastest known algorithm for the minimum dominating set problem having running time: $O(2^{0.610n})$ using polynomial space and $O(2^{0.598n})$ using exponential space.

In the NP-hard problem Minimum Set Cover (MSC) we are given a universe $\mathcal{U}$ of elements and a collection $\mathcal{S}$ of (non-empty) subsets of $\mathcal{U}$. The aim is to determine the minimum cardinality of a subset $\mathcal{S}' \subseteq \mathcal{S}$ which covers $\mathcal{U}$, that is such that $\cup_{S \in \mathcal{S}'} S = \mathcal{U}$. The *frequency* of $u \in \mathcal{U}$ is the number of subsets $S \in \mathcal{S}$ in which u is contained. For the sake of simplicity, we always assume that $\mathcal{S}$ covers $\mathcal{U}$. With this assumption, an instance of MSC is univocally specified by $\mathcal{S}$.

The NP-hard problem Minimum Dominating Set (MDS) asks to determine the smallest cardinality of a dominating set for the input graph G. Recall that a set $D \subseteq V(G)$ is called a dominating set of the graph G if every vertex of G is either in D, or adjacent to some vertex in D. MDS for an input graph G can be naturally reduced to MSC by imposing $\mathcal{U} = V(G)$ and $\mathcal{S} = \{N[v] | \ v \in V\}$, where $N[v]$ denotes the closed neighborhood of vertex v in G. Thus D is a minimum dominating set of G if and only if $\mathcal{S}' = \{N[v] | \ v \in D\}$ is a minimum set cover of $(\mathcal{U}, \mathcal{S})$. Thus an $O(2^{\alpha(|\mathcal{S}|+|\mathcal{U}|)})$ algorithm for MSC implies an $O(2^{2\alpha n})$ algorithm for MDS.

Consider the following simple recursive search tree algorithm `msc` for solving MSC:

```
1  int msc(S) {
2      if(|S| = 0) return 0;
3      if(∃S, R ∈ S : S ⊆ R) return msc(S\{S});
4      if(∃u ∈ U(S)∃ a unique S ∈ S : u ∈ S)
       return 1+msc(del(S, S));
5      take S ∈ S of maximum cardinality;
6      if(|S| = 2) return poly-msc(S)
7      return min{msc(S\{S}), 1+msc(del(S, S))};
8  }
```

Here $del(S, \mathcal{S}) = \{Z | Z = R \backslash S \neq \emptyset, R \in \mathcal{S}\}$ is the instance of `MSC` which is obtained from $\mathcal{S}$ by removing the elements of S from the subsets in $\mathcal{S}$, and by eventually removing the empty sets obtained. Algorithm `poly-msc` is the polynomial-time minimum set cover algorithm solving MSC for instances where all subsets have cardinality two, which can be reduced to a minimum edge cover problem, based on a well-known reduction to maximum matching.

Essentially algorithm `msc` has two reduction rules (in line 3 and 4) and one branching rule (in line 7). If the maximum cardinality of a subset is at least 3 then

the algorithm chooses a subset S of maximum cardinality and branches into the two subproblems $S_{IN} = del(S, \mathcal{S})$ (the case where S belongs to the minimum set cover) and $S_{OUT} = \mathcal{S} \backslash S$ (corresponding to the case S is not in the minimum set cover). It is easy to see that the simple algorithm is correct.

2.2.1 Analyzing the algorithm `msc`

How should we analyze the running time of `msc`? Classical analysis with the natural measure $s(\mathcal{U}', \mathcal{S}') = |\mathcal{S}'| + |\mathcal{U}'|$ for the size of an instance $(\mathcal{U}', \mathcal{S}')$ of MSC provides an upper bound of $O(2^{0.465(|\mathcal{S}|+|\mathcal{U}|)})$ [40]. (The recurrence corresponding to the unique branching rule is $P(s) \leq P(s-1) + P(s-4)$ where $P(s)$ denotes the number of leaves in the search tree generated by the algorithm to solve a problem of size $s = s(\mathcal{U}, \mathcal{S})$.)

We show how to refine the running time analysis to $O(2^{0.305(|\mathcal{S}|+|\mathcal{U}|)})$ via a more careful choice of the measure of an instance of MSC (without modifying the algorithm!).

Intuition. The choice is based on the following observations showing two "side effects" not taken into account by the above classical analysis: Removing a large set has a different impact on the "progress" of the algorithm than removing a small one. In fact, when we remove a large set, we decrease the frequency of many elements. Decreasing elements frequency pays of on long term, since the elements of frequency one can be filtered out (without branching). A dual argument holds for the elements. Removing an element of high frequency is somehow preferable to removing an element of small frequency. In fact, when we remove an element occurring in many sets, we decrease the cardinality of all such sets by one. This is good on long term, since sets of cardinality one can be filtered out. This suggests the idea to give a different weight to sets of different cardinality and to elements of different frequency in the measure of an instance.

The measure. Let n_i denote the number of subsets $S \in \mathcal{S}$ of cardinality i. Let moreover m_j denote the number of elements $u \in \mathcal{U}$ of frequency j. The measure $s = s(\mathcal{U}, \mathcal{S})$ of the size of an instance of MSC is defined to be: $s(\mathcal{U}, \mathcal{S}) = \sum_{i \geq 1} w_i n_i + \sum_{j \geq 1} v_j m_j$, where the weights $w_i, v_j \in (0, 1]$ will be fixed in the following. Note that $s \leq |\mathcal{S}| + |\mathcal{U}|$. Thus when obtaining a running time $O(2^{\alpha s})$ we may conclude that `msc` has running time $O(2^{\alpha(|\mathcal{S}|+|\mathcal{U}|)})$.

Notation.

$$\Delta w_i = \begin{cases} w_i - w_{i-1} & \text{if } i \geq 3, \\ w_2 & \text{if } i = 2, \end{cases} \quad \text{and} \quad \Delta v_i = \begin{cases} v_i - v_{i-1} & \text{if } i \geq 3, \\ v_2 & \text{if } i = 2. \end{cases}$$

Intuitively, Δw_i (Δv_i) is the reduction of the size of the problem corresponding to the reduction of the cardinality of a set (of the frequency of an element) from i to $i-1$. Let us note that this holds also in the case $i = 2$.

Constraints. In order to simplify the running time analysis, we will add the following constraints:

- $w_1 = v_1 = 1$ and $w_i = v_i = 1$ for $i \geq 6$;
- $0 \leq \Delta w_i \leq \Delta w_{i-1}$ for $i \geq 2$.

Let us observe that this implies that only the weights v_2, v_3, v_4, v_5 and w_2, w_3, w_4, w_5 have still to be fixed. Furthermore for every $i \geq 3$, $w_i \geq w_{i-1}$, and $v_i \geq v_{i-1}$.

Recurrences. Let $P_h(s)$ be the number of subproblems of size h, $0 \leq h \leq s$, solved by `msc` to solve an instance of the MSC of size s. As in a classical analysis for all reduction rules and all branching rules we obtain recurrences. Typically the analysis is more difficult and more tedious than in the case of simple measures because now one branching rule can generate a lot of recurrences.

For the detailed analysis we refer to [32]. We only mention all recurrences corresponding to the unique branching rule (which are practically all important recurrences). Suppose the algorithm has chosen a set S with $|S| \geq 3$ in line 5, thus `msc` branches into two subproblems $\mathcal{S}_{IN} = del(S, \mathcal{S})$ and $\mathcal{S}_{OUT} = \mathcal{S} \backslash S$. Let r_i be the number of elements of S of frequency i. Note that there cannot be elements of frequency 1, and that $\sum_{i \geq 2} r_i = |S|$.

For all the possible values of $|S| \geq 3$ and of the r_i such that $\sum_{i=2}^{6} r_i + r_{\geq 7} = |S|$, we have the following set of recurrences:

$$P_h(s) \leq P_h(s - \Delta s_{OUT}) + P_h(s - \Delta s_{IN}),$$

where

$$\begin{aligned} \Delta s_{OUT} &\triangleq w_{|S|} + \sum_{i=2}^{6} r_i \Delta v_i + r_2 w_2 + \delta(r_2) v_2, \\ \Delta s_{IN} &\triangleq w_{|S|} + \sum_{i=2}^{6} r_i v_i + r_{\geq 7} + \Delta w_{|S|} \left(\sum_{i=2}^{6} (i-1) r_i + 6 \cdot r_{\geq 7} \right), \end{aligned}$$

and $\delta(r_2) = 0$ for $r_2 = 0$, and $\delta(r_2) = 1$ otherwise.

Solving recurrences. Fortunately we can restrict our attention to the case $3 \leq |S| \leq 7$. In fact, since $\Delta w_{|S|} = 0$ for $|S| \geq 7$, each recurrence with $|S| \geq 8$ is "dominated" by some recurrence with $|S| = 7$.

Thus we consider a large but finite number of recurrences. For every fixed 8-tuple $(w_2, w_3, w_4, w_5, v_2, v_3, v_4, v_5)$ the number $P_h(s)$ is within a polynomial factor of α^{s-h}, where α is the largest number from the set of real roots of the set of equations

$$\alpha^s = \alpha^{s - \Delta s_{OUT}} + \alpha^{s - \Delta s_{IN}}$$

corresponding to different combinations of values $|S|$ and r_i. Thus the estimation of $P_h(s)$ boils up to choosing the weights minimizing α.

Choosing weights. This optimization problem is interesting in its own and we refer to Eppstein's work [31] on quasiconvex programming for general treatment of such problems. It turns out that $\alpha = \alpha(v, w)$ is a quasiconvex function of the weights (see [31]). We numerically (using a randomized local search algorithm) obtained the following values of the weights:

$$w_i = \begin{cases} 0.3774 & \text{if } i = 2, \\ 0.7548 & \text{if } i = 3, \\ 0.9095 & \text{if } i = 4, \\ 0.9764 & \text{if } i = 5, \end{cases} \quad \text{and} \quad v_i = \begin{cases} 0.3996 & \text{if } i = 2, \\ 0.7677 & \text{if } i = 3, \\ 0.9300 & \text{if } i = 4, \\ 0.9856 & \text{if } i = 5, \end{cases}$$

which yields $\alpha \leq 1.2352\ldots < 1.2353$.[4]

Running time. Let K denote the set of the possible sizes of the subproblems solved. Note that $|K|$ is polynomially bounded. The total number $P(s)$ of subproblems solved satisfies:

$$P(s) \leq \sum_{h \in K} P_h(s) \leq \sum_{h \in K} \alpha^{s-h} \leq |K|\alpha^s.$$

The cost of solving a problem of size $h \leq s$, excluding the cost of solving the corresponding subproblems (if any), is a polynomial $poly(s)$ of s. Thus the time complexity of the algorithm is

$$O(poly(s)|K|\alpha^s) = O(1.2353^{|\mathcal{U}|+|\mathcal{S}|}) = O(2^{0.305(|\mathcal{U}|+|\mathcal{S}|)}).$$

Theorem 1. *Algorithm* `msc` *solves* MSC *in time* $O(2^{0.305(|\mathcal{U}|+|\mathcal{S}|)})$.

By simply combining the reduction from MDS to MSC with algorithm `msc` one obtains algorithm `mds`.

Corollary 2. *Algorithm* `mds` *solves* MDS *in time* $O(2^{0.305(2n)}) = O(2^{0.610n})$.

Applying the memorization technique described in Section 5 to `mds` the running time can be further reduced to $O(2^{0.598\,n})$.

Though search tree algorithms form a very prominent class of parameterized algorithms, it is yet not fully understood in which way Measure & Conquer can be applied to such algorithms. We leave this as an interesting open problem.

[4] Although computing the weights minimizing α is computationally a non-trivial task, given the weights, checking whether a given α is feasible or not is easy.

3 Exponential lower bounds

Impressive improvements on the upper bound of the worst case running time of a particular exponential-time search tree algorithm can be achieved by a refined analysis and use of a suitable measure as we have seen in the previous section. This suggests the possibility that the time complexity of exponential-time exact algorithms might be largely overestimated. Indeed, most running times of exponential-time search tree algorithms could be too pessimistic and the worst case running time of such an algorithm might be significantly faster. None of the tools and methods for analyzing such algorithms is guaranteed to provide tight upper bounds of the running time.

Consequently, while for most of the known polynomial time algorithms, the known running times seem to be tight, this is most likely not the case for exponential time search tree algorithms. Therefore it is natural to ask for lower bounds of the worst case running time of such algorithms. A lower bound may give an idea how far the running time analysis is from being tight. Furthermore lower bounds might also help to compare exponential-time search tree algorithms.

There are several results on lower bounds for different so-called DPLL algorithms for SAT and k-SAT (see e.g. [5, 54]). However not much more is known on lower bounds for existing exponential-time search tree algorithms for other problems, and in particular for graph problems. One of the reasons to this could be that for most of the graph problems the construction of good lower bounds seems to be a difficult and challenging task even for very simple algorithms.

3.1 A lower bound for algorithm `mds`

The following lower bound for the $O(2^{0.610n})$ polynomial-space algorithm `mds` of the previous section has been provided in [32]. Recall that algorithm `mds` solves the Minimum Dominating Set problem based on a reduction to the Minimum Set Cover problem and uses the algorithm `msc`.

Theorem 3. *The worst case running time of* `mds` *is* $\Omega(2^{0.333n})$.

Proof. Consider the following input graph G_n ($n \geq 1$): the vertex set of G_n is $\{a_i, b_i, c_i : 1 \leq i \leq n\}$. The edge set of G_n consists of two types of edges: for each $i = 1, 2 \ldots, n$, the vertices a_i, b_i and c_i induce a triangle T_i; and for each $i = 1, 2, ..., n-1$: $\{a_i, a_{i+1}\}$, $\{b_i, b_{i+1}\}$ and $\{c_i, c_{i+1}\}$ are edges.

Each node of the search tree corresponds to a subproblem of the minimum set cover problem with input $(\mathcal{U}; \mathcal{S} = \{S_v : v \in V\})$ where $S_v = N[v]$.

We give a selection rule for the choice of the vertices v (respectively sets S_v) to be chosen for the branching. The goal is to choose a selection rule

- which is *compatible* with the algorithm, and
- such that the number of nodes in the search tree obtained by the execution of algorithm `msc` on the instance of MSC generated by the graph G_n is as large as possible.

In each round i, $i \in \{2, 3, \ldots, n-1\}$, we start with a pair $C = \{x_i, y_i\}$ of vertices (belonging to triangle T_i), where $\{x, y\} \subset \{a, b, c\}$. Initially $C = \{a_2, b_2\}$. Our choice makes sure that for each branching vertex x the cardinality of its set S_x is five in the current subproblem $\mathcal{S}$, and that none of the rules of line 2,3 and 4 of the algorithm will ever be applied. Consequently only the branching rule is applied, and by line 7 of `msc` either the set S_v is taken into the set cover ($\mathcal{S} := del(\mathcal{S}, S_v)$), or S_v is removed ($\mathcal{S} := \mathcal{S} \setminus S_v$).

For each pair $C = \{x_i, y_i\}$ of nodes we branch in the following 3 ways
1) take S_{x_i},
2) remove S_{x_i}, and then take S_{y_i},
3) remove S_{x_i}, and then remove S_{y_i}.

The following new pairs of vertices correspond to each of the three branches:
1) $C_1 = \{a_{i+2}, b_{i+2}, c_{i+2}\} \setminus x_{i+2}$,
2) $C_2 = \{a_{i+2}, b_{i+2}, c_{i+2}\} \setminus y_{i+2}$,
3) $C_3 = \{x_{i+1}, y_{i+1}\}$.

On each pair C_j we recursively repeat the process. Thus of the three branches of T_i two are proceeded on T_{i+2} and one is proceeded on T_{i+1}.

To show a lower bound on the worst case running time of algorithm `msc` respectively `mds` on input G_n we analyze the number of leaves of the search tree. Let $P(i)$ be the number of leaves in the search tree when all triangles up to T_i have been used for branching. Thus $P(i) = 2 \cdot P(i-2) + P(i-1)$, and hence $P(i) \geq 2^{i-2}$. Consequently the worst case number of leaves in the search tree of `msc` for a graph on n vertices is at least $2^{n/3-2}$. Thus the worst case running time of `mds` is $\Omega(2^{0.333n})$. □

Notice that there is a large gap between the $O(2^{0.610n})$ upper bound and the $\Omega(2^{0.333n})$ lower bound for the worst case running time of algorithm `mds`. This suggests the possibility that the analysis of algorithm `mds` can be further refined.

4 Tree-width based techniques

The notion of tree-width was introduced by Robertson and Seymour [55]. A *tree decomposition* of a graph G is a pair $(\{X_i : i \in I\},\ T)$, where $\{X_i : i \in I\}$ is a collection of subsets of $V(G)$ and T is a tree such that the following three conditions are satisfied:

1. $\bigcup_{i \in I} X_i = V(G)$.

2. For all $\{v, w\} \in E(G)$, there is an $i \in V(T)$ such that $v, w \in X_i$.

3. For all $i, j, k \in V(T)$, if j is on a path from i to k in T then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition $(\{X_i : i \in V(T)\},\ T)$ is $\max_{i \in V(T)} |X_i| - 1$. The *tree-width* of a graph G, denoted by $\mathbf{tw}(G)$, is the minimum width over all its tree decompositions. A tree decomposition of G of width $\mathbf{tw}(G)$ is called an *optimal* tree decomposition of G.

A tree decomposition $(\{X_i : i \in V(T)\},\ T)$ of G with T being a path is called a *path decomposition* of G. The *path-width* of a graph G, denoted by $\mathbf{pw}(G)$, is the minimum width over all its path decompositions.

A *branch decomposition* of a graph G is a pair (T, μ), where T is a tree with vertices of degree one or three and μ is a bijection from the set of leaves L of T to $E(G)$. Let e be an edge of T. The removal of e results in two subtrees of T, say T_1 and T_2. Let G_i be the graph formed by the edge set $\{\mu(f) : f \in L \cap V(T_i)\}$ for $i \in \{1, 2\}$. The *middle set* $\mathrm{mid}(e)$ of e is the intersection of the vertex sets of G_1 and G_2, i.e., $\mathrm{mid}(e) := V(G_1) \cap V(G_2)$.

The *width* of (T, μ) is the maximum size of the middle sets over all edges of T, and the *branch-width* of G, $\mathbf{bw}(G)$, is the minimum width over all branch decompositions of G. (In case where $|E(G)| \leq 1$, we define the branch-width to be 0; if $|E(G)| = 0$, then G has no branch decomposition; if $|E(G)| = 1$, then G has a branch decomposition consisting of a tree with one vertex—the width of this branch decomposition is considered to be 0).

Tree-width and branch-width are related parameters and can be considered as measures of the "global connectivity" of a graph. The following result is due to Robertson and Seymour [(5.1) in [56]].

Theorem 4 ([56]). *For any connected graph G with $|E(G)| \geq 3$, $\mathbf{bw}(G) \leq \mathbf{tw}(G) + 1 \leq \frac{3}{2}\mathbf{bw}(G)$.*

Tree-width is one of the most basic parameters in graph algorithms. There is a well established theory on the design of polynomial (or even linear) time algorithms for many intractable problems when the input is restricted to graphs of bounded tree-width. See [11] for a comprehensive survey. But what is more important for us, many problems on graphs with n vertices and tree-width (branch-width) at most ℓ can be solved in time $c^{\ell} \cdot n^{O(1)}$, where c is some problem dependent constant.

For example, Alber et al. [1] proved that MDS on graphs of tree-width at most ℓ can be solved in time $O(2^{2\ell} n)$. Fomin and Thilikos showed in [36] that for graphs

G given with a branch-decomposition of width at most ℓ, a minimum dominating set of G can be computed in time $O(3^{\frac{3\ell}{2}}m) = O(2^{5.197n})$. (See also [27] for general discussions on transformations of tree-width based dynamic programming algorithms into algorithms on graphs of bounded branch-width and vice versa.) It can be shown that for graphs of path-width at most ℓ the running time of the algorithm of Alber et al. is $O(3^\ell n)$.

All results mentioned above are based on the following observation.

Observation 5. *Let $\mathcal{P}$ be a problem on graphs and $\mathcal{G}$ be a class of graphs such that*

- *for every graph $G \in \mathcal{G}$ of branch-width at most ℓ, the problem $\mathcal{P}$ can be solved in time $2^{c_{\mathcal{P}}\ell} \cdot n^{O(1)}$, where $c_{\mathcal{P}}$ is a constant, and*
- *for every graph $G \in \mathcal{G}$ a branch decomposition (not necessary optimal) of G of width at most $g(n)$ can be constructed in polynomial time.*

Then for every graph $G \in \mathcal{G}$, the problem $\mathcal{P}$ can be solved in time $2^{c_{\mathcal{P}} \cdot g(n)} \cdot n^{O(1)}$.

Similar observations are valid for tree and path decompositions.

In the following subsections we shall see how Observation 5 combined with good combinatorial upper bounds, provide us with fast algorithms for several interesting graph classes.

4.1 Planar graphs

Using a well-known approach of Lipton and Tarjan [49] based on the celebrated planar separator theorem [48], one can obtain algorithms with time complexity $c^{O(\sqrt{n})}$ for many problems on planar graphs. However, the constants "hidden" in $O(\sqrt{n})$ can be crucial for practical implementations. During the last few years some work has been done to compute and to improve the "hidden" constants [3, 4].

Dynamic programming can be seen as a simpler and, sometimes, faster alternative to the approach of Lipton and Tarjan. To use Observation 5 efficiently, we need to establish upper bounds on the tree-width and branch-width of planar graphs.

Upper bounds. Let α_t and α_b be constants such that for every planar graph $\mathbf{tw}(G) \leq \alpha_t \sqrt{n} + O(1)$ and $\mathbf{bw}(G) \leq \alpha_b \sqrt{n} + O(1)$.

In [6] Alon, Seymour, and Thomas proved that any K_r-minor free graph on n vertices has tree-width at most $r^{1.5}\sqrt{n}$. (Here K_r is complete graph on r vertices.) Since no planar graph contains K_5 as a minor, we have that $\alpha_b(G) \leq \alpha_t(G) \leq 6^{1.5} \leq 14.697$. By using deep results of Robertson, Seymour, and Thomas, one can easily prove much better bounds as follows.

Before we proceed, let us remind the notion of a minor. Given an edge $e = \{x,y\}$ of a graph G, the graph G/e is obtained from G by contracting the edge e; that is, to get G/e we identify the vertices x and y and remove all loops and duplicate edges. A graph H obtained by a sequence of edge-contractions is said to be a *contraction* of G. H is a *minor* of G if H is the subgraph of some contraction of G.

The following is a combination of statements (4.3) in [56] and (6.3) in [58].

Theorem 6 ([58]). *Let $k \geq 1$ be an integer. Every planar graph with no $(k\times k)$-grid as a minor has branch-width at most $4k-3$.*

Since a graph on n vertices does not contain a $((\lceil\sqrt{n}\rceil+1)\times(\lceil\sqrt{n}\rceil+1))$-grid as a minor, we have that $\alpha_b(G) \leq 4$. Fomin and Thilikos [38] obtained the following bounds

Theorem 7 ([38]). $\alpha_b \leq \sqrt{4.5} < 2.1214$ *and* $\alpha_t < 3.1820$.

The proof in [38] makes strong use of deep graph theoretic results from [7] and [57, 62]. In particular, Alon, Seymour and Thomas introduced the concept of "majority" in order to study the existence of small separators in planar graphs. On the other side, the results in [62, 57] are strongly based on the notion of "slope". The main idea of the proof in [38] was to show that slopes can be transformed to majorities.

Now to apply Observation 5, we need to construct a tree or a branch decomposition of small width. It is a long standing open problem whether an optimal tree decomposition of a planar graph can be constructed in polynomial time. For branch decompositions the situation is different. An optimal branch decomposition of a planar graph can be constructed in polynomial time by using the algorithm due to Seymour and Thomas (Sections 7 and 9 in [62]). The algorithm can be implemented such that its running time is $O(n^4)$. Recently, the running time of the algorithm was reduced by Gu and Tamaki to $O(n^3)$ [41].

Putting things together. Thus for planar graphs the function $g(n)$ of Observation 5 can be taken $g(n) = \sqrt{4.5n}$. As we already discussed, for Minimum Dominating Set, $c_{\mathcal{P}} \leq 5.1962$, and we arrive at the fastest known algorithms on planar graphs for MDS with running time

$$O(3^{\frac{3}{2}\sqrt{4.5\cdot n}}n + n^3) = O(2^{5.044\sqrt{n}}).$$

Similar approach yields an algorithm for Maximum Independent Set on planar graphs with running time $O(2^{3.182\sqrt{n}})$.

This machinery not only improves the time bounds but also provides an unified approach for many exponential time algorithms emerging from the planar separator theorem of Lipton and Tarjan [48, 49]. (See [37] for further details.)

Non-local problems. Observation 5 cannot be used to obtain $2^{O(\sqrt{n})}$ time algorithms on planar graphs for "non-local" problems like Hamiltonian Cycle (HC), where we are asked if the input graph has a Hamiltonian cycle, i.e. a (simple) cycle containing all vertices of the graph. The reason is that all known algorithms, solving HC on graphs of branch-width at most ℓ have running time $2^{O(\ell \log \ell)}n^{O(1)}$, thus on planar graphs Observation 5 yields only algorithms with running time $2^{O(\sqrt{n}\log n)}$).

The intuition, why only $2^{O(\ell \log \ell)}n^{O(1)}$ time algorithms for HC on graphs of branch-width at most ℓ are known is the following. While performing dynamic programming, we keep for every edge e of the branch decomposition the set of "patterns" which encode all possible information how possible hamiltonian cycles can hit mid(e). The only known way of doing this is basically to keep as the states of dynamic programming all possible permutations of the set mid(e), which ends up in running time $2^{O(\ell \log \ell)}n^{O(1)}$. This seems to be a natural obstacle and no significantly faster algorithm solving Hamiltonian cycle on graphs of bounded branch-width (or tree-width) is known.

Note that for obtaining $2^{O(\sqrt{n})}$ time algorithms for MDS on planar graphs, planarity comes into play twice: First in the upper bound on the branch-width of a graph and second in the polynomial time algorithm constructing an optimal branch decomposition. It is possible to get rid of the logarithmic factor in the exponent for a number of nonlocal problems as well. The main idea to speed-up algorithms obtained by the branch decomposition approach is to exploit planarity for the third time: use planarity in dynamic programming on graphs of bounded branch-width. To explain how planarity can be used in dynamic programming, we need to go deeper into the properties of planar branch decompositions.

It is more convenient to work with graphs embedded on a sphere instead of a plane. Let Σ be a sphere $(x, y, z\colon x^2 + y^2 + z^2 = 1)$. By a Σ-*plane* graph G we mean a planar graph G with the vertex set $V(G)$ and the edge set $E(G)$ drawn (without crossing) in Σ. An O-arc is a subset of Σ homeomorphic to a circle. An O-arc in Σ is called *noose* of a Σ-plane graph G if it meets G only in vertices. The *length* of a noose O is $|O \cap V(G)|$, the number of vertices it meets. Every noose O bounds two open discs Δ_1, Δ_2 in Σ, i.e. $\Delta_1 \cap \Delta_2 = \emptyset$ and $\Delta_1 \cup \Delta_2 \cup O = \Sigma$.

For a Σ-plane graph G, we define a *sphere cut branch decomposition* $\langle T, \mu\rangle$ as a branch decomposition such that for every edge e of T there exists a noose O_e bounding the two open discs Δ_1 and Δ_2 such that $G_i \subseteq \Delta_i \cup O_e$, $1 \leq i \leq 2$. Thus the length of the noose O_e is $|\,\mathrm{mid}(e)|$.

It follows almost directly from results of Seymour and Thomas [62] that the optimal branch decomposition constructed by their algorithm is in fact a sphere cut branch decomposition (see [26] for details).

Let C be a Hamiltonian cycle and let O_e be a noose of a Σ-plane graph G corresponding to an edge e of a sphere cut branch decomposition. Here is the

moment when planarity is used for the third time. Because the graph is Σ-plane, the number of possible ways Hamiltonian cycles can hit the noose O_e (which is mid(e)) can be bounded by the $|\text{mid}(e)|$-th Catalan number, which yields almost immediately an algorithm of running time $2^{O(\mathbf{bw}(G))}n^{O(1)} = 2^{O(\sqrt{n})}$.

With a more careful work involving tricks on compressing the number of states in dynamic programming, Dorn et al. [26] established a $O(2^{6.903\sqrt{n}})$ time algorithm solving HC on planar graphs. A similar approach can be used to obtain an $O(2^{10.8224\sqrt{n}})$ time algorithm for Planar Graph TSP, where one asks for a shortest tour visiting all vertices of a weighted planar graph. Similarly, Planar Longest Cycle is solvable in time $O(2^{7.214\sqrt{n}})$.

Finally, let us note that the separator based approach can be used to obtain a $2^{O(\sqrt{n})}$ time algorithm for HC on planar graphs [23]. However, it seems that by making use of branch decompositions one can prove significantly better bounds on the worst case running time of algorithms on planar graphs.

4.2 Parameterized algorithms on planar graphs

A similar approach (with some modifications) can be used for the design of parameterized algorithms on planar graphs. The last ten years have seen a rapid development of a new branch of computational complexity: Parameterized Complexity. (See the book of Downey and Fellows [28].) Roughly speaking, a parameterized problem with parameter k is *fixed parameter tractable* if it admits a solving algorithm with running time $f(k)|I|^\beta$. (Here f is a function depending only on k, $|I|$ is the length of the non parameterized part of the input and β is a constant.) In many cases, $f(k) = c^k$ is an exponential function for some constant c. Some attention was paid to the construction of parameterized algorithms with running time of the kind $f(k) = c^{\sqrt{k}}$ for different problems on planar graphs. The first paper on the subject was the paper by Alber et al. [1] describing an algorithm with running time $O(4^{6\sqrt{34k}}n) = O(2^{69.972\sqrt{k}}n)$ for the Minimum Dominating Set problem on planar graphs.

Let $\mathcal{L}$ be a parameterized problem, i.e. $\mathcal{L}$ consists of pairs (I, k) where I is the input and k is the *parameter* of the problem. *Reduction to linear problem kernel* is the replacement of problem inputs (I, k) by a reduced problem with inputs (I', k') (linear kernel) with constants c_1, c_2 such that

$$k' \leq c_1 k,\ |I'| \leq c_2 k' \text{ and } (I, k) \in \mathcal{L} \Leftrightarrow (I', k') \in \mathcal{L}.$$

(We refer to Downey and Fellows [28] for discussions on fixed parameter tractability and the ways of constructing kernels.)

Observation 8. *Let $\mathcal{L}$ be a parameterized problem (G, k), where G is a graph such that*

- *there is a linear problem kernel* (G', k') *computable in time* $T_{kernel}(|V(G)|, k)$ *with constants* c_1, c_2 *such that an optimal branch decomposition of* G' *is computable in time* $T_{bw}(|V(G')|)$,
- *for graphs of branch-width at most* ℓ, *problem* $\mathcal{L}$ *can be solved in time* $O(2^{c_3\ell}n)$, *where* c_3 *is a constant, and*
- $\mathbf{bw}(G') \leq c_4\sqrt{k}$, *where* c_4 *is a constant.*

Then $\mathcal{L}$ *can be solved in time* $O(2^{c_3c_4\sqrt{k}}k + T_{bw}(|V(G')|) + T_{kernel}(|V(G)|, k))$.

Proof. The algorithm works as follows. First it computes a linear kernel in time $T_{kernel}(|V(G)|, k)$. Then it constructs a branch decomposition of the kernel G' in time $T_{bw}(|V(G')|)$. (If there is no such kernel, the problems has no solution.) The size of the kernel is at most $c_1c_2k = O(k)$. The branch-width of the kernel is at most $c_4\sqrt{k}$ and it takes time $O(2^{c_3c_4\sqrt{k}}k + T_{bw}(|V(G')|) + T_{kernel}(|V(G)|, k))$ to solve the problem. □

Let us on exemplify on parameterize Dominating Set problem how Observation 8 can be used.

The k-Dominating Set problem asks to compute, given a graph G and a positive integer k, a dominating set of size k or to report that no such set exists. Alber, Fellows and Niedermeier [2] show that the k-Dominating Set problem on planar graphs admits a linear problem kernel. (The size of the kernel is $335k$. Recently this result was improved to $67k$ by Chen et al. [17].) This reduction can be performed in $O(n^3)$ time. As we already mentioned, the MDS on graphs of branch-width at most ℓ can be solved in time $O(2^{3\log_4 3\cdot\ell}m)$ [36]. Thus $c_3 \leq 3\log_4 3$.

What about the constant c_4 for MDS? It is proved in [36] that for every planar graph G with a dominating set of size k, the branch-width of G is at most $3\sqrt{4.5}\sqrt{k}$, i.e. $c_4 \leq 3\sqrt{4.5}$. Therefore by Observation 8, k- Dominating set can be solved in time $O(2^{9\cdot\log_4 3\cdot\sqrt{4.5}\sqrt{k}}k + n^3 + k^3) = O(2^{15.130\sqrt{k}} + n^3)$ on planar graphs. This is the fastest known algorithm for k-Planar Dominating Set.

By similar arguments, one can show that k-Vertex Cover on planar graphs can be solved in time $O(k^4 + 2^{4.5\sqrt{k}}k + kn)$. (See [37] for details.)

Parameterized versions of non-local problems. For non-local problems Observation 8 cannot be applied directly, however similar arguments are valid. Let us consider the following parameterized version of Hamiltonian Cycle problem: In the k-Cycle problem we are given a graph G and a positive integer k, the task is to find a cycle of length at least k, or to conclude that there is no such a cycle. By adopting the technique from [26], a longest cycle in a planar graph of branch-width at most ℓ can be found in time $O(2^{3.4\ell}\ell\, n)$. If the branch-width of

G is at least $4\sqrt{k+1}-3$ then by Theorem 6, G contains a $(\sqrt{k+1}\times\sqrt{k+1})$-grid as a minor and thus contains a cycle of length at least k. If the branch-width of G is less than $4\sqrt{k+1}-3$ then we can find the longest cycle in G in time $O(2^{3.4\cdot 4\sqrt{k+1}}\sqrt{k}\,n) = O(2^{13.6\sqrt{k}}\sqrt{k}\,n+n^3)$. By standard techniques (see for example [28]) the recognition algorithm for k-Cycle on planar graphs can easily be turned into one constructing a cycle of length at least k, if such a cycle exists.

The described technique can be applied to a large collection of parameterized problems (so-called bidimensional problems) and it can also be extended to more general graph classes. See [24, 25, 36] for further details.

4.3 Sparse graphs

Another class of graphs for which tree-width based techniques can be used to design exact algorithms are graphs of small maximum degree and graphs with small number of edges.

One of the usual approaches to obtain exact algorithms on sparse graphs are search tree algorithms. There are quite many exact algorithms in the literature for different NP hard problems on sparse graphs and in particular on graphs of maximum degree three, see e.g. [8, 20, 35, 39, 47]

The following result is due to Fomin and Høie [33].

Theorem 9 ([33]). *For any $\varepsilon > 0$, there exists an integer n_ε such that for every graph G with maximum degree at most three and $|V(G)| > n_\varepsilon$, $\mathbf{pw}(G) \leq (1/6+\varepsilon)|V(G)|$.*

The proof of Theorem 9 provides an algorithm to construct a path decomposition of width at most $(1/6+\varepsilon)|V(G)|$. Theorem 9 and Observation 5 imply the following

Corollary 10. *For graphs of maximum degree at most three* MDS *is solvable in time $3^{n/6}\cdot n^{O(1)} = O(2^{0.265n})$.*

By similar approach one can also obtain the fastest known so far $2^{n/6}\cdot n^{O(1)} = O(2^{0.167n})$-time algorithms for Maximum Independent Set and Max-Cut on graphs of maximum vertex degree three. [5]

The proof of Theorem 9 is based on a result of Monien and Preis [50] about the bisection width of 3-regular graphs.

Let us also mention an interesting upper bound on the tree-width of graphs in terms of the number of edges obtained by Kneis et al. [44]

[5] Recently, Kojevnikov and Kulikov [46] announced a new search tree algorithm for Maximum Independent Set on graphs of maximum degree three with running time $2^{n/6}\cdot n^{O(1)}$.

Theorem 11 ([44]). *For any graph G on m edges,* $\mathbf{tw}(G) \leq m/5.217$.

This implies, for example, that Max-Cut can be solved in time $O(2^{m/5.217})$.

4.3.1 Lower bounds

The worst case running time of the algorithms described in this subsection depends on combinatorial bounds on path-width of graphs with maximum degree three. Thus it is natural to ask, how small can be the path-width or tree-width of graphs of maximum degree three, or even 3-regular graphs.

Lower bounds on these graph parameters can be obtained by making use of Algebraic Graph Theory. In particular, Bezrukov et al. [10] (by making use of the second smallest eigenvalues of Ramanujan graph's Laplacian) showed that there are 3-regular graphs with the bisection width at least $0.082n$. (See [10] for more details.) It can be easily shown that the result of Bezrukov et al. also yields the lower bound $0.082n$ for path-width of graphs with maximum degree three.

The gap between $0.082n$ and $0.167n$ for the upper bound on the path-width of 3-regular graphs provides some hopes for faster algorithms.

5 Memorization

The time complexity of many exponential time search tree algorithms can be reduced at the cost of an exponential space complexity via the *memorization* technique by Robson [59]. Memorization works as follows: the solutions of all the subproblems solved are stored in an (exponential-size) database. If the same subproblem turns up more than once, the algorithm is not to run a second time, but the already computed result is looked up. The database is implemented in such a way that the *query time* is logarithmic in the number of solutions stored and polynomial in the size of the problem: this way the cost of each look up is polynomial.

In order to illustrate the technique better, we will consider a specific NP-hard problem, the Minimum Vertex Cover problem (MVC), and a specific algorithm to solve it. The techniques described in this section can easily be adapted to many other algorithms and problems. Moreover, for the sake of simplicity, we will analyze the algorithm with the standard measure (using Measure & Conquer, better bounds are achievable).

MVC consists in determining the minimum cardinality of a subset V' of vertices (*vertex cover*) such that every edge is incident to at least one vertex in V'. Let us consider the following simple search tree algorithm to solve MVC: (1) if there is a vertex v of degree zero, remove it; (2) if there is a vertex v of degree one, add w to the vertex cover and remove both v and w (with all the edges incident

to them); (3) select v of maximum degree; (3.a) if $deg(v) = 2$, solve the problem with the trivial polynomial-time algorithm; (3.b) otherwise, branch by either including v or its neighborhood $N(v)$ in the vertex cover, and by removing v or its closed neighborhood $N[v]$, respectively. Solve the two subproblems generated recursively. Observe that each subproblem involves an induced subgraph of the original graph. This property is crucial in order to apply memorization, as it will be clearer soon.[6]

Let $P(n)$ be the number of leaves in the search tree recursively generated by the algorithm to solve the problem on a graph with n vertices. The worst case recurrence, corresponding to the case we branch at a vertex of degree 3, is

$$P(n) \leq P(n-1) + P(n-4),$$

from which we obtain $P(n) < 2^{0.465\,n}$. Since each recursive call takes polynomial time, and the total number of subproblems solved is within a polynomial factor from $P(n)$, the running time of the algorithm (according to the standard analysis) is $O(2^{0.465\,n})$. Let $P_h(n)$, $h \leq n$, be the number of subproblems being graphs with h vertices solved when the algorithm solves MVC on a graph with n vertices. Observe that, by basically the same analysis, $P_h(n) < 2^{0.465(n-h)}$.

5.1 The basic technique

The running time can be reduced, at the cost of an exponential space complexity, in the following way. Whenever we solve a subproblem G', we store the pair $(G', mvc(G'))$ in a database. Before solving any subproblem, we check whether its solution is already available in the database. Observe that, since G has $O(2^n)$ induced subgraphs, the database can be easily implemented such that each query takes polynomial time in n.

There are $\binom{n}{h}$ induced subgraphs of G with h vertices, which implies $P_h(n) \leq \binom{n}{h}$ since no subproblem is solved twice. Moreover the upper bound $P_h(n) \leq 2^{0.465(n-h)}$ still holds. Altogether

$$P_h(n) \leq \min\{2^{0.465(n-h)}, \binom{n}{h}\}.$$

By Stirling's approximation, and balancing the two terms, one obtains that, for each h, $P_h(n) \leq 2^{0.465(n-\alpha n)} < 2^{0.425\,n}$, where $\alpha > 0.0865$ satisfies

$$2^{0.465(1-\alpha)} = \frac{1}{\alpha^{\alpha}(1-\alpha)^{1-\alpha}}.$$

As a consequence, the running time is $O(2^{0.425\,n})$.

[6] Chen, Kanj and Jia [18] erroneously applied memorization to a MVC algorithm which does not satisfy this property; this mistake was later corrected in the journal version of their paper [19].

5.2 A refined approach

If the graph considered is disconnected, one can solve the vertex cover problem corresponding to each connected component separately. More precisely, if $G_1, G_2, \ldots, G_p$ are the connected components of G, then

$$mvc(G) = \sum_{i=1}^{p} mvc(G_i).$$

This, in combination with memorization, can help to further reduce the running time bound, provided that the degree of the graph is bounded by a small constant. In fact, the number of connected induced subgraphs on h vertices of a graph of maximum degree d is much smaller than $\binom{n}{h}$, provided that h is sufficiently small.

Theorem 12 ([59]). *Let $d \geq 3$ be a constant and G a graph of maximum degree d. Let $G(h)$ be the set of all connected induced subgraphs of G on h vertices. Then*

$$|G(h)| = O\left(\left(\frac{(d-1)^{d-1}}{(d-2)^{d-2}}\right)^h n^{O(1)}\right).$$

Proof. The claim is trivially true when $h = n$. So let us assume $h < n$. Consider a graph $G' \in G(h)$. Since G is connected, there must be one edge incident to exactly one vertex of G', say $\{u, r\} \in E(G)$ with $r \in V(G')$ and $u \in V - V(G')$.

Let $T'(r)$ be an arbitrary spanning tree of G' rooted at r (there must be one such tree since G' is connected). Consider an arbitrary ordering of the edges. This numbering allows to univocally associate to $T'(r)$ a $(d-1)$-ary tree T'' (where the position of the children of each vertex is taken into account): the neighbors of each vertex w, excluding the parent vertex (u if $w = r$) are ordered following the ordering on the edges; an edge e which is not in $T'(r)$ gives an empty subtree in T'' in the corresponding position.

Thus, given G and the ordering of the edges, there is a one-to-many mapping between $G(h)$ and the set of triples (v, e, T''), where v is a vertex, e is an edge incident to v, and T'' is a $(d-1)$-ary tree. The claim follows by recalling that the number of $(d-1)$-ary trees is upper bounded by $c(d-1)^{d-1}/(d-2)^{d-2}$, for a small constant c [45]. □

For example, if the maximum degree of a graph is at most 4, one obtains

$$P_h(n) = O(\min\{2^{0.465(n-h)}, (27/4)^h\}),$$

and thus a running time of $O(2^{0.465(1-\alpha)n}) = O(2^{0.398\,n})$, where

$$2^{0.465(1-\alpha)} = (27/4)^\alpha \quad \Leftrightarrow \quad \alpha = \frac{\log(2^{0.465})}{\log(2^{0.465}) + \log(27/4)} > 0.1444.$$

This result can easily be extended to the case of arbitrary graphs, by branching on the vertices of degree 5 or larger in a preliminary phase:

$$P(n) \leq \begin{cases} P(n-1) + P(n-6) \\ 2^{0.398\,n} \end{cases} \leq \max\{2^{0.362\,n},\ 2^{0.398\,n}\}.$$

Observe that vertices of degree smaller than two are removed by reduction rules. Thus, without loss of generality, we can consider in the analysis only the connected induced subgraphs of minimum degree 2: even better upper bounds are available on the number of such graphs.

Theorem 13 ([60]). *Let $d \geq 3$ be a constant and G a graph of maximum degree at most d. Let $G(h, 2)$ be the set of connected induced subgraphs of G with h vertices and minimum degree at least 2. Then $|G(h,2)| = O(c(d)^h\, n^{O(1)})$ where*

$$c(d) = \max_{x \in X} \left\{ 2^{-x_0} \prod_{i=0}^{d-1} \left(\binom{d-1}{i} / x_i \right)^{x_i} \right\},$$

and

$$X = \left\{ x = (x_0, x_1, \ldots, x_{d-1}) \in \mathbb{R}_+^d \ \mid \ \sum_{i=0}^{d-1} x_i = 1 \ \textit{and} \ \sum_{i=0}^{d-1} i\, x_i = 1 \right\}.$$

Proof. Consider an arbitrary $G' \in G(h, 2)$. We consider the same many-to-one mapping from the $(d-1)$-ary trees to the spanning trees of G' as in the proof of Theorem 12, but this time we restrict our attention to the spanning trees with the minimum possible number of leaves ℓ. Note that no two leaves of such spanning trees can be adjacent (otherwise we could create a new spanning tree with one less leaf, which contradicts the minimality assumption). Consider one such tree T' and one of its leaves v. Let $u = u(v)$ be a vertex adjacent to v in G' but not in T', selected arbitrarily. Note that u must exist since the minimum degree is 2, and it must be an internal vertex of T' by the minimality assumption. Let w be the lowest level ancestor of v in T' of degree 3 or larger ($w = r$ is no such vertex exists). We can obtain a different tree T'' with the same number of leaves by adding to T' the edge $e(v) = \{u, v\}$ and by cutting the new cycle introduced at the edge $e'(v) = \{w, w'\}$ right below w in T'. Note that there is a one-to-one mapping between v and both $e(v)$ and $e'(v)$. As a consequence, this replacement of edges can be performed simultaneously on an arbitrary subset of the leaves of the original spanning tree without interference, leading each time to a different spanning tree. This implies that there are at least 2^ℓ distinct spanning trees with ℓ leaves.

Let us give a weight 2^{-h_0} to each spanning tree of G' with $h_0 \geq \ell$ leaves. The weighted sum of such trees is at least one (since there are at least 2^ℓ trees of weight $2^{-\ell}$). As a consequence, the weighted sum of all the spanning trees of the graphs in $G(h,2)$ is an upper bound on $|G(h,2)|$. The number of $(d-1)$-ary trees with h_i vertices of out-degree i, $i \in \{0,1,\ldots,d-1\}$, is upper bounded by

$$\binom{h}{h_0,h_1,\ldots,h_{d-1}}\left(\prod_{i=0}^{d-1}\binom{d-1}{i}^{h_i}\right),$$

where the first factor considers the possible ways to assign out-degrees to vertices, and the second takes into account the positions of the children of each vertex in the tree. Note that the $(h_0,h_1,\ldots,h_{d-1})$ must belong to the following set

$$H = \{(h_0,h_1,\ldots,h_{d-1}) \in \mathbb{N}^d \mid \sum_{i=0}^{d-1} h_i = h \quad \text{and} \quad \sum_{i=0}^{d-1} i\,h_i = h-1\}.$$

With the notation $x_i = h_i/h$ (and letting h tend to infinity),

$$\begin{aligned}\sum_H \left(2^{-h_0}\binom{h}{h_0,h_1,\ldots,h_{d-1}}\prod_{i=0}^{d-1}\binom{d-1}{i}^{h_i}\right) &= O\left(\sum_H\left(2^{-x_0}\prod_{i=0}^{d-1}\left(\binom{d-1}{i}/x_i\right)^{x_i}\right)^h\right)\\ &= O\left(|H|\left(\max_{x\in X}\left\{2^{-x_0}\prod_{i=0}^{d-1}\left(\binom{d-1}{i}/x_i\right)^{x_i}\right\}\right)^h\right).\end{aligned}$$

The claim follows by observing that, for any constant d, $|H|$ is polynomially bounded. □

For example if the maximum degree is 4, one obtains $|G(h,2)| = O(5.5981^h)$, corresponding to the case $(x_0,x_1,x_2,x_3) \simeq (0.2440, 0.5359, 0.1962, 0.0239)$. As a consequence, the running time is $O(2^{0.465(1-\alpha)n}) = O(2^{0.392\,n})$, where

$$\alpha = \frac{\log(2^{0.465})}{\log(2^{0.465}) + \log(5.5981)} > 0.1576.$$

By the same arguments as above, this running time bound extends to graphs of arbitrary degree. Based on this approach, Robson obtained the currently fastest $O(2^{0.250\,n})$ exponential space MVC algorithm [60].

Note that the maximization in Theorem 13 must be performed in a very careful way. In fact, underestimating the value of $c(d)$ would lead to wrong running time bounds. The value of $c(d)$ for some values of d are given in Table 1.

Table 1 Upper bounds on $c(d)$ for $d \in \{3, 4, \ldots, 10\}$.

d	$c(d)$
3	3.4143
4	5.5981
5	7.7654
6	9.9275
7	12.0871
8	14.2455
9	16.4031
10	18.5602

5.3 Memorization in parameterized algorithms

The parameterized k-Vertex Cover problem asks to compute, given a graph G and a positive integer k, a vertex cover of size k or to report that no such set exists.

The algorithm described in the previous subsection can be easily adapted to this task: it is sufficient to update k (besides G) at each recursive call in order to keep track of the number of vertices added to the vertex cover along each search path. Using the same notation as in the previous section, but measuring the progress of the algorithm in terms of k (instead of n), we obtain the following tight recurrence

$$P(k) \leq P(k-1) + P(k-3) < 2^{0.552\,k},$$

which corresponds again to the case in which the algorithm branches at a vertex of degree 3. The corresponding running time is $O(2^{0.552\,k})$.

A linear problem kernel of size $2k$ for the k-Vertex Cover problem (not necessary planar) was obtained by Chen et al. [19]. This result is based on graph-theoretical results of Nemhauser and Trotter [51] and Buss and Goldsmith [13]. The running time of the algorithm constructing such a kernel is $O(kn + k^3)$. Thus $T_{kernel}(|I|, k) = O(kn + k^3)$.

By applying such a kernalization to each subproblem generated, and using the basic memorization technique described in Section 5.1, one obtains

$$P_h(k) \leq \min\{2^{0.552(k-h)}, \binom{2k}{2h}\}.$$

As a consequence, the running time is $O(2^{0.552(1-\alpha)k} + kn) = O(2^{0.528\,k} + kn)$ where $\alpha > 0.044$ satisfies

$$2^{0.552(1-\alpha)} = \left(\frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}}\right)^2.$$

By applying a similar (slightly weaker) approach, Niedermeier and Rossmanith [53] derived a $O(2^{0.360\,k} + kn)$ exponential space vertex cover algorithm from their own $O(2^{0.370\,k} + kn)$ polynomial space algorithm [52].

However, it is not clear a priori how to apply the refined approach of Section 5.2 (based on the number of connected induced subgraphs) to the problem. In fact, consider a vertex cover instance (G, k), where the connected components of G are $G_1, G_2, \ldots, G_p$, with $p \geq 2$. A simple-minded idea is to branch on the subproblems (G_1, k), (G_2, k),..., (G_p, k). Though this approach is correct in principle, it leads to a bad running time bound (since the value of the argument does not decrease in the subproblems).

Chandran and Grandoni [63] described a simple way to circumvent this problem. Suppose the maximum degree is bounded by a constant d. If a connected component contains a small (constant) number of vertices, the corresponding vertex cover problem can be solved in constant time by brute force. Thus, without loss of generality, we can assume that each connected component contains at least $dh + 1$ vertices (and hence at least dh edges), for some constant h to be fixed later. Since each vertex of the vertex cover can cover at most d edges, the size of the minimum vertex cover of each component is at least h. As a consequence, we can branch on the subproblems $(G_i, k - (p-1)h)$ instead of (G_i, k). In fact, if $mvc(G_i) > k - (p-1)h$ for some i, then $mvc(G) > k$. This leads to a new set of recurrences of the kind

$$P(k) \leq \sum_{i=1}^{p} P(k - (p-1)h) \leq 2^{k/h}.$$

By choosing a sufficiently large (but still constant) h, we can ensure that these recurrences are not tight (and thus the worst-case running time is not affected by the branching on the connected components). For example, imposing $h = 3$, one obtains $P(k) < 2^{0.334\,k}$.

By combining this idea with the refined memorization technique described in Section 5.2, we obtain for graphs of degree at most 4 a running time $O(2^{0.552(1-\alpha)k} + kn) = O(2^{0.497\,k} + kn)$ where

$$2^{0.552(1-\alpha)} = 5.5981^{2\alpha} \quad \Leftrightarrow \quad \alpha = \frac{\log(2^{0.552})}{\log(2^{0.552}) + 2\log(5.5981)} > 0.0999.$$

Also in this case the same running time bound extends to graphs of arbitrary degree, provided that vertices of degree 5 or larger are removed in a preliminary phase:

$$P(k) \leq \begin{cases} P(k-1) + P(k-5) \\ 2^{0.552\,k} \end{cases} \leq \max\{2^{0.406\,k}, 2^{0.552\,k}\}.$$

Using this approach, Chandran and Grandoni [63] derived a $O(2^{0.350\,k} + kn)$ exponential space algorithm from the $O(2^{0.370\,k} + kn)$ polynomial space algorithm in [52]. This is the currently fastest algorithm for the parameterized k-Vertex Cover problem.[7]

Acknowledgement. Many thanks to Dimitrios M. Thilikos for his helpful remarks and suggestions.

References

[1] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier, *Fixed parameter algorithms for dominating set and related problems on planar graphs*, Algorithmica, 33 (2002), pp. 461–493.

[2] J. Alber, M. R. Fellows, and R. Niedermeier, *Polynomial-time data reduction for dominating set*, Journal of the ACM, 51 (2004), pp. 363–384.

[3] J. Alber, H. Fernau, and R. Niedermeier, *Graph separators: a parameterized view*, J. Comput. System Sci., 67 (2003), pp. 808–832.

[4] ——, *Parameterized complexity: exponential speed-up for planar graph problems*, J. Algorithms, 52 (2004), pp. 26–56.

[5] M. Alekhnovich, E. Hirsch, and D. Itsykon, *Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas*, in Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2005), vol. 3142 of LNCS, Springer, Berlin, 2004, pp. 84–96.

[6] N. Alon, P. Seymour, and R. Thomas, *A separator theorem for nonplanar graphs*, J. Amer. Math. Soc., 3 (1990), pp. 801–808.

[7] ——, *Planar separators*, SIAM J. Discrete Math., 7 (1994), pp. 184–193.

[8] R. Beigel, *Finding maximum independent sets in sparse and general graphs*, in Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA 1999), ACM and SIAM, 1999, pp. 856–857.

[9] R. Beigel and D. Eppstein, *3-coloring in time $O(1.3289^n)$*, Journal of Algorithms, 54 (2005), pp. 168–204.

[10] S. Bezrukov, R. Elsässer, B. Monien, R. Preis, and J.-P. Tillich, *New spectral lower bounds on the bisection width of graphs*, Theoretical Computer Science, 320 (2004), pp. 155–174.

[11] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoretical Computer Science, 209 (1998), pp. 1–45.

[7] Recently Chen et al. [21] announced $O(1.2740^k + kn) = O(2^{0.350\,k} + kn)$-time polynomial space algorithm.

[12] T. Brueggemann and W. Kern, *An improved deterministic local search algorithm for 3-SAT*, Theoretical Computer Science, 329 (2004), pp. 303–313.

[13] J. F. Buss and J. Goldsmith, *Nondeterminism within* P, SIAM J. Comput., 22 (1993), pp. 560–572.

[14] J. M. Byskov, *Enumerating maximal independent sets with applications to graph colouring*, Operations Research Letters, 32 (2004), pp. 547–556.

[15] J. M. Byskov, *Exact algorithms for graph colouring and exact satisfiability*, PhD thesis, University of Aarhus, Denmark, (August, 2004).

[16] J. M. Byskov, Byskov and D. Eppstein, *An algorithm for enumerating maximal bipartite subgraphs*, manuscript, (2004).

[17] J. Chen, H. Fernau, I. A. Kanj, and G. Xia, *Parametric duality and kernelization: Lower bounds and upper bounds on kernel size*, in Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS 2005), vol. 3403 of LNCS, Springer, Berlin, 2005, pp. 269–280.

[18] J. Chen, I. A. Kanj, and W. Jia, *Vertex cover: further observations and further improvements*, in Proceedings of the 26th Workshop on Graph Theoretic Concepts in Computer Science (WG 1999), vol. 1665 of LNCS, Springer, Berlin, 1999, pp. 313–324.

[19] ——, *Vertex cover: further observations and further improvements*, Journal of Algorithms, 41 (2001), pp. 280–301.

[20] J. Chen, I. A. Kanj, and G. Xia, *Labeled search trees and amortized analysis: improved upper bounds for NP-hard problems*, in Proceedings of the 14th Annual International Symposium on Algorithms and Computation (ISAAC 2003), vol. 2906 of LNCS, Springer, Berlin, 2003, pp. 148–157.

[21] ——, *Simplicity is beaty: Improved upper bounds for vertex cover*, manuscript, 2005.

[22] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning, *A deterministic* $(2 - 2/(k + 1))^n$ *algorithm for k-SAT based on local search*, Theoretical Computer Science, 289 (2002), pp. 69–83.

[23] V. G. Deĭneko, B. Klinz, and G. J. Woeginger, *Exact algorithms for the Hamiltonian cycle problem in planar graphs*, Operations Research Letters, (2005), p. to appear.

[24] E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos, *Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs*, Journal of the ACM, (2004, to appear).

[25] ——, *Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs*, ACM Trans. Algorithms, 1 (2005), pp. 33–47.

[26] F. Dorn, E. Penninkx, H. Bodlaender, and F. V. Fomin, *Efficient exact algorithms on planar graphs: Exploiting sphere cut branch decompositions*, in Proceedings of the 13th Annual European Symposium on Algorithms (ESA 2005), vol. 3669 of LNCS, Springer, Berlin, 2005, pp. 95–106.

[27] F. Dorn and J. A. Telle, *Two birds with one stone: the best of branchwidth and treewidth with one algorithm*, 2005. manuscript, http://www.ii.uib.no/ telle/bib/DT.pdf.

[28] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, New York, 1999.

[29] D. Eppstein, *Small maximal independent sets and faster exact graph coloring*, Journal of Graph Algorithms and Applications, 7 (2003), pp. 131–140.

[30] ——, *The travelling salesman problem for cubic graphs*, in Proceedings of the 8th Workshop on Algorithms and Data Structures (WADS 2003), vol. 2748 of LNCS, Springer, Berlin, 2003, pp. 307–318.

[31] D. Eppstein, *Quasiconvex analysis of backtracking algorithms*, in Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA 2004), ACM and SIAM, 2004, pp. 781–790.

[32] F. V. Fomin, F. Grandoni, and D. Kratsch, *Measure and conquer: Domination – a case study*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), vol. 3580 of LNCS, Springer, Berlin, 2005, pp. 191–203.

[33] F. V. Fomin and K. Høie, *Pathwidth of cubic graphs and exact algorithms*, Technical Report 298, Department of Informatics, University of Bergen, Norway, 2005.

[34] F. V. Fomin, D. Kratsch, and I. Todinca, *Exact algorithms for treewidth and minimum fill-in*, in Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004), vol. 3142 of LNCS, Springer, Berlin, 2004, pp. 568–580.

[35] F. V. Fomin, D. Kratsch, and G. J. Woeginger, *Exact (exponential) algorithms for the dominating set problem*, in Proceedings of the 30th Workshop on Graph Theoretic Concepts in Computer Science (WG 2004), vol. 3353 of LNCS, Springer, Berlin, 2004, pp. 245–256.

[36] F. V. Fomin and D. M. Thilikos, *Dominating sets in planar graphs: Branch-width and exponential speed-up*, in 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003), New York, 2003, ACM and SIAM, pp. 168–177.

[37] ——, *A simple and fast approach for solving problems on planar graphs*, in Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science (STACS 2004), vol. 2996 of LNCS, Springer, Berlin, 2004, pp. 56–67.

[38] ——, *New upper bounds on the decomposability of planar graphs*, Journal of Graph Theory, (2005, to appear).

[39] J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith, *Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT*, Discrete Applied Mathematics, 130 (2003), pp. 139–155.

[40] F. Grandoni, *A note on the complexity of minimum dominating set*, Journal of Discrete Algorithms, (to appear).

[41] Q.-P. Gu and H. Tamaki, *Optimal branch-decomposition of planar graphs in $O(n^3)$ time*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), vol. 3580 of LNCS, Springer, Berlin, 2005, pp. 373–384.

[42] M. Held and R. M. Karp, *A dynamic programming approach to sequencing problems*, Journal of SIAM, 10 (1962), pp. 196–210.

[43] K. Iwama, *Worst-case upper bounds for k-SAT*, Bulletin of the EATCS, 82 (2004), pp. 61–71.

[44] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith, *Algorithms based in treewidth of sparse graphs*, in Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2005), LNCS, Springer, Berlin, 2005, to appear.

[45] D. E. Knuth, *The art of computer programming*, Addison-Wesley, second ed., 1975. Vol. 1: Fundamental algorithms.

[46] A. Kojevnikov and A. S. Kulikov, *A new approach for proving upper bounds for MAX-2-SAT*, 2005. manuscript, http://logic.pdmi.ras.ru/ arist/papers.html.

[47] A. S. Kulikov and S. S. Fedin, *Solution of the maximum cut problem in time $2^{|E|/4}$*, Rossiĭskaya Akademiya Nauk. Sankt-Peterburgskoe Otdelenie. Matematicheskiĭ Institut im. V. A. Steklova. Zapiski Nauchnykh Seminarov (POMI), 293 (2002), pp. 129–138, 183.

[48] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[49] ——, *Applications of a planar separator theorem*, SIAM J. Comput., 9 (1980), pp. 615–627.

[50] B. Monien and R. Preis, *Upper bounds on the bisection width of 3- and 4-regular graphs*, in Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001), vol. 2136 of LNCS, Springer, Berlin, 2001, pp. 524–536.

[51] G. L. Nemhauser and L. E. Trotter, Jr., *Properties of vertex packing and independence system polyhedra*, Math. Programming, 6 (1974), pp. 48–61.

[52] R. Niedermeier and P. Rossmanith, *Upper bounds for vertex cover further improved*, in Proceedings of the 16th International Symposium on Theoretical Aspects of Computer Science (STACS 1999), vol. 1563 of LNCS, Springer, Berlin, 1999, pp. 561–570.

[53] ——, *On efficient fixed-parameter algorithms for weighted vertex cover*, Journal of Algorithms, 47 (2003), pp. 63–77.

[54] P. Pudlak and R. Impaglazzio, *A lower bound for DLL algorithms for k-SAT*, in Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), ACM and SIAM, 2000, pp. 128–136.

[55] N. Robertson and P. D. Seymour, *Graph minors. II. Algorithmic aspects of tree-width*, Journal of Algorithms, 7 (1986), pp. 309–322.

[56] ——, *Graph minors. X. Obstructions to tree-decomposition*, J. Combin. Theory Ser. B, 52 (1991), pp. 153–190.

[57] ——, *Graph minors. XI. Circuits on a surface*, J. Combin. Theory Ser. B, 60 (1994), pp. 72–106.

[58] N. Robertson, P. D. Seymour, and R. Thomas, *Quickly excluding a planar graph*, J. Combin. Theory Ser. B, 62 (1994), pp. 323–348.

[59] J. M. Robson, *Algorithms for maximum independent sets*, Journal of Algorithms, 7 (1986), pp. 425–440.

[60] ——, *Finding a maximum independent set in time $O(2^{n/4})$*, 2001. manuscript, `http://dept-info.labri.fr/ robson/mis/techrep.html`.

[61] U. Schöning, *Algorithmics in exponential time*, in Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS 2005), vol. 3404 of LNCS, Springer, Berlin, 2005, pp. 36–43.

[62] P. D. Seymour and R. Thomas, *Call routing and the ratcatcher*, Combinatorica, 14 (1994), pp. 217–241.

[63] L. Sunil Chandran and F. Grandoni, *Refined memorization for vertex cover*, Information Processing Letters, 93 (2005), pp. 125–131.

[64] R. Williams, *A new algorithm for optimal constraint satisfaction and its implications*, in Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004), vol. 3142 of LNCS, Springer, Berlin, 2004, pp. 1227–1237.

[65] G. Woeginger, *Exact algorithms for NP-hard problems: A survey*, in Combinatorial Optimization - Eureka, you shrink!, vol. 2570 of LNCS, Springer-Verlag, Berlin, 2003, pp. 185–207.

[66] ——, *Space and time complexity of exact algorithms: Some open problems*, in Proceedings of the 1st International Workshop on Parameterized and Exact Computation (IWPEC 2004), vol. 3162 of LNCS, Springer-Verlag, Berlin, 2004, pp. 281–290.

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

Quantum complexity is a young research area of increasing importance. In spite of the scepticism of part of the research community regarding the possibility of constructing quantum machines, there is nowadays at least one session devoted to this topic in every complexity conference. Two experts in the area, Peter Høyer and Robert Špalek write in this column a beautiful survey on quantum query complexity, focusing on the methods for proving lower bounds.

Lower Bounds on Quantum Query Complexity

Peter Høyer* Robert Špalek†

Abstract

Shor's and Grover's famous quantum algorithms for factoring and searching show that quantum computers can solve certain computational

*Department of Computer Science, University of Calgary. Supported by Canada's Natural Sciences and Engineering Research Council (NSERC), the Canadian Institute for Advanced Research (CIAR), and The Mathematics of Information Technology and Complex Systems (MITACS). Email: hoyer@cpsc.ucalgary.ca

†CWI and University of Amsterdam. Supported in part by the EU fifth framework project RESQ, IST-2001-37559. Work conducted in part while visiting the University of Calgary. Email: sr@cwi.nl

problems significantly faster than any classical computer. We discuss here what quantum computers *cannot* do, and specifically how to prove limits on their computational power. We cover the main known techniques for proving lower bounds, and exemplify and compare the methods.

1 Introduction

The very first issue of the Journal of the ACM was published in January 1954. It was the first journal devoted to computer science. For its 50th anniversary volume, published in January 2003, editors-in-chief Joseph Y. Halpern asked winners of the Turing Award and the Nevanlinna Prize to discuss up to three problems that they thought would be major problems for computer science in the next 50 years. Nevanlinna Prize winner Leslie G. Valiant [54] describes three problems, the first of which is on physically realizable models for computation and formalizes the setting by defining: "We therefore call our class PhP, the class of physically constructible polynomial resource computers." He then formulates the problem by: "[t]o phrase a single question, the full characterization of PhP," and argues that "this single question appears at this time to be scientifically the most fundamental in computer science."

On January 26, this year, Nobel Laureate David Gross gave a CERN Colloquium presentation on "The future of physics" [28]. He discusses "25 questions that might guide physics, in the broadest sense, over the next 25 years," and includes as questions 15 and 16 "Complexity" and "Quantum Computing." In July, this year, the Science magazine celebrated its 125th anniversary by "explor[ing] 125 big questions that face scientific enquiry over the next quarter-century" [46]. Among the top 25, is the question of "What are the limits of conventional computing?" Charles Seife writes: "[T]here is a realm beyond the classical computer: the quantum," and he discusses the issue of determining "what quantum-mechanical properties make quantum computers so powerful."

In this issue of the Bulletin of the EATCS, we would like to offer an introduction to the topic of studying limitations on the power of quantum computers. Can quantum computers really be more powerful than traditional computers? What can quantum computers not do? What proof techniques are used for proving bounds on the computational power of quantum computers? It is a highly active area of research and flourishing with profound and beautiful theorems. Though deep, it is fortunately also an accessible area, based on basic principles and simple concepts, and one that does not require specialized prior knowledge. One aim of this paper is to show this by providing a fairly complete introduction to the two most successful methods for proving lower bounds on quantum computations, the adversary method and the polynomial method. Our survey is biased towards the

adversary method since it is likely the least familiar method and it yields very strong lower bounds. This paper is meant to be supplemented by the excellent survey of Buhrman and de Wolf [19] on decision tree complexities, published in 2002 in the journal Theoretical Computer Science.

We demonstrate the methods on a running example, and for this, we use one of the most basic algorithmic questions one may think of: that of searching an ordered set. Can one implement ordered searching significantly faster on a quantum computer than applying a standard $\Theta(\log N)$ binary search algorithm?

The rest of the paper is organized as follows. We motivate and define our models of computation in the next section. We then discuss very basic principles used in proving quantum lower bounds in Section 3 and use them to establish our first lower bound method, the adversary method, in Section 4. We discuss how to apply the method in Section 5, and its limitations in Section 6. We then give an introduction to the second method, the polynomial method, in Section 7. We compare the two methods in Section 8 and give a few final remarks in Section 9.

We have aimed at limiting prior knowledge on quantum computing to a bare minimum. Sentences and paragraphs with kets and bras ($|$this is a ket$\rangle$ and $\langle$this is a bra$|$) can either safely be skipped, or substituted with column-vectors and row-vectors, respectively.

2 Quantum query complexity

Many quantum algorithms are developed for the so-called oracle model in which the input is given as an oracle so that the only knowledge we can gain about the input is in asking queries to the oracle. The input is a finite bitstring $x \in \{0,1\}^N$ of some length N, where $x = x_1 x_2 \dots x_N$. The goal is to compute some function $F : \{0,1\}^N \to \{0,1\}^m$ of the input x. Some of the functions we consider are boolean, some not. We use the shorthand notation $[N] = \{1, 2, \dots, N\}$.

As our measure of complexity, we use the query complexity. The query complexity of an algorithm A computing a function F is the number of queries used by A. The query complexity of F is the minimum query complexity of any algorithm computing F. We are interested in proving lower bounds on the query complexity of specific functions and consider methods for computing such lower bounds.

An alternative measure of complexity would be to use the time complexity which counts the number of basic operations used by an algorithm. The time complexity is always at least as large as the query complexity since each query takes one unit step, and thus a lower bound on the query complexity is also a lower bound on the time complexity. For most existing quantum algorithms, including Grover's algorithm [27], the time complexity is within poly-logarithmic

factors of the query complexity. A notorious exception is the so-called Hidden Subgroup Problem which has polynomial query complexity [23], yet polynomial time algorithms are known only for some instances of the problem.

The oracle model is called decision trees in the classical setting. A classical query consists of an index $i \in [N]$, and the answer of the bit x_i. There is a natural way of modeling a query so that it is reversible. The input is a pair (i, b), where $i \in [N]$ is an index and $b \in \{0, 1\}$ a bit. The output is the pair $(i, b \oplus x_i)$, where the bit b is flipped if $x_i = 1$. There are (at least) two natural ways of generalizing a query to the quantum setting, in which we require all operations to be unitary. The first way is to consider a quantum query as a unitary operator that takes two inputs $|i\rangle|b\rangle$, where $i \in [N]$ and $b \in \{0, 1\}$, and outputs $|i\rangle|b \oplus x_i\rangle$. The oracle is then simply just a linear extension of the reversible query given above. We extend the definition of the oracle so that we can simulate a non-query, and we allow it to take some arbitrary ancilla state $|z\rangle$ with $z \geq 0$ as part of the input and that is acted upon trivially,

$$\mathsf{O}'_x|i, b; z\rangle = \begin{cases} |i, b; z\rangle & \text{if } i = 0 \text{ or } x_i = 0 \\ |i, b \oplus 1; z\rangle & \text{if } i \in [N] \text{ and } x_i = 1. \end{cases} \tag{1}$$

The ancilla $|z\rangle$ contains any additional information currently part of the quantum state that is not involved in the query.

The second way is to consider a quantum query as a unitary operator O_x that takes only the one input $|i\rangle$ and outputs $(-1)^{x_i}|i\rangle$, where $i \in [N]$. We say that the oracle is "computed in the phases" by O_x. Both operators O'_x and O_x square to the identity, i.e., they are their own inverses, and thus unitary. The two operators are equivalent up to a factor of two in that one query to either oracle can be simulated by two queries to the other oracle. Though the first way is possibly the more intuitive, we shall adapt the second way as it is very convenient when proving lower bounds. Again, we extend the definition of the oracle O_x so that it also embodies a non-query, and we allow it to take some arbitrary ancilla state $|z\rangle$ that is not acted upon,

$$\mathsf{O}_x|i; z\rangle = \begin{cases} |i; z\rangle & \text{if } i = 0 \\ (-1)^{x_i}|i; z\rangle & \text{if } 1 \leq i \leq N. \end{cases} \tag{2}$$

We may think of one query as a one-round exchange of information between two parties, the algorithm and the oracle. In the classical setting, the algorithm sends an index $i \in [N]$ to the oracle, and the oracle responds with one bit of information, namely x_i. In the quantum setting, the algorithm sends the $\log_2(N)$ qubits $|i\rangle$ to the oracle O_x, and the oracle responds with $(-1)^{x_i}|i\rangle$. The algorithm and oracle thus exchange a total number of $2\log_2(N)$ qubits, and thus, a quantum

query to O_x can convey up to $2\log_2(N)$ classical bits of information about the oracle by Holevo's theorem [31, 20] and superdense coding [18].

Information theoretically, a function $F : \{0,1\}^N \to \{0,1\}^{\log_2(N)}$ that outputs at most $O(\log_2(N))$ bits, can potentially be solved by a constant number of queries to the oracle. An example of such a problem is the Deutsch-Jozsa problem [22], which is to distinguish balanced boolean functions from constant functions. (A function F is constant if $F(x) = F(y)$ for all inputs x, y, and it is balanced if it is not constant and $|F^{-1}(F(x))| = |F^{-1}(F(y))|$ for all inputs x, y.)

A quantum algorithm in the oracle model starts in a state that is independent of the oracle. For convenience, we choose the state $|0\rangle$ in which all qubits are initialized to 0. It then evolves by applying arbitrary unitary operators U to the system, alternated with queries O_x to the oracle x, followed by a conclusive measurement of the final state, the outcome of which is the result of the computation. In symbols, a quantum algorithm A that uses T queries, computes the final state

$$|\psi_x^T\rangle = \mathsf{U}_T\mathsf{O}_x\mathsf{U}_{T-1}\cdots\mathsf{U}_1\mathsf{O}_x\mathsf{U}_0|0\rangle \tag{3}$$

which is then measured. If the algorithm computes some function $F : \{0,1\}^N \to \{0,1\}^m$, we measure the m leftmost bit of the final state $|\psi_x^T\rangle$, producing some outcome w. The success probability p_x of A on input $x \in \{0,1\}^N$ is the probability that $w = F(x)$. For complete functions $F : \{0,1\}^N \to \{0,1\}^m$, we define the success probability of A as the minimum of p_x over all $x \in \{0,1\}^N$. For partial functions $F : S \to \{0,1\}^m$, where $S \subseteq \{0,1\}^N$, we take the minimum over S only. A quantum algorithm A has error at most ϵ if the success probability of A is at least $1 - \epsilon$. Let $Q_\epsilon(F)$ denote the minimum query complexity of any quantum algorithm that computes F with two-sided error at most ϵ, and as common, let $Q_2(F) = Q_{1/3}(F)$ denote the two-sided bounded error complexity with $\epsilon = 1/3$.

As our running example, we use the well-known ordered searching problem. In the oracle model, the input to ordered searching is an N-bit string $x = (x_1, \ldots, x_N)$. We are promised that $x_i \leq x_{i+1}$ for all $1 \leq i < N$ and that $x_N = 1$, and the goal is to find the leftmost 1, i.e., the index $i \in [N]$ for which $x_i = 1$ and no index $j < i$ exists with $x_j = 1$.

Given: An N-bit string $x = (x_1, x_2, \ldots, x_N)$ given as an oracle.

Promise: $x_i \leq x_{i+1}$ for $1 \leq i < N$ and $x_N = 1$.

Output: Index i such that $x_i = 1$ and either $x_{i-1} = 0$ or $i = 1$.

The classical query complexity of ordered searching is $\lceil \log_2(N) \rceil$ and is achieved by standard binary searching. The quantum query complexity is at most $0.45 \log_2 N$, due to the work of high school student M. B. Jacokes in collaboration with Landahl and Brookes [33] (See also [24, 30]). Using the adversary method, we show that their algorithm is within a factor of about two of being optimal.

3 Distinguishing hard inputs

The first quantum lower bound using adversary arguments was given by Bennett, Bernstein, Brassard, and Vazirani in [8]. They show that any quantum query algorithm can be sensitive to at most quadratically many oracle bits, which implies a lower bound of $\Omega(\sqrt{N})$ for Grover's problem [27] and thus proves that Grover's $O(\sqrt{N})$ algorithm is optimal. Grover's problem is a search problem in which we are given an N-bit string $x \in \{0,1\}^N$ as an oracle, and the goal is to find an index i for which $x_i = 1$, provided one exists. Interestingly, the lower bound of Bennett et al. was proved in 1994, well before Grover defined his search problem. In 2000, Ambainis [3] found an important generalization of the method and coined it "adversary arguments."

A constructive interpretation of basic adversary arguments is in terms of *distinguishability*. We will thus not be concerned about computing the function F, but merely interested in distinguishing oracles. Consider some algorithm A that computes some function F in the oracle model, and consider two inputs $x, y \in \{0,1\}^N$ for which $F(x) \neq F(y)$. Since A computes F, it must in particular be capable of distinguishing between oracle x and oracle y. For a given problem we try to identify *pairs of oracles* that are hard to *distinguish*. If we can identify hard input pairs, we may derive a good lower bound. However, a caveat is that using only the very hardest input pairs does not yield good lower bounds for some problems, and we are thus naturally led to also consider less hard input pairs. A remedy is to use *weights* that capture the hardness of distinguishing each pair of oracles, and to do so, we define a matrix Γ of dimension $2^N \times 2^N$ that takes non-negative real values,

$$\Gamma : \{0,1\}^N \times \{0,1\}^N \to \mathfrak{R}_0^+. \tag{4}$$

We require that Γ is symmetric and that $\Gamma[x,y] = 0$ whenever $F(x) = F(y)$. We say that Γ is a *spectral adversary matrix for* F if it satisfies these two conditions. The symmetry condition on Γ states that we are concerned about distinguishing *between* any two inputs x, y. We are not concerned about distinguishing x *from* y, nor distinguishing y *from* x. We discuss this subtlety further in Section 5 below when considering alternative definitions of weighted adversary arguments. The spectral adversary matrix Γ allows us to capture both total and partial functions, as well as non-boolean functions. Since we are only concerned about distinguishability, once we have specified the entries of Γ, we may safely ignore the underlying function F.

Weighted adversary arguments were first used by Høyer, Neerbek, and Shi in [30] to prove a lower bound of $\Omega(\log N)$ for ordered searching and $\Omega(N \log N)$ for sorting. Barnum and Saks [16] used weighted adversary arguments to prove a lower bound of $\Omega(\sqrt{N})$ for read-once formulae, and introduced the notion Γ

that we adapt here. Barnum, Saks, and Szegedy extended their work in [17] and derived a general lower bound on the query complexity of F in terms of spectral properties of matrix Γ. Their lower bound has a very elegant and short formulation, a basic proof, and captures important properties of adversary methods, and we shall thus adapt much of their terminology.

As discussed above, the key to prove a good lower bound is to pick a good adversary matrix Γ. For our running example of ordered searching, which is a partial non-boolean function, we use the following weights.

Example: Ordered Seaching 1. *The weight on the pair (x, y) is the inverse of the Hamming distance of x and y,*

$$\Gamma^{\text{search}}[x,y] = \begin{cases} \frac{1}{|F(x)-F(y)|} & \text{if } x \text{ and } y \text{ are valid and distinct inputs to } F \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The larger Hamming distance between x and y, the easier it is to distinguish them, and the smaller weight is assigned to the pair.

We have to choose how to measure distinguishability. The possibly simplest measure is to use inner products. Two quantum states are distinguishable with certainty if and only if they are orthogonal, and they can be distinguished with high probability if and only if their inner product has small absolute value.

Fact 1. *Suppose we are given one of two known states $|\Psi_x\rangle, |\Psi_y\rangle$. There exists a measurement that correctly determines which of the two states we are given with error probability at most ϵ if and only if $|\langle\Psi_x|\Psi_y\rangle| \leq \epsilon'$, where $\epsilon' = 2\sqrt{\epsilon(1-\epsilon)}$.*

Since a unitary operator is just a change of basis, it does not change the inner product between any two quantum states, and thus the inner product can only change as a consequence of queries to the oracle.

4 Adversary lower bounds

Adversary lower bounds are information theoretical of nature. A basic idea in adversary lower bounds is to upper bound the amount of information that can be learned in a single query. If little information can be learned in any one query, then many queries are required. We use spectral properties of Γ to put an upper bound on the amount of information the algorithm learns about the oracle.

Let A be some quantum algorithm that computes some function F with bounded two-sided error. For every integer $t \geq 0$ and every oracle x, let

$$|\psi_x^t\rangle = \mathsf{U}_t\mathsf{O}_x \cdots \mathsf{U}_1\mathsf{O}_x\mathsf{U}_0|0\rangle \tag{6}$$

denote the quantum state after t queries to the oracle. To measure the progress of the algorithm, we define similarly to [3, 30, 16, 17] a weight function

$$W^t = \sum_{x,y} \Gamma[x,y]\delta_x\delta_y \cdot \langle\psi_x^t|\psi_y^t\rangle, \tag{7}$$

where δ is a fixed principal eigenvector of Γ, i.e., a normalized eigenvector corresponding to the largest eigenvalue of Γ, and where δ_x denotes the x^{th} entry of δ.

The algorithm starts in a quantum state $|\psi_x^0\rangle = \mathsf{U}_0|0\rangle$ which is independent of the oracle x, and thus the total initial weight is

$$W^0 = \sum_{x,y} \Gamma[x,y]\delta_x\delta_y = \lambda(\Gamma), \tag{8}$$

where $\lambda(\Gamma)$ denotes the spectral norm of Γ. The final state of the algorithm after T queries is $|\psi_x^T\rangle$ if the oracle is x, and it is $|\psi_y^T\rangle$ if the oracle is y. If $F(x) \neq F(y)$, we must have that $|\langle\psi_x^T|\psi_y^T\rangle| \leq \epsilon'$ by Fact 1, and hence $W^T \leq \epsilon' W^0$. If the total weight can decrease by at most Δ by each query, the algorithm requires $\Omega(\frac{W^0}{\Delta})$ queries to the oracle.

Following Barnum, Saks, and Szegedy [17], we upper bound Δ by the largest spectral norm of the matrices Γ_i, defined by

$$\Gamma_i[x,y] = \begin{cases} \Gamma_i[x,y] & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i, \end{cases} \tag{9}$$

for each $1 \leq i \leq n$. The theorem of [17] is here stated (and proved) in a slightly more general form than in [17] so that it also applies on non-boolean functions. Our proof aims at emphasizing distinguishability and differs from the original.

Theorem 2 (Spectral method [17]). *For any adversary matrix Γ for any function $F : \{0,1\}^N \to \{0,1\}^m$,*

$$Q_2(F) = \Omega\Big(\frac{\lambda(\Gamma)}{\max_i \lambda(\Gamma_i)}\Big). \tag{10}$$

Proof. We prove that the drop in total weight $W^t - W^{t+1}$ by the $t+1^{\text{th}}$ query is upper-bounded by the largest eigenvalue of the matrices Γ_i.

For each $0 \leq i \leq N$, let $\mathsf{P}_i = \sum_{z\geq 0} |i;z\rangle\langle i;z|$ denote the projection onto the subspace querying the i^{th} oracle bit. Let $\beta_{x,i} = |\mathsf{P}_i|\psi_x^t\rangle|$ denote the absolute value of the amplitude of querying the i^{th} bit in the $t+1^{\text{th}}$ query, provided the oracle is x. Note that $\sum_{i=0}^N \beta_{x,i}^2 = 1$ for any oracle x, since the algorithm queries one of the N bits $x_1, \ldots, x_N$, or simulates a non-query by querying the oracle with $i = 0$. The $t+1^{\text{th}}$ query changes the inner product by at most the overlap between the

projections of the two states onto the subspace that corresponds to indices i on which x_i and y_i differ,

$$\left|\langle\psi_x^t|\psi_y^t\rangle - \langle\psi_x^{t+1}|\psi_y^{t+1}\rangle\right| = \left|\langle\psi_x^t|(\mathsf{I}-\mathsf{O}_x\mathsf{O}_y)|\psi_y^t\rangle\right| =$$

$$= \left|2\sum_{i:x_i\neq y_i}\langle\psi_x^t|\mathsf{P}_i|\psi_y^t\rangle\right| \leq 2\sum_{i:x_i\neq y_i}\beta_{x,i}\beta_{y,i}. \tag{11}$$

The bigger amplitudes of querying the bits i on which x_i and y_i differ, the larger the drop in the inner product can be.

Define an auxiliary vector $a_i[x] = \delta_x\beta_{x,i}$ and note that

$$\sum_{i=0}^{N} a_i^2 = \sum_{i=0}^{N}\sum_x \delta_x^2\beta_{x,i}^2 = \sum_x \delta_x^2\sum_{i=0}^{N}\beta_{x,i}^2 = \sum_x \delta_x^2 = 1.$$

The drop in the total weight is upper bounded by

$$\begin{aligned}
\left|W^t - W^{t+1}\right| &= \left|\sum_{x,y}\Gamma[x,y]\delta_x\delta_y(\langle\psi_x|\psi_y\rangle - \langle\psi'_x|\psi'_y\rangle)\right| \\
&= \left|2\sum_{x,y}\sum_{i:x_i\neq y_i}\Gamma[x,y]\delta_x\delta_y\langle\psi_x|\mathsf{P}_i|\psi_y\rangle\right| \\
&\leq 2\sum_{x,y}\sum_i \Gamma_i[x,y]\delta_x\delta_y\cdot\beta_{x,i}\beta_{y,i} \\
&= 2\sum_i a_i^*\Gamma_i a_i \\
&\leq 2\sum_i \lambda(\Gamma_i)a_i^2 \\
&\leq 2\max_i\lambda(\Gamma_i)\cdot\sum_i a_i^2 \\
&= 2\max_i\lambda(\Gamma_i).
\end{aligned}$$

Here a_i^* denotes the transpose of a_i. The first inequality bounds the drop in inner product for a specific pair and follows from Equation 11. The second inequality follows from the spectral norm of Γ. The second and third inequalities state that the best possible query distributes the amplitude of the query according to the largest principal eigenvector of the query matrices Γ_i. □

Example: Ordered Seaching 2. *Returning to our example of ordered searching, for $N = 4$, the adversary matrix with respect to the ordered basis* $(0001, 0011, 0111, 1111)$ *is given by*

$$\Gamma^{\text{search}(4)} = \begin{bmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{3} \\ 1 & 0 & 1 & \frac{1}{2} \\ \frac{1}{2} & 1 & 0 & 1 \\ \frac{1}{3} & \frac{1}{2} & 1 & 0 \end{bmatrix}.$$

The spectral norm is easily seen to be lower bounded by the sum of the entries in the first row, $\lambda(\Gamma^{\text{search}(4)}) \geq 1 + \frac{1}{2} + \frac{1}{3}$. *In general,* $\lambda(\Gamma^{\text{search}})$ *is lower bounded by the harmonic number* H_{N-1}, *which is at least* $\ln(N)$. *The spectral norm of the query matrices* $\lambda(\Gamma_i^{\text{search}})$ *is maximized when* $i = \lfloor N/2 \rfloor$, *in which case it is upper bounded by the spectral norm of the infinite Hilbert matrix* $[1/(r+s-1)]_{r,s\geq 1}$, *which is* π. *We thus reprove the lower bound of* $(1-\epsilon')\frac{\ln(N)}{\pi}$ *for ordered searching in given [30].*

5 Applying the spectral method

The spectral method is very appealing in that it has a simple formulation, a basic proof, and gives good lower bounds for many problems. Špalek and Szegedy [51] show that for any problem, the best lower bound achievable by the spectral method is always at least as good as the best lower bound achievable by any of the previously published adversary methods. Their proof is constructive and illuminating: given any lower bound in any of the previously published adversary methods, they construct an adversary matrix Γ and prove it achieves the same lower bound.

The first general quantum lower bound using adversary arguments was introduced by Ambainis in [3]. As shown in [51], it can be derived from the spectral method by applying simple bounds on the spectral norm of Γ and each Γ_i. By definition, the numerator $\lambda(\Gamma)$ is lower-bounded by $\frac{1}{|d|^2} d^* \Gamma d$ for any non-negative vector d, and by Mathias' lemma [39], the denominator $\lambda(\Gamma_i)$ is upper-bounded by the product of a row-norm and a column-norm.

Lemma 3 ([39, 51]). *Let* G *be any non-negative symmetric matrix and* M, N *non-negative matrices such that* $G = M \circ N$ *is the entrywise product of* M *and* N. *Then*

$$\lambda(G) \leq \max_{\substack{x,y \\ G[x,y]>0}} r_x(M)\, c_y(N),$$

where $r_x(M)$ *is the* ℓ_2*-norm of the* x^{th} *row in* M, *and* $c_y(N)$ *is the* ℓ_2*-norm of the* y^{th} *column in* N.

Applying these two bounds, we obtain Ambainis' lower bound in [3]. We refer to the method as an unweighted adversary method since it considers only two types of inputs: easy inputs and hard inputs. We construct a zero-one valued adversary matrix Γ that corresponds to a uniform distribution over the hard input pairs.

Theorem 4 (Unweighted method [3]). *Let* F *be a partial boolean function, and let* $A \subseteq F^{-1}(0)$ *and* $B \subseteq F^{-1}(1)$ *be subsets of (hard) inputs. Let* $R \subseteq A \times B$ *be a relation, and set* $R_i = \{(x,y) \in R : x_i \neq y_i\}$ *for each* $1 \leq i \leq n$. *Let* m, m' *denote the*

minimal number of ones in any row and any column in relation R, respectively, and let ℓ, ℓ' denote the maximal number of ones in any row and any column in any of the relations R_i, respectively. Then $Q_2(f) = \Omega(\sqrt{mm'/\ell\ell'})$.

Proof. Let $S = \{(x,y) : (x,y) \in R \vee (y,x) \in R\}$ be a symmetrized version of R. Define a column vector d from the relation S by setting $d_x = \sqrt{|\{y : (x,y) \in S\}|}$, and an adversary matrix Γ by setting $\Gamma[x,y] = \frac{1}{d_x d_y}$ if and only if $(x,y) \in S$. Then $\lambda(\Gamma) \geq \frac{1}{|d|^2} d^* \Gamma d = 1$. For each of the matrices Γ_i, we apply Lemma 3 with $M[x,y] = N[y,x] = \frac{1}{d_x}$ if and only if $(x,y) \in S$. For every $x \in A$, $r_x(M) \leq \sqrt{\ell/d_x^2} \leq \sqrt{\ell/m}$ and $c_y(N) \leq \sqrt{\ell'/d_y^2} \leq \sqrt{\ell'/m'}$. For every $x \in B$, the inequalities are swapped. By Lemma 3, $\lambda(\Gamma_i) \leq \max_{x,y:\Gamma_i[x,y]>0} r_x(M) c_y(N) \leq \sqrt{\ell\ell'/mm'}$. □

The unweighted adversary method is very simple to apply as it requires only to specify a set R of hard input pairs. It gives tight lower bounds for many computational problems, including inverting a permutation [3], computing any symmetric function and counting [42, 10, 14], constant-level and-or trees [3, 29], and various graph problems [21]. For some computational problems, the hardness does however not necessarily rely only on a few selected hard instances, but rather on more global properties of the inputs. Applying the unweighted method on ordered searching would for instance only yield a lower bound of a constant. In these cases, we may apply the following weighted variant of the method, due to Ambainis [4] and Zhang [57].

Theorem 5 (Weighted method [4, 57]). *Let $F : S \to \{0,1\}^m$ be a partial function. Let w, w' denote a weight scheme as follows:*

- *Every pair $(x,y) \in S^2$ is assigned a non-negative weight $w(x,y) = w(y,x)$ that satisfies $w(x,y) = 0$ whenever $F(x) = F(y)$.*
- *Every triple $(x,y,i) \in S^2 \times [N]$ is assigned a non-negative weight $w'(x,y,i)$ that satisfies $w'(x,y,i) = 0$ whenever $x_i = y_i$ or $F(x) = F(y)$, and $w'(x,y,i)w'(y,x,i) \geq w^2(x,y)$ for all x, y, i such that $x_i \neq y_i$.*

Then

$$Q_2(F) = \Omega\left(\min_{\substack{x,y,i \\ w(x,y)>0 \\ x_i \neq y_i}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,i)}} \right),$$

where $wt(x) = \sum_y w(x,y)$ and $v(x,i) = \sum_y w'(x,y,i)$ for all $x \in S$ and $i \in [N]$.

At first glance, the weighted method may look rather complicated, both in its formulation and use, though it is not. We first assign weights to pairs (x,y) of inputs for which $F(x) \neq F(y)$, as in the spectral method. We require the weights

to be symmetric so that they represent the difficulty in distinguishing *between x and y*.

We then afterwards assign weights $w'(x,y,i)$ that represent the difficulty in distinguishing *x from y by querying index i*. The harder it is to distinguish x from y by index i, compared to distinguishing y from x by index i, the more weight we put on (x,y,i) and the less on (y,x,i), and visa-versa.

To quantify this, define $t(x,y,i) = w'(x,y,i)/w'(y,x,i)$. Then $t(x,y,i)$ represents the relative amount of information we learn about input pairs (x,z) compared to the amount of information we learn about input pairs (u,y), by querying index i. If we, by querying index i, learn little about x compared to y, we let $t(x,y,i)$ be large, and otherwise small. Consider we query an index i for which $x_i \neq y_i$. Then we learn whether the oracle is x or y. However, at the same time, we also learn whether the oracle is x or z for any other pair (x,z) for which $x_i \neq z_i$ and $F(x) \neq F(z)$; and similarly, we learn whether the oracle is u or y for any other pair (u,y) for which $u_i \neq y_i$ and $F(u) \neq F(y)$. The less information querying index i provides about pairs (x,z) compared to pairs (u,y), the larger we choose $t(x,y,i)$. Having thus chosen $t(x,y,i)$, we set $w'(x,y,i) = w(x,y)\sqrt{t(x,y,i)}$ and $w'(y,x,i) = w(x,y)/\sqrt{t(x,y,i)}$.

We show next that the weighted method yields a lower bound of $\Omega(\log N)$ for the ordered searching problem. This proves that the weighted method is strictly stronger than the unweighted method. The weighted method yields strong lower bounds for read-once formula [16] and iterated functions [4]. Aaronson [2], Santha and Szegedy [50], and Zhang [58] use adversary arguments to prove lower bounds for local search, a distributed version of Grover's problem. Špalek and Szegedy prove in [51] that the weighted method is equivalent to the spectral method—any lower bound that can be achieved by one of the two methods can also be shown by the other. Their proof is constructive and gives simple expressions for converting one into the other. The main weights $w(x,y)$ are the coefficients of the weight function W^t for the input pair (x,y), that is, $w(x,y) = \Gamma[x,y]\delta_x\delta_y$, and the secondary weights $w'(x,y,i)$ follow from Mathias' lemma [39] (Lemma 3).

Example: Ordered Seaching 3. *To apply the weighted method on ordered searching, we pick the same weights* $w(x,y) = \Gamma^{\text{search}}[x,y]\,\delta_x\delta_y$ *as in the spectral method as there are no strong reasons for choosing otherwise. Now, consider* $t(x,y,i)$ *with* $F(x) \leq i < F(y)$ *so that* $x_i \neq y_i$. *By querying index i, we also learn to distinguish between x and z for each of the* $F(y) - i$ *inputs z with* $i < F(z) \leq F(y)$, *and we learn to distinguish between u and y for each of the* $i - F(x) + 1$ *inputs u with* $F(x) \leq F(u) \leq i$. *We thus choose to set*

$$t(x,y,i) = \frac{|F(y) - i| + 1}{|F(x) - i| + 1}.$$

Plugging these values into the weighted method yields a lower bound of $\Omega(\log N)$ *for ordered searching.*

6 Limitations of the spectral method

The spectral method and the weighted adversary method bound the amount of information that can be learned in any one query. They do not take into account that the amount of information that can be learned in the j^{th} query might differ from the amount of information that can be learned in the k^{th} query.

In 1999, Zalka [56] successfully managed to capture the amount of information that can be learned in each individual query for a restricted version of Grover's problem [27]. In this restricted version, we are promised that the input oracle x is either the zero-string (so $|x| = 0$) or exactly one entry in x is one (so $|x| = 1$), and the goal is to determine which is the case. By symmetry considerations, Zalka demonstrates that Grover's algorithm saturates some improved inequalities (which are similar to Eq. 11) and hence is optimal, even to within an additive constant.

Since current adversary methods do not capture the amount of information the algorithm currently knows, we may simply assume that the algorithm already knows every bit of the oracle and that it tries to prove so. This motivates a study of the relationship between the best bound achievable by the spectral method and the certificate complexity. A *certificate* for an input $x \in \{0,1\}^N$, is a subset $C \subseteq [N]$ of input bits such that for any other input y in the domain of F that may be obtained from x by flipping some of the indices not in C, we have that $F(x) = F(y)$. The certificate complexity $\mathrm{C}_x(F)$ of input x is the size of a smallest certificate for x. The *certificate complexity* $\mathrm{C}(F)$ of a function F is the maximum certificate complexity of any of its inputs. We also define the z-certificate complexity $\mathrm{C}_z(F)$ when taking the maximum only over inputs that map to z. The spectral theorem can then never yield a lower bound better than a quantity that can be expressed in terms of certificate complexity.

Lemma 6 ([38, 57, 51]). *Let* $F : S \to \{0,1\}$ *be any partial boolean function. The spectral adversary lower bound* $\mathrm{Adv}(F)$ *is at most* $\min\{\sqrt{\mathrm{C}_0(F)N},\ \sqrt{\mathrm{C}_1(F)N}\}$. *If* F *is total, the method is limited by* $\sqrt{\mathrm{C}_0(F)\mathrm{C}_1(F)}$.

The certificate complexity of a function $F : \{0,1\}^N \to \{0,1\}^m$ is itself polynomially related to the block sensitivity of the function. An input $x \in \{0,1\}^N$ is *sensitive* to a block $B \subseteq [N]$ if $F(x) \neq F(x^B)$, where x^B denotes the input obtained by flipping the bits in x with indices from B. The block sensitivity $\mathrm{bs}_x(F)$ of input x is the maximum number of disjoint blocks $B_1, B_2, \ldots, B_k \subseteq [N]$ on which x is sensitive. The *block sensitivity* $\mathrm{bs}(F)$ of F is the maximum block sensitivity of

any of its inputs. We also define the z-block sensitivity $\mathrm{bs}_z(F)$ when taking the maximum only over inputs that map to z.

For any boolean function $F : \{0,1\}^N \to \{0,1\}$, the certificate complexity is upper bounded by $\mathrm{C}(F) \leq \mathrm{bs}_0(F)\mathrm{bs}_1(F)$, and thus so is the spectral adversary method. Conversely, $\mathrm{Adv}(F) \geq \sqrt{\mathrm{bs}(F)}$ by a zero-one valued adversary matrix Γ: Let $x' \in \{0,1\}^N$ be an input that achieves the block sensitivity of F, and let $B_1, B_2, \ldots, B_k \subseteq [N]$ be disjoint blocks on which x' is sensitive, where $k = \mathrm{bs}(F)$. Set $\Gamma(F)[x, x^B] = 1$ if and only if $x = x'$ and B is one of the k blocks B_i and close Γ under transposition. Then $\lambda(\Gamma) = \sqrt{k}$ and $\max_i \lambda(\Gamma_i) = 1$, and thus

$$\sqrt{\mathrm{bs}(F)} \leq \mathrm{Adv}(F) \leq \mathrm{bs}_0(F)\mathrm{bs}_1(F). \tag{12}$$

The spectral adversary method is not suitable for proving lower bounds for problems related to property testing. If function $F : S \to \{0,1\}$ is a partial function with $S \subseteq \{0,1\}^N$ such that every zero-input is of Hamming distance at least εn from every one-input, then the spectral theorem does not yield a lower bound better than $1/\varepsilon$.

Laplante and Magniez introduce in [38] a lower-bound method based on Kolmogorov complexity. They show by direct constructions that their method is at least as strong as each of the two methods, the spectral and weighted adversary method. Špalek and Szegedy then show in [51] that the spectral method is at least as strong as the Kolmogorov complexity method, allowing us to conclude that the three methods are equivalent. Having such a variety of representations of the same method shows that the adversary method is very versatile and captures fundamental properties of functions. Indeed, Laplante, Lee, and Szegedy [37] show that the square of the adversary bound is a lower bound on the formula size. The following lower-bound method is a combinatorial version of the Kolmogorov complexity method.

Theorem 7 (Minimax method [38, 51]). *Let $F : S \to \{0,1\}^m$ be a partial function and A a bounded-error quantum algorithm for F. Let $p : S \times [N] \to \mathfrak{R}_0^+$ be a set of $|S|$ probability distributions such that $p_x(i)$ denotes the average probability of querying the i^{th} input bit on input x, where the average is taken over the whole computation of A. Then the query complexity Q_{A} of algorithm A satisfies*

$$Q_{\mathsf{A}} \geq M_p = \max_{x,y:F(x)\neq F(y)} \frac{1}{\sum_{i:x_i\neq y_i} \sqrt{p_x(i)\, p_y(i)}}.$$

The previous methods satisfy the property that if we plug in some matrix or relation, we get a valid lower bound. The minimax method is principally different. A lower bound computed by the minimax theorem holds for one particular algorithm A, and it may not hold for some other and better algorithm. However,

we may obtain a universal lower bound that holds for *every* bounded error algorithm by simply taking the minimum of the bound M_p over all possible sets of probability distributions p. The spectral bound and the minimax bound are in a primal-dual relation: the best lower bound that can be obtained by any adversary matrix Γ equals the smallest bound that can be obtained by a set of probability distributions p [51]. Primal methods are used for obtaining concrete lower bounds and dual methods are used for proving limitations of the method, as in Lemma 6.

A useful property of the adversary method is that it composes. Consider a function of the form $H = F \circ (G_1, \ldots, G_k)$, where $F : \{0,1\}^k \to \{0,1\}$ and $G_i : \{0,1\}^{N_i} \to \{0,1\}$ for $i = 1, \ldots, k$ are partial boolean functions. A composition theorem states the complexity of function H in terms of the complexities of F and $G_1, \ldots, G_k$. Barnum and Saks [16] use composition properties to prove a query lower bound of $\Omega(\sqrt{N})$ for any read-once formula, Ambainis [4] proves a composition lower bound for iterated boolean functions, and Laplante, Lee, and Szegedy [37] prove a limitation on composition lower bounds for functions G_i for which the adversary bound is upper bounded by a common bound b. To formulate a composition theorem for arbitrary cases when the functions G_i may have different adversary bounds, we require a weighted version of the spectral method.

Let $F : \{0,1\}^N \to \{0,1\}$ be a partial boolean function and $\alpha = (\alpha_1, \ldots, \alpha_N)$ a string of positive reals. Let

$$\mathrm{Adv}_\alpha(F) = \max_\Gamma \min_i \left\{ \alpha_i \frac{\lambda(\Gamma)}{\lambda(\Gamma_i)} \right\},$$

where Γ ranges over all adversary matrices for F. If the weights are all 1, then our new quantity $\mathrm{Adv}_\alpha(F)$ coincides with the spectral adversary bound and is thus a lower bound on the quantum query complexity of F. If the weights α are non-uniform, then $\mathrm{Adv}_\alpha(F)$ is a new abstract complexity measure that assigns cost α_i to querying the i^{th} input bit. We can then prove [32] that the quantity Adv_α composes in the following sense.

Theorem 8 (Composition Theorem [16, 4, 37, 32]). *For any composite function $H = F \circ (G_1, \ldots, G_k)$, where $F : \{0,1\}^k \to \{0,1\}$ and $G_i : \{0,1\}^{N_i} \to \{0,1\}$ are partial boolean functions,*

$$\mathrm{Adv}_\alpha(H) = \mathrm{Adv}_\beta(F),$$

where $\beta_i = \mathrm{Adv}_{\alpha^i}(G_i)$, and $\alpha = (\alpha^1, \ldots, \alpha^k)$ is a k-tuple of strings $\alpha^i \in \Re^{+N_i}$.

A natural generalization of Grover's problem is the so-called k-fold search problem in which we are promised that exactly k entries of the input oracle x are one (so $|x| = k$), and the goal is to find all of these k indices. We say an algorithm A succeeds if it outputs a subset $S \subseteq [N]$ of size k and S contains all indices

$i \in [N]$ for which $x_i = 1$. Thus, by definition, it fails even if it outputs all but one of the k indices. The k-fold search problem can be solved in $O(\sqrt{kn})$ queries, essentially by sequentially running Grover's search algorithm k times. Klauck, Špalek, and de Wolf [35] show that if the number of queries is less than $\epsilon\sqrt{kn}$ for some constant ϵ, then the success probability of A is exponentially small in k. They thus prove a strong direct product theorem for the k-fold search problem. One of the main elements of the proof is the polynomial method which we discuss in the next section.

In very recent work, Ambainis [5] proposes an extension of the adversary method and uses it to reprove the strong direct product theorem of [35]. Though the following very brief description of the proof does not give full justice to the method, we hope it conveys some of the intuition on which [5] is based. The algorithm runs on a uniform superposition of all inputs. During the computation, the input register gets entangled with the workspace of the algorithm due to the queries to the oracle. We trace out the workspace and examine the eigenspaces of the density matrix of the input register. Due to symmetries, there are exactly $k+1$ eigenspaces, indexed by the number of ones the algorithm "knows" at that stage of the algorithm. In the beginning, all amplitude is in the 0^{th} eigenspace. One query can only move little amplitude from the i^{th} eigenspace to the $i+1^{\text{th}}$ eigenspace. If the algorithm has a good success probability, the quantum amplitude of high eigenspaces must be significant, since the algorithm must "know" most of the k indices, which implies a lower bound on the query complexity.

7 Polynomial lower bounds

There are essentially two different methods known for proving lower bounds on quantum computations. The historically first method is the adversary method we discuss above. It was introduced in 1994 by Bennett, Bernstein, Brassard, and Vazirani, and published in 1997 in the SIAM Journal on Computing, in a special section that contains some of the most outstanding papers on quantum computing. The second method was introduced shortly after, in 1998, by Beals, Buhrman, Cleve, Mosca, and de Wolf [9], and implicitly used by Fortnow and Rogers in [25]. Their approach is algebraic and follows earlier very successful work on classical lower bounds via polynomials (see for instance Beigel's 1993 survey [11] and Regan's 1997 survey [44]). We first establish that any partial boolean function $F : S \to \{0,1\}$, where $S \subseteq \{0,1\}^N$, can be represented by a real-valued polynomial $p : \Re^N \to \Re$.

Definition 9. *Let $F : S \to \{0,1\}$ be a partial boolean function, where $S \subseteq \{0,1\}^N$. An N-variable polynomial p* represents *F if $p(x) = F(x)$ for all $x \in S$, and it*

approximates F *if* $|p(x)-F(x)| \leq \frac{1}{3}$ *for all* $x \in S$. *The* degree *of* F, *denoted* $\deg(F)$, *is the minimal degree of a polynomial representing* F. *The* approximate degree *of* F, *denoted* $\widetilde{\deg}(F)$, *is the minimal degree of a polynomial approximating* F.

The crux in [9] is in showing that any quantum algorithm A computing some function F gives rise to some polynomial p_{A} that represents or approximates F.

Theorem 10 ([9]). *Let* A *be a quantum algorithm that computes a partial boolean function* $F : S \to \{0,1\}$, *where* $S \subseteq \{0,1\}^N$, *using at most* T *queries to the oracle* O'_x. *Then there exists an* N*-variate real-valued multilinear polynomial* $p_{\mathsf{A}} : \Re^N \to \Re$ *of degree at most* $2T$, *which equals the acceptance probability of* A.

Proof. In this theorem, we use the oracle O'_x which is equivalent to the oracle O_x, since it allows for simple formulations. We first rewrite the action of O'_x as

$$\mathsf{O}'_x|i,b;z\rangle = (1-x_i)|i,b;z\rangle + x_i|i,b\oplus 1;z\rangle \tag{13}$$

where we define $x_i = 0$ for $i = 0$ so that we can simulate a non-query by querying x_i with $i = 0$. Suppose we apply O'_x on some superposition $\sum_{i,b,z} \alpha_{i,b,z}|i,b;z\rangle$ where each amplitude $\alpha_{i,b,z}$ is an N-variate complex-valued polynomial in x of degree at most j. Then, by Eq. 13, the resulting state $\sum_{i,b,z} \beta_{i,b,z}|i,b;z\rangle$ is a superposition where each amplitude $\beta_{i,b,z}$ is an N-variate complex-valued polynomial in x of degree at most $j+1$. By proof by induction, after T queries, each amplitude can be expressed as a complex-valued polynomial in x of degree at most T. The probability that the final measurement yields the outcome 1, corresponding to accepting the input, is obtained by summing some of the absolute values of the amplitudes squared. The square of any of the absolute amplitudes can be expressed as a real-valued polynomial p_{A} in x of degree at most $2T$. Theorem 10 follows. □

The above theorem states that to any quantum algorithm A computing a boolean function $F : S \to \{0,1\}$, where $S \subseteq \{0,1\}^N$, we can associate an N-variate polynomial $p_{\mathsf{A}} : \Re^N \to \Re$ that expresses the acceptance probability of the algorithm on any given input. If algorithm A is exact, i.e., if A always stops and outputs the correct answer, then $p_{\mathsf{A}}(x) = F(x)$ for all $x \in S$, and thus p_{A} represents F. If A has bounded error, then $0 \leq p_{\mathsf{A}}(x) \leq 1/3$ if $F(x) = 0$ and $2/3 \leq p_{\mathsf{A}}(x) \leq 1$ if $F(x) = 1$, and thus p_{A} approximates F. The degree of p_{A} is at most twice the number of queries used by algorithm A. Consequently, the degree of a function is a lower bound on the quantum query complexity, up to a factor of two.

Corollary 11 (Polynomial method [9]). *For any partial boolean function* $F : S \to \{0,1\}$, *where* $S \subseteq \{0,1\}^N$, *we have* $Q_E(F) \geq \deg(F)/2$ *and* $Q_2(F) \geq \widetilde{\deg}(F)/2$.

8 Applying the polynomial method

The challenge in applying the polynomial method lies in the dimensionality of the input. Typically, the method is applied by first identifying a univariate or bivariate polynomial that captures essential properties of the problem, and then proving a lower bound on the degree of that polynomial. The second part is typically reasonably straightforward since polynomials have been studied for centuries and much is known about their degrees. The possibly simplest nontrivial example is when F is the threshold function Thr_t defined by $\mathsf{Thr}_t(x) = 1$ if and only if $|x| \geq t$. It is easy to see that $\deg(\mathsf{Thr}_t) = \Theta(N)$ for all nontrivial threshold functions, and thus $Q_E(\mathsf{Thr}_t) = \Omega(N)$. Paturi [43] shows that $\widetilde{\deg}(\mathsf{Thr}_t) = \Theta(\sqrt{(t+1)(N-t+1)})$, and we thus readily get that $Q_2(\mathsf{Thr}_t) = \Omega(\sqrt{(t+1)(N-t+1)})$, which is tight by quantum counting [14, 9]. This degree argument extends to any symmetric function F by writing F as a sum of threshold functions. The same tight lower bounds for symmetric functions can also be obtained by the unweighted adversary method (see the paragraph after Theorem 4).

For general non-symmetric functions, the polynomial method is, however, significantly harder to apply. For problems that are "close" to being symmetric, we can sometimes succeed in constructing a univariate or bivariate polynomial that yields a non-trivial lower bound. The first and, in our view, most important such a result was obtained by Aaronson in [1] in which he proves a lower bound of $\Omega(N^{1/5})$ on any bounded-error quantum algorithm for the collision problem.

The collision problem is a non-boolean promise problem. The oracle is an N-tuple of positive integers between 1 and M, which we think of as a function $X : [N] \to [M]$. We model the oracle O''_X so that a query to the i^{th} entry of the oracle returns the integer $X(i)$. Specifically, O''_X takes as input $|i, r; z\rangle$ and outputs $|i, r \oplus X(i); z\rangle$ where $0 \leq r < 2^m$ for $m = \lceil \log_2(M+1) \rceil$, and $r \oplus X(i)$ denotes bitwise addition modulo 2. We are promised that either X is a one-to-one function, or X is two-to-one, and the goal is to determine which is the case.

The result of Aaronson was shortly after improved by Shi [47] to $\Omega(N^{1/4})$ for general functions $X : [N] \to [M]$, and to $\Omega(N^{1/3})$ in the case the range is larger than the domain by a constant factor, $M \geq \frac{3}{2}N$. The lower bounds of Aaronson and Shi appears as a joint article [7]. Finally, Kutin [36] and Ambainis [6] independently found remedies for the technical limitations in Shi's proof, yielding an $\Omega(N^{1/3})$ lower bound for all functions, which is tight by an algorithm that uses Grover search on subsets by Brassard, Høyer, and Tapp [13].

The best lower bound for the collision problem that can be obtained using the adversary method is only a constant, since any one-to-one function is of large Hamming distance to any two-to-one function. Koiran, Nesme, and Portier [34] use the polynomial method to prove a lower bound of $\Omega(\log N)$ for Simon's problem [48], which is tight [48, 12]. Simon's problem is a partial boolean function

having properties related to finite abelian groups. Also for this problem, the best lower bound that can be obtained using the adversary method is a constant.

In contrast, for any *total* boolean function $F : \{0,1\}^N \to \{0,1\}$, the adversary and polynomial method are both polynomially related to block sensitivity,

$$\sqrt{\mathrm{bs}(F)/6} \leq \widetilde{\deg}(F) \leq \deg(F) \leq \mathrm{bs}^3(F) \tag{14}$$

$$\sqrt{\mathrm{bs}(F)} \leq \mathrm{Adv}(F) \leq \mathrm{bs}^2(F). \tag{15}$$

It follows from [19] that $\deg(F) \leq \mathrm{bs}^3(F)$, and from Nisan and Szegedy [41] that $6\widetilde{\deg}(F)^2 \geq \mathrm{bs}(F)$. Buhrman and de Wolf [19] provides an excellent survey of these and other complexity measures of boolean functions.

The polynomial lower bound is known to be inferior to the weighted adversary method for some total boolean functions. In [4], Ambainis gives a boolean function $F : \{0,1\}^4 \to \{0,1\}$ on four bits, which can be described as "the four input bits are sorted" [37], for which $\deg(F) = 2$ and for which there exists an adversary matrix Γ^F satisfying that $\lambda(\Gamma^F)/\max_i \lambda(\Gamma_i^F) = 2.5$. We compose the function with itself and obtain a boolean function $F_2 = F \circ (F,F,F,F) : \{0,1\}^{16} \to \{0,1\}$ defined on 16 bits for which $\deg(F_2) = 4$, and for which $\lambda(\Gamma^{F_2})/\max_i \lambda(\Gamma_i^{F_2}) = 2.5^2$, by the composition theorem. Iterating n times, yields a function F on $N = 4^n$ bits of degree $\deg(F) = 2^n$, with spectral lower bound $2.5^n = \deg(F)^{1.32\cdots}$, by the composition theorem. The thus constructed function F is an example of an iterated function of low degree and high quantum query complexity. It is the currently biggest known gap between the polynomial method and the adversary method for a total function. Another iterated total function for which the adversary methods yield a lower bound better than the degree, is the function described by "all three input bits are equal" [4].

The polynomial method is very suitable when considering quantum algorithms computing functions with error ϵ that is sub-constant, whereas the adversary method is not formulated so as to capture such a fine-grained analysis. Buhrman, Cleve, de Wolf, and Zalka [10] show that any quantum algorithm for Grover's problem that succeeds in finding an index i for which $x_i = 1$ with probability at least $1 - \epsilon$, provided one exists, requires $\Omega(\sqrt{N \log(1/\epsilon)})$ queries to the oracle. A possibly more familiar example is that any polynomial approximating the parity function with any positive bias $\epsilon > 0$ (as opposed to bias $\frac{1}{6}$ where $\frac{1}{6} = \frac{2}{3} - \frac{1}{2}$) has degree N, since any such polynomial gives rise to a univariate polynomial of no larger degree with N roots. Hence, any quantum algorithm computing the parity function with arbitrary small bias $\epsilon > 0$ requires $N/2$ queries to the oracle, which is tight.

A useful property of representing polynomials is that they compose. If p is a polynomial representing a function F, and polynomials $q_1, q_2, \ldots, q_k$ represent functions $G_1, \ldots, G_k$, then $p \circ (q_1, \ldots, q_k)$ represents $F \circ (G_1, \ldots, G_k)$, when well-

defined. This composition property does not hold for approximating polynomials: if each sub-polynomial q_i takes the value 0.8, say, then we cannot say much about the value $p(0.8, \ldots, 0.8)$ since the value of p on non-integral inputs is not restricted by the definition of being an approximating polynomial. To achieve composition properties, we require that the polynomials are insensitive to small variations of the input bits. Buhrman, Newman, Röhrig, and de Wolf give in [15] a definition of such polynomials, and refer to them as being robust.

Definition 12 (Robust polynomials [15]). *An approximate N-variate polynomial p is* robust *on $S \subseteq \{0,1\}^N$ if $|p(y) - p(x)| \leq \frac{1}{3}$ for every $x \in S$ and $y \in \mathfrak{R}^M$ such that $|y_i - x_i| \leq \frac{1}{3}$ for every $i = 1, \ldots, M$. The* robust degree *of a boolean function $F : S \to \{0,1\}$, denoted* $\mathrm{rdeg}(F)$*, is the minimal degree of a robust polynomial approximating F.*

Robust polynomials compose by definition. Buhrman et al. [15] show that the robust degree of any total function $F : \{0,1\}^N \to \{0,1\}$ is $O(N)$ by giving a classical algorithm that uses a quantum subroutine for Grover's problem [27] which is tolerant to errors, due to Høyer, Mosca, and de Wolf [29]. Buhrman et al. [15] also show that $\mathrm{rdeg}(F) \in O(\widetilde{\deg}(F) \log \widetilde{\deg}(F))$ by giving a construction for turning any approximating polynomial into a robust polynomial at the cost of at most a logarithmic factor in the degree of F. This implies that for any composite function $H = F \circ (G, \ldots, G)$, we have $\widetilde{\deg}(H) \in O(\widetilde{\deg}(F)\widetilde{\deg}(G) \log \widetilde{\deg}(F))$. It is not known whether this is tight. Neither is it known if the approximate degree of H can be significantly smaller than the product of the approximate degrees of F and G. The only known lower bound on the approximate degree of H is the trivial bound $\Omega(\widetilde{\deg}(F) + \widetilde{\deg}(G))$.

An and-or tree of depth two is a composed function $F \circ (G, \ldots, G)$ in which the outer function F is the logical AND of $\sqrt{N}$ bits, and the inner function G is the logical OR of $\sqrt{N}$ bits. By the unweighted adversary method, computing and-or trees of depth two requires $\Omega(\sqrt{N})$ queries. Høyer, Mosca, and de Wolf [29] give a bounded-error quantum algorithm that uses $O(\sqrt{N})$ queries, which thus is tight. The existence of that algorithm implies that there exists an approximating polynomial for and-or tree of depth two of degree $O(\sqrt{N})$. No other characterization of an approximating polynomial for and-or trees of depth two of degree $O(\sqrt{N})$ is currently known. The best known lower bound on the approximate degree of and-or trees of depth two is $\Omega(N^{1/3})$, up to logarithmic factors in N, by a folklore reduction from the element distinctness problem on $\sqrt{N}$ integers [7].

9 Concluding remarks

We have been focusing on two methods for proving lower bounds on quantum query complexity: the adversary method and the polynomial method. Adversary lower bounds are in general easy to compute, but are limited by the certificate complexity. Known lower bounds are constructed by identifying hard input pairs, finding weights accordingly, and computing either the spectral norm of some matrices, or applying the weighted method. Polynomial lower bounds may yield stronger bounds, but are hard to prove. Known lower bounds by the polynomial methods are constructed by identifying symmetries within the problem, reducing the number of input variables to one or two, and proving a lower bound on the degree of the reduced polynomial.

Barnum, Saks, and Szegedy give in [17] a third lower bound method that exactly characterizes the quantum query complexity, but this strength turns out also to be its weakness: it is very hard to apply and every known lower bound obtained by the method can also be shown by one of the other two methods. In a very recent work, Ambainis [5] extends the adversary method and uses it to reprove a strong direct product theorem by Klauck, Špalek, and de Wolf [35] obtained by techniques that include the polynomial method. Klauck et al. [35] show that their strong direct product theorem implies good quantum time-space tradeoffs, including a quantum lower bound of $T^2 \cdot S = \Omega(N^3)$ for sorting. A significant body of work have been conducted on lower bounds on communication complexity, primarily using the polynomial method. We refer to de Wolf's excellent survey [55] as a possible starting point.

There is a range of problems for which we do not currently know tight quantum query bounds. One important example is binary and-or trees of logarithmic depth. A binary and-or tree on $N = 4^n$ variables is obtained by iterating the function $F(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ in total n times. The classical query complexity for probabilistic algorithms is $\Theta(N^{0.753})$ [52, 49, 45]. No better bounded-error quantum algorithm is known. The best known lower bound on the quantum query complexity is $\Omega(\sqrt{N})$ by embedding the parity function on $\sqrt{N}$ bits and noting that the parity function has linear query complexity, which can be shown by either method.

Magniez, Santha, and Szegedy give in [40] a quantum algorithm for determining if a graph on N vertices contains a triangle which uses $O(N^{1.3})$ queries to the adjacency matrix. The best known lower bound is $\Omega(N)$ by the unweighted adversary method, and has been conjectured not to be tight [4]. The problem of triangle-identification is an example of a graph property, which is a set of graphs closed under isomorphism. Sun, Yao, and Zhang [53] show that there exists a non-trivial graph property of quantum query complexity $O(\sqrt{N})$, up to logarithmic factors in N.

Gasarch, in a survey on private information retrieval, published in this Computational Complexity Column in the Bulletin [26], writes: "A field is interesting if it answers a fundamental question, or connects to other fields that are interesting, or uses techniques of interest." It is our hope that the reader will find that thus surveyed area of quantum lower bounds fulfills each of those three criteria.

Acknowledgments

We thank Michal Koucký and Kolja Vereshchagin for discussions on the proof of the spectral adversary bound.

References

[1] S. Aaronson. Quantum lower bound for the collision problem. In *Proceedings of 34th ACM Symposium on Theory of Computing*, pages 635–642, 2002.

[2] S. Aaronson. Lower bounds for local search by quantum arguments. In *Proceedings of 36th ACM Symposium on Theory of Computing*, pages 465–474, 2004.

[3] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64:750–767, 2002.

[4] A. Ambainis. Polynomial degree vs. quantum query complexity. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 230–239, 2003.

[5] A. Ambainis. *A new quantum lower bound method, with an application to strong direct product theorem for quantum search.* quant-ph/0508200, 2005.

[6] A. Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1:37–46, 2005.

[7] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, 2004.

[8] H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[9] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001.

[10] H. Buhrman, R. Cleve, R. de Wolf, and Ch. Zalka. Bounds for small-error and zero-error quantum algorithms. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 358–368, 1999.

[11] R. Beigel. The polynomial method in circuit complexity. In *Proceedings of the 8th Annual Structure in Complexity Theory Conference*, IEEE Computer Society Press. pages 82–95, 1993.

[12] Gilles Brassard and Peter Høyer. An exact quantum polynomial-time algorithm for Simon's problem. In *Proceedings of Fifth Israeli Symposium on Theory of Computing and Systems*, pages 12–23, 1997.

[13] G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *SIGACT News*, 28:14–19, 1997.

[14] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, AMS Contemporary Mathematics Series, Volume 305, 2002.

[15] H. Buhrman, I. Newman, H. Röhrig, and R. de Wolf. Robust quantum algorithms and polynomials. In *Proceedings of 22nd International Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 3404, 593–604, 2005.

[16] H. Barnum and M. Saks. A lower bound on the quantum query complexity of read-once functions. *Journal of Computer and Systems Sciences*, 69(2):244–258, 2004.

[17] H. Barnum, M. Saks, and M. Szegedy. Quantum query complexity and semi-definite programming. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, pages 179–193, 2003.

[18] C. H. Bennett and S. J. Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69(20):2881–2884, 1992.

[19] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[20] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp. Quantum entanglement and the communication complexity of the inner product function. In *Proceedings of the 1st NASA QCQC conference*, Lecture Notes in Computer Science 1509, pages 61–74, 1998.

[21] Ch. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems. In *Proceedings of 31st International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 3142, pages 481–493, 2004.

[22] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society, London*, A439:553–558, 1992.

[23] M. Ettinger and P. Høyer, and E. Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43-48, 2004.

[24] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. *Invariant quantum algorithms for insertion into an ordered list*. quant-ph/9901059, 1999.

[25] L. Fortnow and J. D. Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999.

[26] W. Gasarch. A survey on private information retrieval. The computational complexity column, in the *Bulletin of the EATCS*, 82:72–107, 2004.

[27] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM Symposium on Theory of Computing*, pages 212–219, 1996.

[28] D. Gross. *The Future of Physics.* CERN Colloquium, January 26, 2005. CERN, Switzerland.

[29] P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. In *Proceedings of 30th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 2719, pages 291–299, 2003.

[30] P. Høyer, J. Neerbek, and Y. Shi. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica*, 34(4):429–448, 2002.

[31] A. S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973. English translation is *Problems in Information Transmission*, 9:177–183, 1973.

[32] P. Høyer and R. Špalek. *Tight adversary bounds for composite functions*. quant-ph/0509067, 2005.

[33] M. B. Jacokes, A. J. Landahl, and E. Brookes. *An improved quantum algorithm for searching an ordered list.* Manuscript, 2005.

[34] P. Koiran, V. Nesme and N. Portier. A quantum lower bound for the query complexity of Simon's problem. In *Proceedings of 32nd International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 3580, pages 1287–1298, 2005.

[35] H. Klauck, R. Špalek, and R. de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 12–21, 2004.

[36] S. Kutin. Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1:29–36, 2005.

[37] S. Laplante, T. Lee, and M. Szegedy. The quantum adversary method and formula size lower bounds. In *Proceedings of 20th IEEE Conference on Computational Complexity*, 2005.

[38] S. Laplante and F. Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. In *Proceedings of 19th IEEE Conference on Computational Complexity*, pages 294–304, 2004.

[39] R. Mathias. The spectral norm of a nonnegative matrix. *Linear Algebra and its Applications*, 139:269–284, 1990.

[40] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. In *Proceedings of 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 1109–1117, 2005.

[41] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. In *Proceedings of 24th ACM Symposium on Theory of Computing*, pages 462–467, 1992.

[42] A. Nayak and F. Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of 31st ACM Symposium on Theory of Computing*, pages 384–393, 1999.

[43] R. Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *Proceedings of the 24th ACM Symposium on Theory of Computing*, pages 468–474, 1992.

[44] K. Regan. Polynomials and combinatorial definitions of languages. In *Complexity Theory Retrospective II*, Springer-Verlag, pages 261–293, 1997.

[45] M. Santha. On the Monte Carlo decision tree complexity of read-once formulae. *Random Structures and Algorithms*, 6(1):75–87, 1995.

[46] Ch. Seife. "What are the limits of conventional computing?" *Science*, 309(5731):96, 1 July 2005.

[47] Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. In *Proceedings of the 43rd Annual Symposium on the Foundations of Computer Science*, pp. 513-519, 2002.

[48] D. R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.

[49] M. Snir. Lower bounds on probabilistic decision trees. *Theoretical Computer Science*, 38:69–82, 1985.

[50] M. Santha and M. Szegedy. Quantum and classical query complexities of local search are polynomially related. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 494–501, 2004.

[51] R. Špalek and M. Szegedy. All quantum adversary methods are equivalent. In *Proceedings of 32nd International Colloquium on Automata, Languages and Programming* , Lecture Notes in Computer Science 3580, pages 1299–1311, 2005.

[52] M. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating games trees. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 29–38, 1986.

[53] X. Sun, A. C. Yao, and S. Zhang. Graph properties and circular functions: How low can quantum query complexity go? In *Proceedings of 19th IEEE Conference on Computational Complexity*, pages 286–293, 2004.

[54] L. G. Valiant. Three problems in computer science. *Journal of Association of Computing Machinery*, 50(1):96–99, 2003.

[55] R. de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1): 337–353, 2002.

[56] Ch. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60:2746–2751, 1999.

[57] S. Zhang. On the power of Ambainis's lower bounds. In *Proceedings of 31st International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 3142, pages 1238–1250, 2004.

[58] S. Zhang. *(Almost) tight bounds for randomized and quantum local search on hypercubes and grids*. quant-ph/0504085, 2005.

The Concurrency Column

BY

Luca Aceto

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

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

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

Concurrency theory is home to a plethora of formalisms for the description of reactive systems, and even when focusing on a single model of concurrent computation there is a wide choice of possible notions of observable behaviour that can be used to induce a notion of "process equivalence" over it—as witnessed, for instance, by van Glabbeek's linear time-branching time spectrum. This multiplicity of possible semantic views of concurrent computation requires the development of tools that we can use to put some structure into it. I believe that expressiveness results play a fundamental role in concurrency theory, since they offer a powerful, formal way of understanding the relationships amongst models and formalisms for concurrent computation, and their relative computational power.

This contribution to the concurrency column studies the expressiveness of two approaches to the description of infinite behaviours in process calculi like ACP, CCS, CSP and various flavours of the π-calculus, namely *recursion* and *replication*. More specifically, Catuscia Palamidessi and Frank Valencia offer a unified presentation of work on the relative expressiveness of recursion and replication in CCS, the π-calculus, and the Ambient calculus. They also overview the work on this subject in less established calculi such as tcc and calculi for cryptographic protocols.

I trust that this piece will make some very interesting, recent work by Catuscia, Frank and others on the expressiveness of facilities for the description of infinite behaviours accessible to the concurrency theory community at large. Enjoy it!

As some of the readers of this column might have already noticed, I have been maintaining a Process Algebra Diary on the web for about eighteen months now. Those of you who are interested in following my irregular postings are invited to have a look at

http://www.cs.aau.dk/~luca/PA-DIARY/.

Comments and guest postings are most welcome.

Recursion vs Replication in Process Calculi: Expressiveness*

Catuscia Palamidessi
INRIA Futurs and LIX, École Polytechnique
catuscia@lix.polytechnique.fr

Frank D. Valencia
CNRS and LIX, École Polytechnique
fvalenci@lix.polytechnnique.fr

1 Introduction

Process calculi such as CCS [12], the π-calculus [14] and Ambients [6] are among the most influential formal methods for modelling and analyzing the behaviour of concurrent systems; i.e. systems consisting of multiple computing agents, usually called *processes*, that interact with each other. A common feature of these calculi is that they treat processes much like the λ-calculus treats computable functions. They provide a language in which the structure of *terms* represents the structure of processes together with an *operational semantics* to represent computational steps. Another common feature, also in the spirit of the λ-calculus, is that they pay special attention to economy. That is, there are few process constructors, each one with a distinct and fundamental role.

For example, a typical process term is the *parallel composition* $P \mid Q$, which is built from the terms P and Q with the constructor $\mid$ and it represents the process

*Supported by the Project Rossignol of the ACI Sécurité Informatique (Ministère de la recherche et nouvelles technologies)

that results from the parallel execution of the processes P and Q. Another typical term is the *restriction* $(\nu x)P$ which represents a process P with a private resource x—e.g., a location, a link, or a name. An operational semantics may dictate that if P can reduce to (or evolve into) P', written $P \longrightarrow P'$, then we can also have the reductions $P \mid Q \longrightarrow P' \mid Q$ and $(\nu x)P \longrightarrow (\nu x)P'$.

Infinite behaviour is ubiquitous in concurrent systems (e.g., browsers, search engines, reservation systems). Hence, it ought to be represented by process terms. Two standard term representations of them are *recursive process expressions* and *replication*.

Recursive process expressions are reminiscent of the recursive expressions used in other areas of computer science, such as for example Functional Programming. They may come in the form $\mu X.P$ where P may have occurrences of X. The process $\mu X.P$ behaves as P with the (free) occurrences of X replaced by $\mu X.P$. Another presentation of recursion is by using *parametric processes* of the form $A(y_1, \ldots, y_n)$ each assumed to have a unique, possibly recursive, *definition* $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$ where the x_i's are pairwise distinct, and the intuition is that $A(y_1, \ldots, y_n)$ behaves as its P with each y_i replacing x_i.

Replication, syntactically simpler than recursion, takes the form $!P$ and it is reminiscent of Girard's bang operator; an operator used to express unlimited number of copies of a given resource in linear-logic [8]. Intuitively, $!P$ means $P \mid P \mid \cdots$; an unbounded number of copies of the process P.

Now, it is not uncommon that a given process calculus, originally presented with one form of defining infinite behavior, is later presented with the other. For example, the π-calculus was originally presented with recursive expressions and later with replication [16]. The Ambient calculus was originally presented with replication and later with recursion [11]. This is reasonable as a variant may simplify the presentation of the calculus or be tailored to specific applications.

From the above intuitive description it should be easy to see that $\mu X.(P \mid X)$ expresses the unbounded parallel behaviour of $!P$. It is less clear, however, whether replication can be used to express the unbounded behaviour of $\mu X.P$. In particular, processes that allows for unboundedly many *nested* restrictions as, for example, in $\mu X.(\nu x)(P \mid X)$ which behaves as $(\nu x)(P \mid (\nu x)(P \mid (\nu x)(P \mid \cdots)))$. In fact, the ability of expressing recursive behaviours via replication depends on the particular process calculus under consideration.

The above discussion raises the issue of *expressiveness*. What does it mean for one variant to be as expressive as another ? The answer to this question is definite in the realm of computability theory via the notion of language equivalence. In concurrency theory, however, this issue is not quite settled.

One approach to comparing expressiveness of two given process calculus variants is by comparing them w.r.t. some standard process equivalence, say $\sim$. If for

every process P in one variant there is a Q in the other variant such that $Q \sim P$ then we say that the latter variant is at least as expressive as the former.

Another approach consists in telling two variants apart by showing that in one variant one can solve some fundamental problem (e.g., leader election) while in the other one cannot. It should be noticed that, unlike computability theory, the capability of two variants of simulating Turing Machines does not imply equality in their expressiveness. For example, [18] shows that under some reasonable assumptions the asynchronous version of the π-calculus, which can certainly encode Turing Machines, is strictly less expressive than the original calculus.

In this paper, we shall discuss the work on the relative expressiveness of Recursion and Replication in various process calculi. In particular, CCS, the π-calculus, and the Ambient calculus. We shall begin with the π-calculus, then CCS and then the Ambients calculus. For the simplicity of the presentation we shall consider the polyadic variant of the π-calculus [13]. Finally, we shall also overview the work on this subject in related calculi such as tcc [19] and calculi for Cryptographic Protocols [10].

2 The Polyadic Pi Calculus: pπ

One of the earliest discussions about the relative expressiveness between replication and recursion was in the context of the polyadic π-calculus [13]; one of the main calculi for mobility. It turns out that in this calculus replication is just as expressive as recursion. This results was rather surprising since replication seems such an elementary construct without much control power.

In what follows we shall introduce the polyadic π-calculus and the variants relevant for this paper. The various CCS and Ambients variants will be presented in the next sections as extension/restrictions of the polyadic π-calculus.

2.1 Finite Pi-calculus

Names are the most primitive entities in the π-calculus. We presuppose a countable set of (port, links or channel) *names*, ranged over by $x, y, \ldots$. For each name x, we assume a *co-name* $\overline{x}$ thought of as *complementary*, so we decree that $\overline{\overline{x}} = x$. We shall use $l, l', \ldots$ to range over names and co-names. We use $\vec{x}$ to denote a finite sequence of names $x_1 x_2 \cdots x_n$. The other entity in the π-calculus is a *process*. Process are built from names by the following syntax:

$$\begin{aligned} P, Q, \ldots &:= \sum_{i \in I} \alpha_i.P_i \;\mid\; (\nu x)P \;\mid\; P \mid Q \\ \alpha &:= \overline{x}\vec{y} \;\mid\; x(\vec{y}) \end{aligned} \tag{1}$$

where I is a finite set of indexes.

Let us recall briefly some notions as well as the intuitive behaviour of the various constructs.

The construct $\sum_{i \in I} \alpha_i.P_i$ represents a process able to perform one–but only one–of its α_i's actions and then behave as the corresponding P_i. The actions prefixing the P_i's can be of two forms: An output $\overline{x}y_1 \cdots y_n$ and an input $x(y_1 \cdots y_n)$. In both cases x is called the *subject* and $y_1 \cdots y_n$ the *object*. The action $\overline{x}\vec{y}$ represents the capability of sending the names $\vec{y}$ on channel x. The action $x(\vec{y})$, with $\vec{y} = y_1, \cdots, y_m$ and no name occurring twice in $\vec{y}$, represents the capability of receiving the names on channel x, say $z_1 \cdots z_m$, and replacing each y_i with z_i in its corresponding continuation.

Furthermore, in $x(\vec{y}).P$ the input actions binds the names $\vec{y}$ in P. The other name binder is the *restriction* $(\nu x)P$ which declares a name x private to P, hence bound in P. Given Q we define in the standard way its *bound names* $bn(Q)$ as the set of variables with a bound occurrence in Q, and its *free names* $fn(Q)$ as the set of variables with a non-bound occurrence in Q.

Finally, the process $P \mid Q$ denotes *parallel composition*; P and Q running in parallel.

Convention 2.1. *We write the summation as* 0 *if* $|I| = 0$, *and drop the "*$\sum_{i \in I}$*" if* $|I| = 1$. *Also we write* $\pi_1.P_1 + \cdots + \pi_n.P_n$ *for* $\sum_{i \in \{1,\ldots,n\}} \pi_i.P_i$.

For simplicity, we omit "()" in processes of the form $x().P$ *as well as the ".0" in processes of the form* $x(\vec{y}).0$. *We use* $(\nu x_1 x_2 \cdots x_n)P$ *as an abbreviation* $(\nu x_1)(\nu x_2) \cdots (\nu x_n)P$ *and* $\prod_{i \in I} P_i$, *where* $I = \{i_1, \ldots, i_n\}$, *as an abbreviation of* $P_{i_1} \mid \cdots \mid P_{i_n}$. *Furthermore,* $P\sigma$, *where* $\sigma = \{z_1/y_1, \ldots, z_n/y_n\}$, *denotes the process that results from the substitution in* P *of each* z_i *for* y_i, *applying* α*-conversion wherever necessary to avoid captures.*

Reduction Semantics of Finite Processes. The above intuition about process behaviour is made precise by the rules in Table 1. The *reduction* relation $\longrightarrow$ is the least binary relation on processes satisfying the rules in Table 1. The rules are easily seen to realize the above intuition.

We shall use $\longrightarrow^*$ to denote the reflexive, transitive closure of $\longrightarrow$. A reduction $P \longrightarrow Q$ basically says that P can evolve, after some communication between its subprocesses, into Q. The reductions are quotiented by the *structural congruence* relation $\equiv$ which postulates some basic process equivalences.

Definition 2.2 (Structural Congruence). *Let $\equiv$ be the smallest congruence over processes satisfying the following axioms:*

1. *$P \equiv Q$ if P and Q differ only by a change of bound names (α-equivalence).*
2. *$P \mid 0 \equiv P$, $P \mid Q \equiv Q \mid P$, $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$.*
3. *If $x \notin fn(P)$ then $(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q$.*
4. *$(\nu x)0 \equiv 0$, $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$.*

$$\text{REACT: } \frac{}{(\cdots + \overline{x}\, z_1 \cdots z_n.P) \mid (\cdots + x(y_1 \cdots y_n).Q) \longrightarrow P \mid Q\{z_1/y_1, \ldots, z_n/y_n\}}$$

$$\text{PAR: } \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \text{RES: } \frac{P \longrightarrow P'}{(\nu x)P \longrightarrow (\nu x)P'}$$

$$\text{STRUCT: } \frac{P \equiv P' \longrightarrow Q' \equiv Q}{P \longrightarrow Q}$$

Table 1: Reductions Rules.

2.2 Infinite Processes in the Polyadic Pi-Calculus

In the literature there are at least two alternatives to extend the above syntax to express infinite behavior. We describe them next.

Pi with Parametric Recursive Definitions: $\mathbf{p}\pi_{\mathbf{D}}$

A typical way of specifying infinite behavior is by using parametric recursive definitions [14]. In this case we extend the syntax of finite processes (Equation 1) as follows:

$$P, Q, \ldots := \ldots \mid A(y_1, \ldots, y_n) \tag{2}$$

Here $A(y_1, \ldots, y_n)$ is an *identifier* (also *call*, or *invocation*) of arity n. We assume that every such an identifier has a unique, possibly recursive, *definition* $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$ where the x_i's are pairwise distinct, and the intuition is that

$A(y_1, \ldots, y_n)$ behaves as its P with each y_i replacing x_i. We shall presuppose finitely many such definitions. Furthermore, for each $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$ we require

$$fn(P) \subseteq \{x_1, \ldots, x_n\}. \tag{3}$$

The reduction semantics of the extended processes is obtained simply by extending the structural congruence $\equiv$ in Definition 2.2 with the following axiom:

$$A(y_1, \ldots, y_n) \;\equiv\; P[y_1, \ldots, y_n / x_1, \ldots, x_n] \text{ if } A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P. \tag{4}$$

As usual $P[y_1 \ldots y_n / x_1 \ldots x_n]$ results from syntactically replacing every free occurrence of x_i with y_i and by applying *name α-conversion*, wherever needed to avoid capture.

We shall use $\mathrm{p}\pi_{\mathrm{D}}$ to denote the polyadic π-calculus with parametric recursive definitions with the above syntactic restrictions.

Pi with Replication: $\mathrm{p}\pi_{!}$

A simple way of expressing infinite behaviour in the π-calculus is by using replication. We shall use $\mathrm{p}\pi_{!}$ to denote the polyadic π-calculus with replication.

In the $\mathrm{p}\pi_{!}$ case, the syntax of finite processes (Equation 1) is extended as follows:

$$P, Q, \ldots := \ldots \mid\, !P. \tag{5}$$

Intuitively $!P$ behaves as $P \mid P \mid \ldots \mid P \mid !P$; unboundedly many copies of P.

The reduction semantics for $\mathrm{p}\pi_{!}$ is obtained simply by extending the structural congruence $\equiv$ in Definition 2.2 with the following axiom:

$$!P \;\equiv\; P \mid !P. \tag{6}$$

Barbed Bisimilarity

We shall often state expressiveness results by claiming the existence of a process in one calculus which is equivalent to some given process in another calculus. For this purpose, here we recall a standard way of comparing processes. We shall use $\mathrm{p}\pi_{!}$ to denote the calculus with replication.

Let us begin by recalling a basic notion of observation for the π-calculus. Intuitively, given $l = x$ ($l = \overline{x}$) we say that (the barb) l can be *observed* at P, written $P \downarrow_l$, iff P can have an input (output) with subject x. Formally,

Definition 2.3 (Barbs). *Define* $P \downarrow_{\overline{x}}$ *iff* $\exists \vec{z}, \vec{y}, R : P \equiv (\nu \vec{z})(\overline{x}\vec{y}.Q \mid R)$ *and* x *is not in* $\vec{z}$*. Similarly,* $P \downarrow_x$ *iff* $\exists \vec{z}, \vec{y}, Q, R : P \equiv (\nu \vec{z})(x(\vec{y}).Q \mid R)$ *and* x *is not in* $\vec{z}$*. Furthermore,* $P \Downarrow_l$ *iff* $\exists Q : P \longrightarrow^* Q \downarrow_l$ *.*

Let us now recall the notion of barbed (weak) bisimilarity and congruence. Remember that a process *context* C in a given calculus is an expression with a hole [.] such that placing a process in the hole produces a well-formed process term in the calculus.

For technical purposes, we shall use $\mathrm{p}\pi_{\mathrm{D}+!}$ as the calculus whose process syntax arises from extending the syntax of finite processes (Equation 1) with both replication and recursive definitions. The reduction semantics of $\mathrm{p}\pi_{\mathrm{D}+!}$ of the extended processes is obtained by extending the structural congruence $\equiv$ in Definition 2.2 with the axioms in Equations 4 and 6.

Definition 2.4 (Barbed Bisimilarity). *A* (weak) barbed-simulation *is a binary relation* $\mathcal{R}$ *satisfying the following:* $(P, Q) \in \mathcal{R}$ *implies that:*

1. *if* $P \longrightarrow P'$ *then* $\exists Q' : Q \longrightarrow^* Q' \wedge (P', Q') \in \mathcal{R}$*.*

2. *if* $P \downarrow_l$ *then* $Q \Downarrow_l$ *.*

The relation $\mathcal{R}$ *is a* barbed bisimulation *iff both* $\mathcal{R}$ *and its converse* $\mathcal{R}^{-1}$ *are barbed -simulations. We say that P and Q are* (weak) barbed bisimilar *iff* $(P, Q) \in \mathcal{R}$ *for some barbed bisimulation* $\mathcal{R}$*. Furthermore, we say that P and Q are barbed congruent, written* $P \approx Q$*, iff for each context* $C[\cdot]$ *in* $\mathrm{p}\pi_{\mathrm{D}+!}$*,* $C[P] \sim C[Q]$*.*

2.3 Recursive Definitions vs Replication in Pi

Here we recall a result stating that the variants $\mathrm{p}\pi_!$ and $\mathrm{p}\pi_{\mathrm{D}}$ can be regarded as being equally expressive w.r.t (weak) barbed congruence $\approx$ given in Definition 2.4. More precisely, the expressiveness criteria w.r.t to barbed congruence we shall use in this section can be stated as follows.

Criteria 2.5. *We say that a* π*-calculus variant is as expressive as another iff for every process P in the second variant one can construct a process* $[\![P]\!]$ *in the first variant such that* $[\![P]\!]$ *is (weakly) barbed congruent to P.*

All the results presented in this section are consequences of the expressiveness results in [20].

From $\mathrm{p}\pi_{\mathrm{D}}$ to $\mathrm{p}\pi_{!}$ and back: Encodings

We shall now provide encodings from one variant into the other and state their correctness. We shall say that a map $[\![]\!]$ is a *homomorphism for parallel composition* iff $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$. The notion of homomorphism for the other operators is defined analogously.

Definition 2.6. *Let $[\![\cdot]\!]_0$ be the map from $\mathrm{p}\pi_{\mathrm{D}}$ processes and recursive definitions into $\mathrm{p}\pi_{!}$ processes given by:*

$$\begin{array}{lcl} [\![0]\!]_0 & = & 0, \\ [\![A_i(\vec{x_i}) \stackrel{\mathtt{def}}{=} P_i]\!]_0 & = & !\, a_i(\vec{x_i}).[\![P_i]\!]_0, \\ [\![A_i(\vec{y_i})]\!]_0 & = & \overline{a_i}\vec{y_i}, \end{array}$$

and for all other processes $[\![\cdot]\!]_0$ is a homomorphism.

Let P be an arbitrary $\mathrm{p}\pi_{\mathrm{D}}$ process with $\{\, A_1(\vec{x_i}) \stackrel{\mathtt{def}}{=} P_1, \ldots, A_n(\vec{x_n}) \stackrel{\mathtt{def}}{=} P_n \,\}$ as the set of recursive definitions of its process identifiers. The encoding *of P, denoted $[\![P]\!]$, is defined as*

$$[\![P]\!] = (\nu a_1 \cdots a_n)([\![P]\!]_0 \mid \prod_{i \in \{1,\ldots,n\}} [\![A_i(\vec{x_i}) \stackrel{\mathtt{def}}{=} P_i]\!]_0)$$

where $a_1, \ldots, a_n \notin fn(P)$.

Intuitively, each $A(\vec{y})$, with $A(\vec{x}) \stackrel{\mathtt{def}}{=} P$, is translated into a particle $\overline{a}\vec{y}$ which excites a copy of P (with $\vec{y}$ substituted for $\vec{x}$) by interacting with a replicated resource, a provider of instances of P, of the form $!\, a(\vec{x}).[\![P]\!]$. The correctness of the encoding is stated below.

Theorem 2.7. *Let $[\![\cdot]\!]$ be the encoding in Definition 2.6. For each P in $\mathrm{p}\pi_{\mathrm{D}}$, $P \approx [\![P]\!]$.*

Let us now give an encoding of $\mathrm{p}\pi_{!}$ into $\mathrm{p}\pi_{\mathrm{D}}$. The idea is simple: Each $!P$ is translated into a process A_P, recursively defined as $A_P(\vec{x}) \stackrel{\mathtt{def}}{=} P \mid A_P(\vec{x})$ which can provide an unbounded number of copies of P.

Definition 2.8. *Let $[\![\cdot]\!]_0$ be the map from $\mathrm{p}\pi_{!}$ processes into $\mathrm{p}\pi_{!}$ processes given by:*

$$\begin{array}{lcl} [\![0]\!] & = & 0, \\ [\![!P]\!] & = & A_P(\vec{x}) \textit{ where } A_P(\vec{x}) \stackrel{\mathtt{def}}{=} P \mid A_P(\vec{x}) \textit{ and } fn(P) \subseteq \{\vec{x}\} \end{array}$$

and for all other processes $[\![\cdot]\!]_0$ is a homomorphism.

We can now state the correctness with respect to barbed congruence.

Theorem 2.9. *Let $[\![\cdot]\!]$ be the encoding in Definition 2.8. For each P in $\mathrm{p}\pi_{!}$, $P \approx [\![P]\!]$.*

2.4 Recursion vs Replication in the Private Pi Calculus

The Private π-calculus [20] is a sub-calculus with a restricted form of communication. The idea is that only *bound-outputs* are allowed; i.e, outputs of the form $(\nu\vec{z})\overline{x}\vec{z}.P$. Such bound-outputs are usually abbreviated as $\overline{x}(\vec{z})$ assuming that no name occur more than once in $\vec{z}$.

The above syntactic restriction results in a pleasant symmetry between input and outputs in that they both can be seen as binders. Moreover, the restriction ensures that α-conversion is the only kind of substitution required in the calculus. In fact, the rule REACT in Table 1, which applies a substitution to the continuation of the input, can be replaced by the following rule:

$$\overline{x}(\vec{z}).P \mid x(\vec{z}).P \longrightarrow (\nu\vec{z})(P \mid Q) \tag{7}$$

Let us denote by $\mathrm{Privp}\pi_!$ the calculus that results from applying to $\mathrm{p}\pi_!$ the syntactic restriction mentioned above. The $\mathrm{Privp}\pi_\mathrm{D}$ calculus is analogously defined as a restriction on $\mathrm{p}\pi_\mathrm{D}$ except that we need an extra-condition to ensure that α-conversion is the only substitution needed in the calculus: In every invocation $A(\vec{z})$, no name may occur more than once in the vector $\vec{z}$.

Now, if we wish an encoding $[\![\cdot]\!]$ from $\mathrm{Privp}\pi_!$ into $\mathrm{Privp}\pi_\mathrm{D}$ such that $[\![P]\!] \approx P$, we can simply take that of Definition 2.8 restricted to the $\mathrm{Privp}\pi_!$ case. As shown below, however, the above restriction makes impossible the existence of an encoding from $\mathrm{Privp}\pi_\mathrm{D}$ into $\mathrm{Privp}\pi_\mathrm{D}$.

Consider for example the process $P = A(z_0)$ where

$$A(x) \stackrel{\mathtt{def}}{=} \overline{x}(z).A(z).$$

The process P, in parallel with a suitable R, can perform a sequence of actions where the subject of an action is the object of the next one. This kind of sequences are called *logical threads* [20]. Moreover, P can perform the infinite logical thread $\overline{z_0}(z_1).\overline{z_1}(z_2).\ldots.$.

Interestingly, as an application of the type theory for $\mathrm{Privp}\pi_!$, the results in [20] state that *no process in* $\mathrm{Privp}\pi_!$ *can exhibit an infinite logical thread*. Together with P above, this property of $\mathrm{Privp}\pi_!$ can be used to prove the following result.

Theorem 2.10. *There is a process P in* $\mathrm{Privp}\pi_\mathrm{D}$ *such that $P \not\approx Q$ for every Q in* $\mathrm{Privp}\pi_!$.

Therefore, we cannot have an expressiveness result of the kind we have for $\mathrm{p}\pi_\mathrm{D}$ and $\mathrm{p}\pi_!$ in the previous section. I.e., there is no encoding $[\![\cdot]\!]$ from $\mathrm{Privp}\pi_\mathrm{D}$ processes into $\mathrm{Privp}\pi_!$ processes such that $[\![P]\!] \approx P$.

3 The Calculus of Communicating Systems (CCS)

Undoubtedly CCS [12], a calculus for synchronous communication, remains as a standard representative of process calculi. In fact, many foundational ideas in the theory of concurrency have sprung from this calculus. In the following we shall consider some variants of CCS without relabelling operations.

3.1 Finite CCS

The finite CCS processes can be obtained as a restriction of the finite processes of the Polyadic π-calculus by requiring all inputs and outputs to have empty subjects only. Intuitively, this means that in CCS there is no sending/receiving of links but synchronization on them. (Notice that the ability of transmitting names is used for the encoding of recursion into replication in Definition 2.6.) More, precisely, the syntax of finite CCS processes is obtained by replacing the second line of Equation (1) with

$$\alpha \;:=\; \overline{x} \;\big|\; x \;\big|\; \tau \tag{8}$$

where τ represents a distinguished action; the *silent* action, with the decree that $\overline{\tau} = \tau$.

The (unlabelled) reduction relation $\longrightarrow$ for finite CCS processes can be obtained from that for the π-calculus given in the previous section. However, since α-conversion does not hold for one of the CCS variants we consider next, we find it convenient to define $\longrightarrow$ in terms of labelled reduction of CCS given in Table 2. A transition $P \xrightarrow{\alpha} Q$ says that P can perform an action α and evolve into Q. The reduction relation is then defined as $\longrightarrow \stackrel{\texttt{def}}{=} \xrightarrow{\tau}$.

$$\text{SUM}\ \frac{}{\sum_{i\in I}\alpha_i.P_i \xrightarrow{\alpha_j} P_j}\ \text{if } j \in I \qquad \text{RES}\ \frac{P \xrightarrow{\alpha} P'}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'}\ \text{if } \alpha \notin \{x, \overline{x}\}$$

$$\text{PAR}_1\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \text{PAR}_2\ \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'} \qquad \text{COM}\ \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\overline{\alpha}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{RED}\ \frac{P \xrightarrow{\tau} Q}{P \longrightarrow Q}$$

Table 2: An operational semantics for finite CCS.

3.2 Infinite CCS Processes

Both recursion and replication are found in the CCS literature in the forms we saw for the polyadic π-calculus. Nevertheless, as recursion in CCS comes in other forms. Some forms of recursion exhibit *dynamic* name scoping while others, as in the π-calculus, have *static* name scoping. By dynamic scoping we mean that, unlike the static case, the occurrence of a name can get dynamically (i.e., during execution) captured under a restriction. Surprisingly, this will have an impact on their relative expressiveness.

In the literature there are at least four alternatives to extend the above syntax to express infinite behavior. We describe them next.

CCS with Parametric Definitions: $\mathrm{CCS_p}$

The processes of $\mathrm{CCS_p}$ calculus are the finite CCS processes plus recursion using parametric definition exactly as in $\mathrm{p}\pi_{\mathrm{D}}$. So in particular we have the restriction on parametric definitions in Equation 3. The calculus is the variant in [14]. The rules for $\mathrm{CCS_p}$ are those in Table 2 plus the rule:

$$\text{CALL}\ \frac{P_A[y_1,\ldots,y_n/x_1,\ldots,x_n] \xrightarrow{\alpha} P'}{A(y_1,\ldots,y_n) \xrightarrow{\alpha} P'}\ \text{if } A(x_1,\ldots,x_n) \stackrel{\mathtt{def}}{=} P_A \qquad (9)$$

As usual $P[y_1 \ldots y_n/x_1 \ldots x_n]$ results from syntactically replacing every free occurrence of x_i with y_i renaming bound names, i.e., *name α-conversion*, wherever needed to avoid capture. (Of course if $n = 0$, $P[y_1 \ldots y_n/x_1 \ldots x_n] = P$).

As shown in [14] in $\mathrm{CCS_p}$ we can identify process expression differing only by renaming of bound names; i.e., *name α-equivalence*—hence $(\nu x)P$ is the same as $(\nu y)P[y/x]$.

Constant Definitions: $\mathrm{CCS_k}$

We now consider the CCS alternative for infinite behavior given in [12]. We refer to identifiers with arity zero and their corresponding definitions as *constant* and *constant* (or *parameterless*) definitions, respectively. We omit the "( )" in $A($ $)$.

Given $A \stackrel{\mathtt{def}}{=} P$, requiring all names in $fn(P)$ to be formal parameters, as we did in $\mathrm{p}\pi_{\mathrm{D}}$ (Equation 3), would be too restrictive—P would not have visible actions. Consequently, let us drop the requirement to consider a fragment allowing *only* constant definitions but *with possible occurrence of free names in their bodies*. The rules for this fragments are those of $\mathrm{CCS_p}$. We shall refer to this fragment as $\mathrm{CCS_k}$. In this case Rule CALL, which for $\mathrm{CCS_k}$ we prefer to call CONS, takes the

form

$$\text{CONS}\ \frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'}\ \text{if } A \stackrel{\text{def}}{=} P \tag{10}$$

i.e., there is no α-conversion involved; thus allowing name captures. As illustrated in the next section, this causes scoping to be dynamic and α-equivalence not to hold. This is also the reason we cannot just take the reduction relation $\longrightarrow$ of the π-calculus restricted to CCS_k processes as such a relation assumes α-conversion due to the structural rule.

Recursion Expressions: CCS_μ

Hitherto we have seen process expressions whose recursive behavior is specified in an underlying set of definitions. It is often convenient, however, to have expressions which can specify recursive behavior on their own. Let us now extend the finite CCS processes to include such recursive expressions. The extended syntax is given by:

$$P, Q, \ldots := \ldots \mid X \mid \mu X.P \tag{11}$$

Here $\mu X.P$ binds the occurrences of the *process variable* X in P. As for bound and free names, the *bound variables* of P, $bv(P)$ are those with a bound occurrence in P, and the *free variables* of P, $fv(P)$ are those with a non-bound occurrence in P. An expression generated by the above syntax is said to be a *process (expression)* iff it is closed (i.e., it contains no free variables). The process $\mu X.P$ behaves as P with the free occurrences of X replaced by $\mu X.P$. Applying *variable* α-conversions wherever necessary to avoid captures. The semantics $\mu X.P$ is given by the rule:

$$\text{REC}\ \frac{P[\mu X.P/X] \xrightarrow{\alpha} P'}{\mu X.P \xrightarrow{\alpha} P'} \tag{12}$$

We call CCS_μ the resulting calculus. From [7] it follows that in CCS_μ we can identify processes up-to name α-equivalence.

Remark 3.1 (Static and Dynamic Scope: Preservation of α-Equivalence). An interesting issue of the substitution $[\mu X.P/X]$ applied to P is whether it *also requires* the renaming of *bound names* in P to avoid captures (i.e., *name α-conversion*). Such a requirement seems necessary should we want to identify process up-to *α-equivalence*. In fact, the requirement gives CCS_μ *static* scope of names. Let us illustrate this with an example.

Example 3.2. Consider $\mu X.P$ with $P = (x \mid (\nu x)(\overline{x}.t \mid X))$. First, let us assume we perform name α-conversions to avoid captures. So, $[\mu X.P/X]$ in P renames the bound X by a fresh name, say z, thus avoiding the capture of $P's$ free z in the replacement: I.e,

$$P[\mu X.P/X] = (x \mid (\nu z)(\overline{z}.t \mid \mu X.P)) = (x \mid (\nu z)(\overline{z}.t \mid \mu X.(x \mid (\nu x)(\overline{x}.t \mid X))))$$

The reader may care to verify (using the rules in Table 2 plus Rule REC) that t will not be performed; i.e., there is no $\mu X.P \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} \ldots$ s.t. $\alpha_i = t$.

Now let us assume that the substitution makes no name α-conversion, thus causing a free occurrence of x in P, shown in a box below, to get bound, *dynamically in the scope* of the outermost restriction: I.e.,

$$P[\mu X.P/X] = (x \mid (\nu x)(\overline{x}.t \mid \mu X.P)) = (x \mid (\nu x)(\overline{x}.t \mid \mu X.(\boxed{x} \mid (\nu x)(\overline{x}.t \mid X)))).$$

The reader can verify that now t can eventually be performed. Such an execution of t cannot be performed by $\mu X.Q$ where Q is $(x \mid (\nu z)(\overline{z}.t \mid X))$ i.e, P with the binding and bound occurrence of x syntactically replaced with z. This shows that name α-*equivalence does not hold in this dynamic scope case*. □

It should be pointed out that using recursive expressions with no name α-conversion is in fact equivalent to using instead constant definitions as in the previous calculus CCS_k. In fact, in presenting CCS, [12] uses alternatively both kinds of constructions; using Rule REC, with no name α-conversion, for one and Rule CONS for the other. For example, by taking $A \stackrel{\text{def}}{=} P$ with P as in Example 3.2 one can verify that in CCS_k, A exhibits exactly the same dynamic scoping behavior illustrated in the above example. So, *name* α-*equivalence does not hold in CCS*. Notice that the above observations imply some semantics differences between CCS and the π-calculus. The former does not satisfy name α-equivalence because of the dynamic nature of name scoping—see Example 3.2. The latter uses static scoping and satisfies α-equivalence. □

Replication: $\text{CCS}_!$

The processes of $\text{CCS}_!$ are those finite CCS processes plus replication exactly as in $\text{p}\pi_!$. This variant is presented in [2]. In the context of CCS, this operators are studied in [2, 3, 9].

The operational rules for $\text{CCS}_!$ are those in Table 2 plus the following rule:

$$\text{REP}\ \frac{P \mid\ !P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'} \tag{13}$$

From [14] we know that in $\text{CCS}_!$ one can identify processes up to name α-equivalence.

3.3 Expressiveness Results for CCS

In this section we report results from [2, 3, 9] on the expressiveness for the CCS variants above.

The following theorem summarizes the expressiveness of the various calculi and it is an immediate consequence of the results in [2] and [9]. As for the π-calculus we compare expressiveness w.r.t. barbed congruence with the obvious restriction to CCS contexts (see Criteria 2.5).

Theorem 3.3. *The following holds for the CCS variants:*

1. *CCS_{k} is exactly as expressive as CCS_{p} w.r.t to barbed congruence.*
2. *CCS_{μ} is exactly as expressive as $CCS_{!}$ w.r.t to barbed congruence.*
3. *The divergence problem (i.e., whether a given process P has an infinite sequence of $\longrightarrow$ reductions) is undecidable for the calculi in (1) but decidable for those in (2).*

The results (1-3) are summarized in Figure 1. Let us now elaborate on the significance and implications of the above results. A noteworthy aspect of (1) is that any finite set of parametric (possibly mutually recursive) definitions can be replaced by a set, *finite* as well, of parameterless definitions . This arises as a result of the restricted nature of communication in CCS (e.g., absence of mobility). Related to this result is that of [12] which shows that, in the context of value-passing CCS, a parametric definition can be encoded using an set of constant definitions and infinite sums. However, this set is *infinite*.

Regarding (1) some readers may feel that given a process P with a parametric definition D, one could simply create as many constant definitions as permutations of possible parameters w.r.t. the finite set of names in P and D. This would not work for $\mathrm{CCS}_{\mathrm{p}}$; the unfolding of call to D within a restriction may need α-conversions to avoid name captures, thus generating new names (i.e., names not in P nor D) during execution.

Regarding (2), we wish to recall the encoding $[\![\cdot]\!]$ of CCS_{μ} into $\mathrm{CCS}_{!}$ which resembles that of Definition 2.6 in the context of the π-calculus.

Definition 3.4. *The encoding $[\![\cdot]\!]$ of CCS_{μ} processes into $CCS_{!}$ is homomorphic over all operators in the sub-calculus defining finite behavior and is otherwise defined as follows:*

$$\begin{array}{lcl} [\![X_i]\!] & = & \overline{x_i} \\ [\![\mu X_i.P]\!] & = & (\nu x_i)(!x_i.[\![P]\!] \mid \overline{x_i}) \end{array}$$

where the names x_i's are fresh.

The above encoding is correct w.r.t. barbed congruence, i.e., $[\![P]\!] \approx P$. It is important to notice that it would not be correct had we adopted dynamic scoping in the Rule REC for CCS_k (see Remark 3.1). The $\mu X.P$ in Example 3.2 actually gives us a counter-example.

Another noteworthy aspect of the results mentioned above is the distinction between static and dynamic name scoping for the calculi under consideration. Static scoping renders the calculus with recursion decidable, *w.r.t. the divergence problem*, and no more expressive than the calculus with replication. In contrast, dynamic scoping renders the calculus with constant definitions undecidable and as expressive as that with parametric definitions. This is interesting since as discussed in Section 3.2 the difference between the calculi with static or dynamic scoping is very subtle. Using static scoping for recursive expressions was discussed in the context of ECCS [7], an extension of CCS whose ideas lead to the design of the π-calculus [14].

It should be noticed that preservation of divergence is not a requirement for equality of expressiveness w.r.t to barbed congruence since *barbed congruence does not preserve divergence*. Hence, although the results in [2] prove that divergence is decidable for $CCS_!$ (and undecidable for CCS_p), it does not follow directly from the arrows in Figure 1 that it is also decidable for CCS_μ. The decidability of the divergency problem for $CCS_!$ is proven in [9]

Finally, it is worth pointing out that, as exposed in [15], decidability of divergence does not imply lack of *Turing* expressiveness. In fact the authors in [3] show that $CCS_!$ is Turing-complete. They do this by showing how construct, given a two-counter machine, a process that can nondeterministically simulate such a machine. Two-counter machines are standard Turing-complete devices.

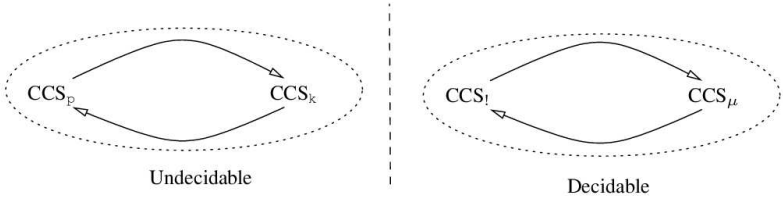


Figure 1: Classification of CCS variants. An arrow from X to Y indicates that for every P in Y one can construct a process $[\![P]\!]$ in X which is barbed congruent to P. (Un)decidability is meant w.r.t. the existence of divergent computations

4 The Mobile Ambients Calculus

The calculus of Mobile Ambients is a formalism for the description of distributed and mobile systems in terms of *ambients*; i.e. a named collection of active processes and nested sub-ambients.

The work in [4] studies the expressiveness of recursion versus replication in Mobile Ambients. In particular, the authors of [4] study the expressive power of ambient mobility in the (Pure) Mobile Ambients variants with replication and recursion.

4.1 Finite Processes of Ambients

The Pure Ambient Calculus focuses on ambient and processes interaction. Unlike the π-calculus, it abstracts away from process communication.

The syntax of the finite processes can be derived from those of the $\mathbf{p}\pi$-calculus by (1) introducing ambients, and the actions for ambient and processes interaction, (2) eliminating the action for process communication and (3) restricting summations to have arity at most one. In summary, we obtain the following syntax:

$$\begin{array}{rcl} P, Q, \ldots & := & 0 \mid \alpha.P \mid n[P] \mid (\nu x)P \mid P \mid Q \\ \alpha & := & in\ x \mid out\ x \mid open\ x \end{array} \tag{14}$$

The intuitive behaviour of the ambient $n[P]$ and α actions is better explained after presenting the reduction semantics of Ambients. The intuitive behaviour of the others constructs can be described exactly as in the π-calculus.

Reduction Semantics of Finite Processes. The *reduction* relation $\longrightarrow$ for Ambients can be obtained by adding the axiom $(\nu n)(m[P]) \equiv m[(\nu n)P]$ if $m \neq n$ to the structural congruence in Definition 2.2 and the following rules for ambients and process interaction to the rules of the $\mathbf{p}\pi$-calculus in Table 1:

1. $n[in\ m.P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$
2. $m[n[out\ m.P \mid Q] | R] \longrightarrow n[P \mid Q] \mid m[R]$
3. $open\ n.P \mid n[Q] \longrightarrow P \mid Q$
4. $\dfrac{P \longrightarrow Q}{n[P] \longrightarrow n[Q]}$

Rules (1-3) describe ambients and their actions and Rule (4) simply says that reduction can occur underneath ambients. Rule (1) describes how, by using the *in* action, an ambient named *n* can enter another ambient named *m*. Similarly, Rule (2) describes how an ambient named *n* can exit another ambient named *m* by using the *out* action. Finally Rule (3) describes how a process can dissolve an ambient boundary to access its contents by performing the *open* action over the name *n* of the ambient.

4.2 Infinite Process of Ambients

Infinite behaviour in Ambients can be represented by using replication as in $\mathrm{p}\pi_!$ or recursive expressions of the form $\mu X.P$.

The $MA_!$ calculus

The calculus $MA_!$ extends the syntax of the finite Ambients processes with $!P$. Its reduction semantics $\longrightarrow$ is obtained by adding the structural axiom $!P \equiv P \mid !P$ to the structural axioms of finite Ambients processes.

The MA_r calculus

The calculus MA_r extends the syntax of the finite Ambients processes with recursive expression of the form $\mu X.P$ exactly as in CCS_μ (Section 3.2). Its reduction semantics $\longrightarrow$ is obtained by adding the structural axiom $\mu X.P \equiv P[\mu X.P/X]$ to the structural axioms of finite Ambients processes.

Notice that the issue of the substitution $[\mu X.P/X]$ applied to P we discussed in Section 3.2 arises again: Whether the substitution *also requires* the renaming of *bound names* in P to avoid captures (i.e., *name α-conversion*). Such a requirement seems necessary should we want to identify process up-to *α-equivalence*–which is included in the structural congruence $\equiv$ for Ambients. The CCS examples in Section 3.2 (see Remark 3.1) can easily be adapted here to illustrate that we obtain dynamic scoping of names if we do not perform the α-conversion in the substitution.

It should be noticed that the above has not been completely clarified in the literature of Ambients. In fact, it raises a technical issue in the results on expressiveness which we shall recall in the next section.

Expressiveness Results

To isolate the expressiveness of restriction and ambient actions in $MA_!$ and MA_r, [4] considers the following fragments of MA_c with $c \in \{!, r\}$: (1) $MA_c^{-\nu}$, the MA_c

calculus without the restriction constructor $(\nu x)P$, (2) MA_c^{-mv}, the MA_c calculus without the *in* and *out* actions, and finally (3) $MA_c^{-mv,\nu}$, the corresponding calculus with no *in*/*out* action nor restriction.

The separation results in [4] among the various calculi are given in terms of the decidability of *termination*; i.e., the problem of whether given a process P does not have any infinite sequence of reductions. Obviously, if the question is decidable in a given calculus then we know that there is no termination-preserving encoding of Turing Machines into the calculus. The results in [4] are summarized in Figure 2.

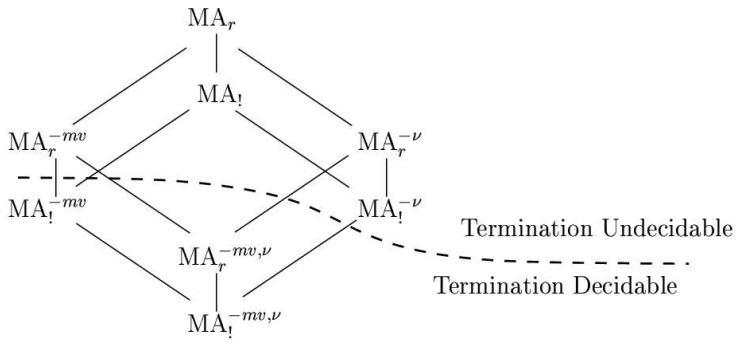


Figure 2: Hierarchy of Ambient Calculi.

Remark 4.1. The undecidability of process termination for MA_r^{-mv} is obtained by a reduction from termination of RAM machines, a Turing Equivalent formalism. First [4] uses a CCS fragment with recursion and *dynamic scope of names* to provide a termination-preserving encoding of RAMs. Then the CCS fragment is claimed to be a sub-calculus of MA_r^{-mv}. The undecidability of process termination for MA_r^{-mv} follows immediately.

Nevertheless, as illustrated in Section 3.2 Remark 3.1 such dynamic scope causes α-equivalence not to be preserved. In principle, this may cause a technical problem in the proof of the result since MA_r^{-mv} requires α-equivalence to be preserved; i.e., the CCS fragment used to simulate RAMs is not a sub-calculus of MA_r^{-mv}.

One way to deal with the above problem is to use a more involved notion of α-conversion in MA_r^{-mv} [5]. Another way would be to consider parametric recursion in MA_r, as in $\mathrm{CCS_p}$ or $\mathrm{p}\pi_\mathrm{D}$, and then use $\mathrm{CCS_p}$ as the sub-calculus of MA_r^{-mv} to encode RAMs. Nevertheless, either way we will be changing the original semantics of MA_r^{-mv} given in [11] which treats α-conversion and recursion as in CCS_μ [21].

5 Recursion vs Replication in Other Calculi

Here, we shall briefly survey work studying the relative expressive power of Recursion vs Replication in other process calculi.

In the context of calculi for security protocols, the work in [10] uses a process calculus to analyze the class of ping-pong protocols introduced by Dolev and Yao. The author show that all nontrivial properties, in particular reachability, become undecidable for a very simple recursive variant of the calculus. The recursive variant is capable of an implicit description of the active intruder, including full analysis and synthesis of messages . The authors then show that the variant with replication renders reachability decidable.

In the context of calculi for Timed Reactive System, the work in [17] studies the expressive power of some variants of Timed concurrent constraint programming (tcc). The tcc model is a process calculus introduced in [19] aimed at specifying timed systems, following the paradigms of Synchronous Languages [1]. The work states that: (1) recursive procedures with parameters can be encoded into parameterless recursive procedures with dynamic scoping, and vice-versa. (2) replication can be encoded into parameterless recursive procedures with static scoping, and vice-versa. (3) the languages from (1) are strictly more expressive than the languages from (2). Furthermore, it states that behavioral equivalence is undecidable for the languages from (1), but decidable for the languages from (2). The undecidability result holds even if the process variables take values from a fixed finite domain.

The reader may have noticed the strong resemblance of the work on tcc and that of CCS described in the previous section; e.g., static-dynamic scoping issue w.r.t recursion. In fact, [17] had a great influence in the work we described in this paper for CCS. In particular, in the discovery of the dynamic name scoping exhibited by the CCS presentation in [12].

6 Final Remarks

The expressiveness differences between recursion and replication we have surveyed in this paper may look surprising to those acquainted with the π-calculus where recursion is a derived operation. Our interpretation of this difference is that the link mobility of the π-calculus is a powerful mechanism which makes up for the weakness of replication.

The expressiveness of the replication $!P$ arises from unbounded parallel behaviour, which with recursion can be defined as $\mu X.(P \mid X)$. The additional expressive power of recursion arises from the unbounded nested scope of $\mu X.P$ as in $R = \mu X.(\nu x)(P \mid X)$ which behaves as $(\nu x)(P \mid (\nu x)(P \mid (\nu x)(P \mid \cdots)))$.

This, in general, cannot be simulated with replication. However, suppose that the unfolding of recursion applies α-conversion to avoid captures as we saw in Section 3.2. For example for the process R above we will have the unfolding $(\nu x_1)(P[x_1/x] \mid (\nu x_2)(P[x_2/x] \mid (\nu x_3)\cdots))$ and each x_i will only occur in $P[x_i/x]$. It is easy to see the replication $!(\nu x)P$ captures the behaviour of R. Therefore, R does not really exhibit (significant) unbounded nesting of scope.

All in all, the ability of expressing recursive behaviours via replication in a given process calculus may depend on the mechanisms of the calculus to compensate for the restriction of replication as well as on how meaningful the unbounded nesting of the recursive expressions are.

References

[1] G. Berry and G. Gonthier. The ESTEREL synchronous programming language: design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, November 1992.

[2] N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. recursive definitions in channel based calculi. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2003.

[3] N. Busi, M. Gabbrielli, and G. Zavattaro. Comparing recursion, replication, and iteration in process calculi. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2004.

[4] N. Busi and G. Zavattaro. On the expressive power of movement and restriction in pure mobile ambients. *Theoretical Computer Science*, 322(3):477–515, September 2004.

[5] N. Busi and G. Zavattaro. *Personal Communication*, May 2005.

[6] L. Cardelli and A. Gordon. Mobile Ambients. In M. Nivat, editor, *Proc. of Foundations of Software Science and Computation Structures (FoSSaCS), European Joint Conferences on Theory and Practice of Software (ETAPS'98)*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155, Lisbon, Portugal, 1998. Springer-Verlag, Berlin.

[7] U. Engberg and M. Nielsen. A calculus of communicating systems with label-passing. Technical report, University of Aarhus, 1986.

[8] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[9] P. Giambiagi, G. Schneider, and F. Valencia. On the expressiveness of infinite behavior and name scoping in process calculi. In *FoSSaCS*, pages 226–240, 2004.

[10] H. Huttel and J. Srba. Recursion vs. replication in simple cryptographic protocols. In *Proceedings of the 31st Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'05)*, volume 3381 of *LNCS*, pages 175–184. Springer-Verlag, 2005.

[11] Francesca Levi and Davide Sangiorgi. Mobile safe ambients. *ACM Transactions on Programming Languages and Systems*, 25(1):1–69, January 2003.

[12] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989. SU Fisher Research 511/24.

[13] R. Milner. The polyadic π-calculus: A tutorial. In F. L Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, Berlin, 1993.

[14] R. Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[15] S. Maffeis and I. Phillips. On the computational strength of pure ambient calculi. In *EXPRESS'03*, 2003.

[16] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Part I + II. *Information and Computation*, 100(1):1–77, 1992.

[17] M. Nielsen, C. Palamidessi, and F. Valencia. On the expressive power of concurrent constraint programming languages. In *Proc. of the 4th International Conference on Principles and Practice of Declarative Programming (PPDP 2002)*, pages 156–167. ACM Press, October 2002.

[18] C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous pi-calculus. In ACM Press, editor, *POPL'97*, pages 256–265, 1997.

[19] V. Saraswat, R. Jagadeesan, and V. Gupta. Foundations of timed concurrent constraint programming. In *Proc. of the Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 71–80, 4–7 July 1994.

[20] D. Sangiorgi and D. Walker. *The $\pi-$calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[21] D. Sangiorgi. *Personal Communication*, May 2005.

The Formal Specification Column

by

Hartmut Ehrig

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

Augur—A Tool for the Analysis of Graph Transformation Systems *

Barbara König Vitali Kozioura
Universität Stuttgart
Institut für Formale Methoden der Informatik
{koenigba,koziouvi}@fmi.uni-stuttgart.de

Abstract

We describe the tool Augur for the verification of systems with dynamically evolving structure specified by graph transformation. After giving a short introduction to graph transformation systems (GTSs), we describe the verification techniques used by the tool, namely the approximation of GTSs by Petri nets. Instead of verifying properties directly in the original system, they can be checked on the approximating Petri net. We explain the workings of the different modules of the Augur tool using two small case studies where we model reconfigurable networks and mobile processes.

1 Introduction

The idea behind Augur is to provide a tool to verify systems with dynamically evolving structure using suitable approximation techniques. Systems of this kind

*Research supported by DFG project SANDS.

appear in many places: as pointer structures on a program heap or as reconfigurable networks with mobile processes. They are characterized by the creation and deletion of objects and by changes in the system topology during runtime.

AUGUR takes as input language a simple yet expressive specification language: graph transformation systems (GTS) [18, 10]. Graph transformation systems extend static graph structures with the possibility to describe dynamic changes using transformation rules. They are well-suited to describe dynamic behavior, especially of concurrent and distributed systems.

GTSs are in general Turing-powerful and hence abstraction or over-approximation techniques are needed for the analysis of such systems. In our case we approximate GTSs by Petri nets, which are a conceptually simpler formalism and for which several verification techniques have already been developed. More specifically, the tool is based on an approximate unfolding technique for GTSs, presented in [3].

We are currently mainly interested in verifying that all reachable graphs satisfy certain structural properties. In the current implementation we support the specification of paths using regular expressions, where we check that such paths are absent in every reachable graph. These properties can be translated to coverability properties of the approximating Petri net. In order to check coverability for Petri nets we use standard algorithms, such as coverability graphs [13] and backward reachability [1].

If the obtained over-approximation is too coarse and does not allow to verify the property, techniques for refining the approximation are available. One such technique is counterexample-guided abstraction refinement which starts from a concrete counterexample found by coverability checking. Another possibility is to use depth-based refinement, which constructs an over-approximation exact up to a pre-defined depth in the unfolding.

We have conducted successful case studies such as the verification of mutual exclusion in an extending ring of processes [8] or the analysis of insertion of new elements into red-black trees [2].

2 Graph Transformation Systems

A graph transformation system consists of an initial graph and a set of rewriting rules. In order to obtain more flexibility we consider hypergraphs where an edge (a hyperedge) is connected to a sequence of nodes (instead of a pair of nodes as in a directed graph). The initial graph is a hypergraph describing the initial state of the system. Rewriting rules consist of two hypergraphs (left-hand side and right-hand side) and specify the possible dynamic transformations of the system. If an instance of the left-hand side is found in the current state of the system, then

this rule can be applied and the instance of the left-hand side of the rule will be replaced by its right-hand side. Embedding rules specify how this right-hand side is connected to the rest of the graph.

One of the most common approaches to graph rewriting is the DPO (double-pushout) approach, which derives its name from the fact that a rewriting step is described by two pushouts modelling the gluing of graphs. We are currently supporting restricted versions of DPO rules, where we only allow discrete interfaces, i.e., we can not describe the preservation of edges, and merging as well as deletion of nodes is forbidden. Edges, however, can be deleted. The extension to non-discrete interfaces is not very difficult from a theoretical point of view, whereas merging and deletion lead to more serious problems. Especially deletion means that we would have to handle negative application conditions, which can only be modelled using inhibitor arcs in Petri nets.

Other approaches to graph transformation (such as the single-pushout approach) could also be handled provided that the restrictions mentioned above are satisfied. For more information on graph transformation systems see [18, 9, 10]. Below we give an example of a very simple GTS meant to illustrate the main features of AUGUR.

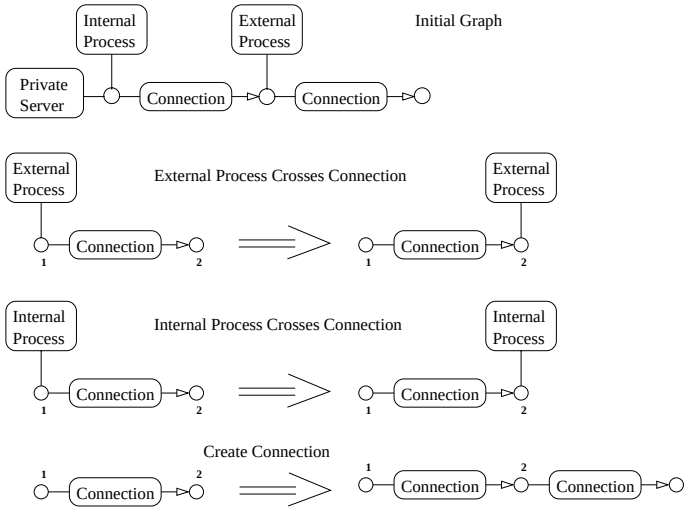


Figure 1: Example graph transformation system

In this system external and internal processes may cross connections and new connections can be created. This means we produce a tree-like structure of connections—starting with two connections—and let the mobile processes move non-deterministically along some branch of the tree. Transformations extending the

network and movement of processes can be interleaved. The initial graph consists of a private server with an internal process connected to it. Separated by one connection there is an external process.

In this example we plan to verify the following property: "An external process will never reach a private server", i.e., a hyperedge representing an external process and a hyperedge representing a private server will never share the same node.

3 Verification Techniques

We demonstrate the verification technique using the example of the previous section. To analyze this GTS the tool constructs an over-approximation, which is a so-called Petri graph (i.e., a hypergraph with a Petri net structure over it, see [3]). The hyperedges are at the same time the places of the net. For instance Fig. 2 shows the 0-depth (i.e., the coarsest) over-approximation of the GTS in Fig. 1. In Fig. 2 the small black rectangles and the arrows attached to them represent Petri net transitions, black dots represent the initial marking and the remaining structure depicts a hypergraph. Note that the places of the net coincide with the hyperedges of the graph.

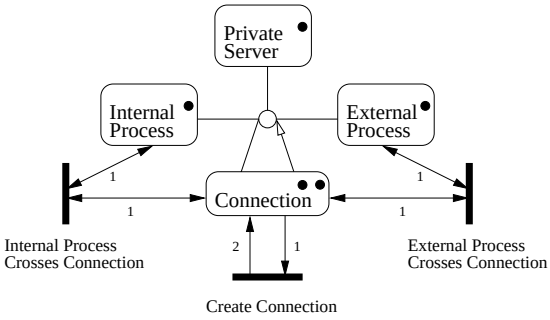


Figure 2: 0-depth approximation

This Petri graph is an over-approximation in the following sense: (i) every reachable graph can be mapped to its hypergraph component via a (usually non-injective) graph morphism and (ii) the multi-set image of its edges corresponds to a reachable marking of the net. More generally there exists a simulation relation between the reachable graphs and the reachable markings of the net. For a marking m we say that *m represents* a graph G whenever there is a mapping from G to the underlying hypergraph such that the number of edges mapped to a place agrees with the marking m.

Specifically, the initial marking represents the initial graph of the GTS (together with other graphs). Furthermore each transition is associated with a rule and over-approximates the effect of this rule when applied to a graph.

We do not describe here how the Petri graph is computed, apart from saying that the computation is based on an approximative unfolding algorithm. The algorithm is designed in such a way that nice properties of the GTS model, such as locality (state changes are only described locally) and concurrency (no unnecessary interleaving of events) are preserved in the approximating Petri net. More details can be found in [3, 5].

In this case the over-approximation is rather coarse. Observe specifically that *every* graph consisting of edges of the four types ("Private Server", "Connection", etc.) can be mapped to the underlying graph since all nodes have been merged into one. The only information we obtain via this approximation is the number of edges of a certain type. For instance the initial marking reports that in the initial graph there is one edge of type "Private Server", two edges of type "Connection", etc.

Since the approximation is too coarse, it is not possible to see whether a process has already crossed a certain connection and whether external processes may visit private servers. In fact the initial marking represents (in the sense defined above) graphs violating the property to be checked. It is also evident, that this counterexample is spurious, i.e., it has no counterpart in the original system since the "real" initial graph does not violate the property. In general there is a technique implemented in AUGUR telling the user if a counterexample is spurious, where a counterexample is a run of the Petri net producing a marking that represents graphs violating the property to be analyzed.

If some spurious counterexample is found, the over-approximation can be refined, which can be done in two different ways.

(i) One can change the level of accuracy (the depth) of the over-approximation.

(ii) One can construct a refined over-approximation by forbidding to merge certain nodes.

In our example we choose the second possibility, which usually leads to smaller Petri graphs. We take the counterexample obtained above and construct the refined over-approximation (see Fig. 3). The edges representing the private server and the external process are now separated (since the corresponding nodes have been separated) and the spurious counterexample found above is eliminated. It is also evident that no marking is coverable which represents a "bad" graph, i.e. a graph where an external process is connected to the private server . This can be shown using AUGUR, which means that the property can be successful verified in an automatic way.

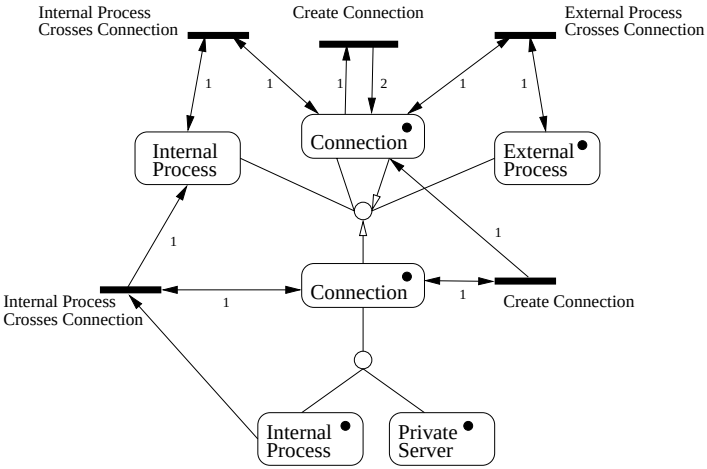


Figure 3: 1-depth approximation

4 System Description

In this section we look at AUGUR in more detail. We briefly describe the following modules of AUGUR: AUNFOLD, SPONGE, BWRA and ABSTREF (see Fig. 4 for an overview of AUGUR).

AUNFOLD is a module constructing the k-depth approximated unfolding (k-covering) for the given graph transformation system. As input and output format two XML-based standards are used (GTXL for describing graph transformation systems and the GXL for describing the over-approximating Petri graph obtained by the construction described in Section 3). For more details on GXL (Graph eXchange Language) and GTXL (Graph Transformation eXchange Language) see also [23, 21, 12]. The approximation is constructed according to the algorithms proposed in [3, 5].

Given a hypergraph and a regular expression r (describing "forbidden" paths), the module SPONGE generates a set M of markings with the following property: a marking m represents a graph containing a path corresponding to r if and only if m covers at least one marking of M. This information can be used in order to show that no graphs containing undesirable paths can be reached.

This task is performed by BWRA, which is a module using the backward reachability algorithm from [1] in order to determine the coverability of a marking. If the markings are not coverable, then the property to be verified is true, otherwise BWRA generates a counterexample.

Finally, module ABSTREF checks first if the counterexample found by BWRA

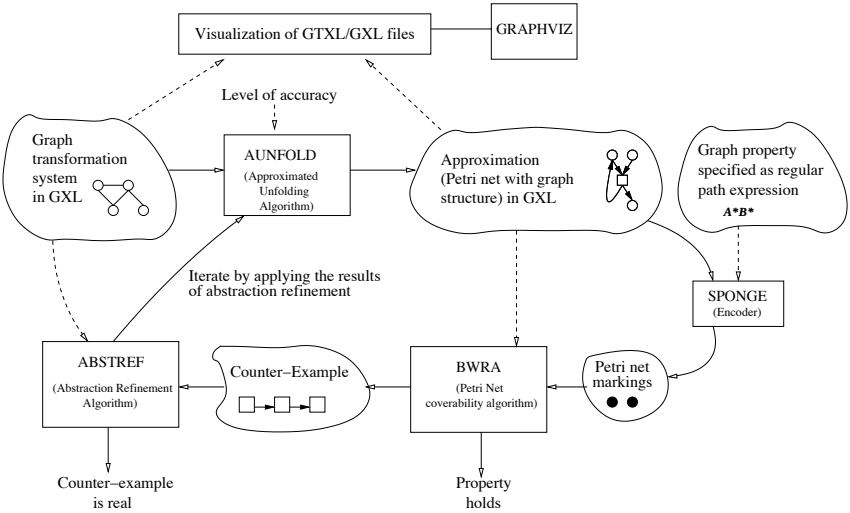


Figure 4: Overview of AUGUR

is real. If this is the case, then the property to be verified is false. Otherwise refinement conditions will be computed and the approximation procedure starts again taking into account all refinement conditions obtained earlier.

The procedure can be iterated in this way until either the property is verified, a counterexample is found or the pre-defined timeout is reached.

AUGUR also con'tains a visualization module, based on the GRAPHVIZ package. It can produce postscript files depicting graph transformation systems specified in GTXL and over-approximating Petri graphs represented as GXL files (see Fig. 5).

AUGUR can be obtained via `http://www.fmi.uni-stuttgart.de/szs/tools/augur`. We would like to encourage people who download and try the tool to report their problems and experiences.

5 An Extended Example

In this section we describe the verification of a more complex example and give some more details concerning the usage of AUGUR. We consider the system "Public/Private Servers" in Fig. 6 (see also example file `pub_priv_serv.xml` in the AUGUR distribution), which is an extension of the previous example. This system consists of public and private servers linked by network connections. Generators produce an unlimited number of public servers and one private server. The servers in turn produce mobile processes (internal processes by the private and external by

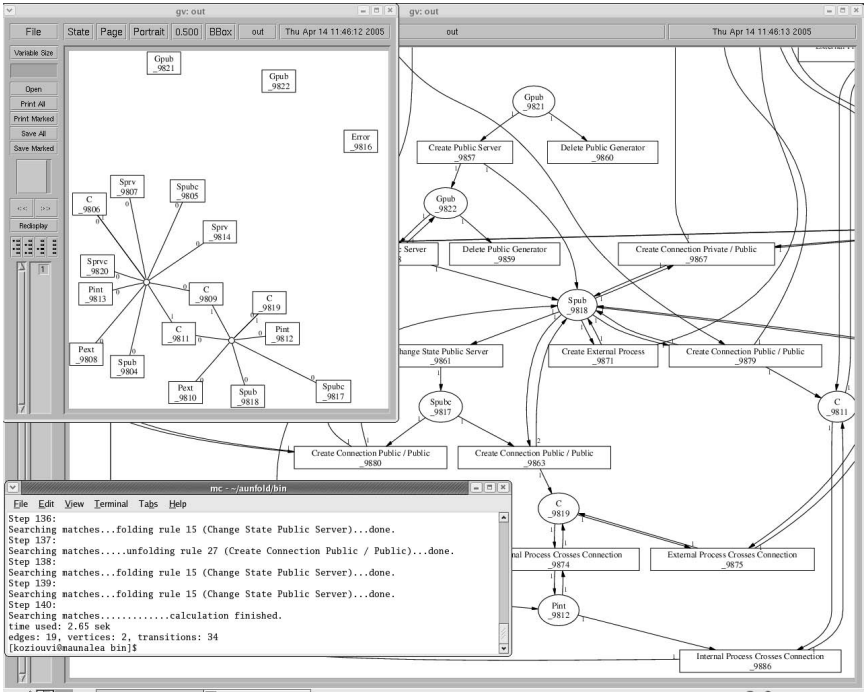


Figure 5: Screenshot of AUGUR at work

the public servers). New connections can be created between the servers, where however no connection is allowed from a private to a public server. Processes may cross these connections. Furthermore at some point in time the private server may decide to become a public server. The property we want to verify here is (as in the previous example): “No external process will ever reach a private server”.

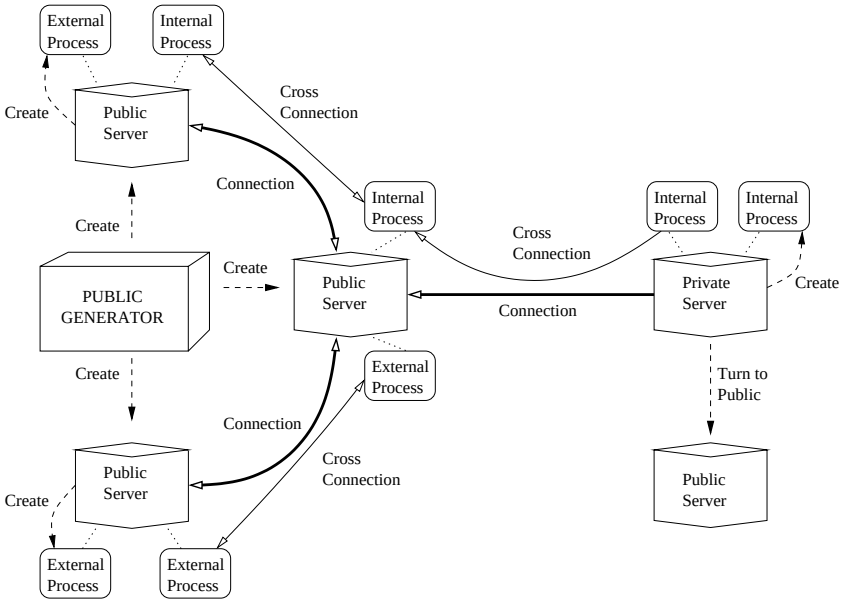


Figure 6: Example “Public/Private Servers”

The corresponding graph transformation systems consists of an initial graph containing only two generators and a set of rules describing the transformations schematically depicted in Fig. 6. The entire GTS is too large to be depicted in this paper.

In order to verify it, we first construct the approximated unfolding using AUNFOLD. NOW we can call SPONGE with the regular expression “`Private Server`” “`External Process`” in order to obtain a set of markings M with the following meaning: all markings which cover any marking of M are exactly the markings representing “bad” graphs, i.e., graphs which violate the property to be verified.

As in the previous example the 0-depth approximation is too coarse and BWRA tells us that indeed some markings contained in M are coverable in over-approximation and gives a counterexample. After checking the obtained counterexample with ABSTREF we see that this counterexample is spurious. Using ABSTREF one can

now construct the refined over-approximation as described earlier, which leads to successful verification.

Regular expression may also be more complex, for example `"Private Server" "Connection"* "External Process"` represents the property that no connection will ever be created from a public to private server. This property can also be verified after one refinement step.

Fig. 5 shows a screenshot of the verification of the "Public/Private Servers" system with AUGUR. The small window in the upper left corner shows the hypergraph underlying the approximation, whereas the large windows shows a part of the Petri net component. A step-by-step tutorial for AUGUR that shows how to verify this example is contained in [11].

6 Conclusion

In order to conclude we would like to mention and classify related work on the verification of graph transformation systems. While some research groups [22, 7] pursue the idea of translating graph transformation systems into the input language of a model checker, others attempt to develop new specialized methods for graph rewriting. Work from our side goes in this latter direction, as well as [15, 14, 16], which led to the tool GROOVE for verifying finite-state GTS. Although it is tempting to use existing optimized model checking tools, there is good reason for developing new techniques, even for finite state spaces. Existing tools usually do not directly support the creation (and deletion) of an arbitrary number of objects while still maintaining a finite state space, making entirely non-trivial their use for checking finite-state GTSs. A nice overview comparing these two fundamentally different approaches can be found in [17].

Further related work is on shape analysis [19], i.e., techniques which address directly the problem of verifying pointer structures. We have recently started to address the problem of encoding simple pointer-manipulating programs into graph rewriting, which will enable us to directly apply our techniques to a given piece of code.

Analysis techniques for GTSs are not restricted to reachability analysis and model checking. Other properties (such as termination and confluence via critical pair analysis) can be analyzed using the AGG tool [20].

Let us also mention that structural properties of graphs can not only be specified using regular expressions, but also using a monadic second-order graph logic. While the theory for this logic has already been established [6], we have not yet implemented the encoding of graph logic into properties of Petri net markings.

Finally, while the techniques presented in this paper are in principle also applicable to finite-state systems, it will usually be better to use specialized methods,

such as finite complete prefixes of unfoldings, a partial order representation for finite-state GTSs [4]. We have already started to extend our implementation in this direction.

Acknowledgements: We would like to thank Paolo Baldan, Andrea Corradini and Tobias Heindel for many interesting discussions.

References

[1] Parosh Aziz Abdulla, Bengt Jonsson, Mats Kindahl, and Doron Peled. A general approach to partial order reductions in symbolic verification. In *Proc. of CAV '98*, pages 379–390. Springer, 1998. LNCS 1427.

[2] Paolo Baldan, Andrea Corradini, Javier Esparza, Tobias Heindel, Barbara König, and Vitali Kozioura. Verifying red-black trees. In *Proc. of COSMICAH '05*, 2005. Proceedings available as report RR-05-04 (Queen Mary, University of London).

[3] Paolo Baldan, Andrea Corradini, and Barbara König. A static analysis technique for graph transformation systems. In *Proc. of CONCUR '01*, pages 381–395. Springer-Verlag, 2001. LNCS 2154.

[4] Paolo Baldan, Andrea Corradini, and Barbara König. Verifying finite-state graph grammars: an unfolding-based approach. In *Proc. of CONCUR '04*, pages 83–98. Springer-Verlag, 2004. LNCS 3170.

[5] Paolo Baldan and Barbara König. Approximating the behaviour of graph transformation systems. In *Proc. of ICGT '02 (International Conference on Graph Transformation)*, pages 14–29. Springer-Verlag, 2002. LNCS 2505.

[6] Paolo Baldan, Barbara König, and Bernhard König. A logic for analyzing abstractions of graph transformation systems. In *Proc. of SAS '03 (International Static Analysis Symposium)*, pages 255–272. Springer-Verlag, 2003. LNCS 2694.

[7] Fernando Luís Dotti, Luciana Foss, Leila Ribeiro, and Osmar Marchi Santos. Verification of distributed object-based systems. In *Proc. of FMOODS '03*, pages 261–275. Springer, 2003. LNCS 2884.

[8] Fernando Luís Dotti, Barbara König, Osmar Marchi dos Santos, and Leila Ribeiro. A case study: Verifying a mutual exclusion protocol with process creation using graph transformation systems. Technical Report 08/2004, Universität Stuttgart, 2004.

[9] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.2: Applications, Languages and Tools*. World Scientific, 1999.

[10] H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.3: Concurrency, Parallellism, and Distribution*. World Scientific, 1999.

[11] Barbara König and Vitali Kozioura. AUGUR—a tool for the analysis of graph transformation systems using approximative unfolding techniques, January 2005. Available from `http://www.fmi.uni-stuttgart.de/szs/tools/augur/documentation.ps`.

[12] Leen Lambers. A new version of GTXL: An exchange format for graph transformation systems. In *Proc. Workshop on Graph-Based Tools (GraBaTs'04)*, 2004.

[13] W. Reisig. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1985.

[14] Arend Rensink. Model checking graph grammars. In *Proc. of AVOCS '03 (Workshop on Automated Verification of Critical Systems)*, 2003.

[15] Arend Rensink. Canonical graph shapes. In *Proc. of ESOP '04*, pages 401–415. Springer, 2004. LNCS 2986.

[16] Arend Rensink. State space abstraction using shape graphs. In *Proc. of AVIS '04 (Third International Workshop on Automatic Verification of Infinite-State Systems)*, ENTCS, 2004. to appear.

[17] Arend Rensink and Dániel Varró. Model checking graph transformations: A comparison of two approaches. In *Proc. of ICGT '04*, pages 226–241. Springer, 2004. LNCS 3256.

[18] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, volume 1. World Scientific, 1997.

[19] Mooly Sagiv, Thomas Reps, and Reinhard Wilhelm. Parametric shape analysis via 3-valued logic. *TOPLAS (ACM Transactions on Programming Languages and Systems)*, 24(3):217–298, 2002.

[20] Gabriele Taentzer. AGG: A tool environment for algebraic graph transformation. In *Proc. of AGTIVE '99 (Applications of Graph Transformations with Industrial Relevance, International Workshop)*, pages 481–488. Springer, 1999. LNCS 1779.

[21] Gabriele Taentzer. Towards common exchange formats for graphs and graph transformation systems. In *Proc. of UniGra '01 (Uniform Approaches to Graphical Process Specification Techniques)*, volume 44.4 of *ENTCS*, 2001.

[22] Dániel Varró. Towards symbolic analysis of visual modeling languages. In *Workshop on Graph Transformation and Visual Modeling Techniques '02*, volume 72 of *ENTCS*. Elsevier, 2002.

[23] A. Winter. GXL—overview and current status. In *Proc. of GraBaTs '02 (Workshop on Graph-Based Tools)*, volume 72.2 of *ENTCS*, 2002.

Integration of the Generic Component Concepts for System Modeling with Adhesive HLR Systems

Julia Padberg
Technische Universität Berlin
Fakultät IV – Informatik und Elektrotechnik
padberg@cs.tu-berlin.de

Abstract

The integration of two well established theories is presented in this paper, namely the generic framework for components in system modeling and the adhesive HLR systems. The first theory describes components and composition at an abstract level that is independent of the specification technique. The second theory formalizes rules and transformations for an abstract replacement at the same abstract level.

The main results are the definition of a weak adhesive HLR category for components, its immediate results as Church-Rosser Theorem, Parallelism Theorem and Concurrency Theorem, and the new compatibility result for the transformation and hierarchical composition. We discuss the instantiation with deterministic automata, Petri nets and others. For these instantiations we immediately have the results mentioned above.

1 Introduction

The generic component concept for system modeling has been introduced and elaborated in a series of acknowledged papers [19, 20, 21, 22, 14, 36]. Their core motivation is to describe components independently of a specific specification technique. The main concepts are a self-contained semantics and the composition of components, based on a generic transformation concept. Here we use the categorical formalization as in [36] where pushouts to characterize the main construction. There have been quite different formal and semi-formal specification

techniques used within this framework, such as process algebras, Petri nets, UML and automata, being introduced in this paper.

Adhesive high-level replacements (HLR) systems [23] are an abstract theory for the transformation of objects of an arbitrary category in the style of the double pushout approach to graph transformations [9]. The main characteristic is that rules are given as a span of morphisms and transformations as two pushouts in a suitable category. The advantage of adhesive HLR systems is the level of abstraction. It is the same as for the generic components, namely the structural part (categorically spoken objects and arrows) and some assumptions are given, but the precise specification technique is left open.

The main result of this paper is that generic components form a weak adhesive HLR system, hence there are rules, transformations and various theorems hold; the Church-Rosser Theorem, the Parallelism Theorem and the Concurrency Theorem. These concern the sequential, parallel or concurrent application of rules. The transformation of components allows the change of a component by changing its interfaces and/or its body specification. Using components transformations, changes of components either during development or maintenance can be described formally. A formal description of change facilitates the complex task as there are precise conditions for correctness and consistency, as there is the possibility to define tools with a precise semantics, and as there is the possibility to model the change itself. The integration of the framework for generic components with adhesive HLR systems yields a thorough and extensive theory for the transformation of generic components. A detailed version of this paper including examples as well as the proofs is [37].

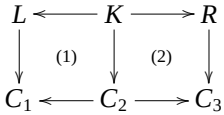
The paper continues with the review of adhesive HLR systems in section 2 and of the generic framework for components in system modeling in section 3. In section 4 we show that components yield a weak adhesive HLR category and state the compatibility of hierarchical composition with transformations. Then we discuss the instantiation with automata and with Petri nets in section 5. The conclusion summarizes the results and sketches some further instantiations.

2 Review of Adhesive HLR Systems

High-Level replacement (HLR) systems have been introduced in [16] as a generalization of the double pushout approach to graph transformations. Basically the replacement is carried out in an arbitrary category and not in the category of graphs. The basic notions remain more or less the same, so the notions of rules and transformations need an additional subclass $\mathcal{M}$ of monomorphisms.

Definition 2.1 (Rules and transformations).
A rule is given by $r = (L \leftarrow K \rightarrow R)$ where L and R are the left and right hand

side objects, K is an intermediate object,[1] and the morphisms $K \to L$ and $K \to R$ belong to a subclass of monomorphisms $\mathcal{M}$. Given a rule r and a context object C_2, we use morphisms $K \to L, K \to R$ and $K \to C_2$ to express a transformation as pushout constructions (1) and (2) leading to a double pushout as depicted below:

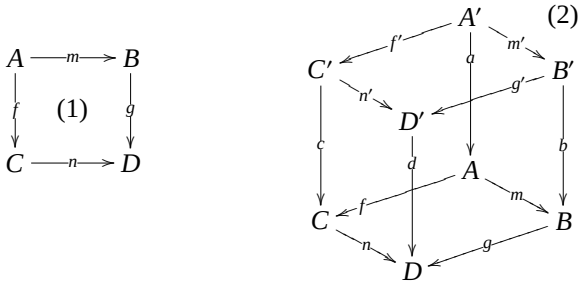


An application of a rule is called direct transformation and describes the change of an object by that application of the rule. A sequence of these rule applications yields a transformation. ◇

The theory of HLR systems has been developed as an abstract framework for different types of graph and Petri net transformation systems. Moreover the HLR framework has been applied to algebraic specifications [18], where the interface of an algebraic module specification can be considered as a production of an algebraic specification transformation system [15]. HLR systems are instantiated with various types of graphs, as hypergraphs, attributed and typed graphs, structures, algebraic specifications, various Petri net classes, elementary nets, place/transition nets, Colored Petri nets, or algebraic high-level nets, and more (see [16, 23]). Adhesive categories have been introduced in [32] and have been combined with HLR categories and systems in [17] leading to the new concept of (weak) adhesive HLR categories and systems. The main reason why adhesive categories are important for the theory of graph transformation and its generalization to high-level replacement systems is the fact that most of the HLR conditions required in [16] are shown to be already valid in adhesive categories (see [32]). The fundamental construct for (weak) adhesive (HLR) categotries and systems are van Kampen (VK) squares.

Definition 2.2 (van Kampen square).
A pushout (1) is a van Kampen (VK) square, if for any commutative cube (2) with (1) in the bottom and back faces being pullbacks, the follwowing holds:
the top is pushout ⇔ the front faces are pullbacks.

[1] In graph and net transformations K is often called the interface (of L and R) but in the context of components, this notion gets too confusing.



◇

(Weak) adhesive HLR systems can be considered as abstract graph transformation systems in the double pushout approach based on adhesive or weak adhesive HLR categories.

Definition 2.3 (Weak adhesive HLR category and system [23]).
A category **Cat** with a morphism class $\mathcal{M}$ is called weak adhesive HLR category (**Cat**, $\mathcal{M}$), if

1. $\mathcal{M}$ is a class of monomorphisms closed under isomorphisms and closed under composition ($f : A \to B \in \mathcal{M}$, $g : B \to C \in \mathcal{M} \Rightarrow g \circ f \in \mathcal{M}$) and decomposition ($g \circ f \in \mathcal{M}$, $g \in M \Rightarrow f \in \mathcal{M}$),

2. **Cat** has pushouts and pullbacks along $\mathcal{M}$-morphisms and $\mathcal{M}$-morphisms are closed under pushouts and pullbacks,

3. pushouts in **Cat** along $\mathcal{M}$-morphisms are weak VK squares, i.e. the VK square property holds for all commutative cubes with $m \in \mathcal{M}$ and ($f \in \mathcal{M}$ or $b, c, d \in \mathcal{M}$) see definition 2.2.

An adhesive HLR system $AHS = (\mathbf{Cat}, \mathcal{M}, P)$ consists of an adhesive HLR category (**Cat**, $\mathcal{M}$) and a set of rules P. ◇

Results 2.4 (for weak adhesive HLR systems).

Parallelism (chapter 5 in [23]) The Church-Rosser Theorem states a local confluence in the sense of formal languages. The Parallelism Theorem states that sequential or parallel independent transformations can be carried out either in arbitrary sequential order or in parallel. In the context of step-by-step development these theorems are important as they provide conditions for the independent development of different parts or views of the system.

Concurrency and pair factorization (chapter 5 in [23]) The Concurrency theorem handles general transformations, which may be non-sequentially independent. Roughly spoken, for a sequence there is a concurrent rule that allows the construction of a corresponding direct transformation.

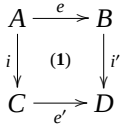
Embedding and local confluence (chapter 6 in [23]) Further important results for transformation systems are the Embedding, Extension and the Local Confluence Theorems. The first two allow to embed transformations into larger contexts and with the third one we are able to show local confluence of transformation systems based on the confluence of critical pairs.

3 Review of the Generic Component Concept

We now present the work concerning the generic concept of components in a categorical frame. In this framework a component consists of an import, an export and the body. The import states the prerequisites the component assumes. The body represents the internal functionality. The export gives an abstraction of the body that can be used by the environment. In [36] we present a categorical formalization of the concepts of the generic framework using specific kinds of pushout properties.

Definition 3.1 (Generic framework $\mathcal{T}$ for components [36]).
A generic framework for components $\mathcal{T} = (\mathbf{Cat}, \mathcal{I}, \mathcal{E})$ consists of an arbitrary category and two classes of morphisms $\mathcal{I}$, called import morphisms and $\mathcal{E}$, called export morphisms such that the following *extension conditions* hold:

1. $\mathcal{E}$-$\mathcal{I}$-Pushout Condition:
   Given the morphisms $A \xrightarrow{e} B$ with $e \in \mathcal{E}$ and $A \xrightarrow{i} C$ with $i \in \mathcal{I}$, then there exists the pushout D in **Cat** with morphisms $B \xrightarrow{i'} D$ and $C \xrightarrow{e'} D$ as depicted adjacently.



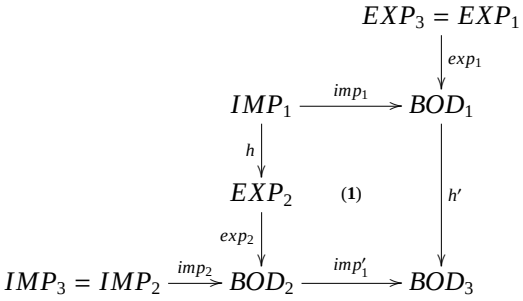
2. $\mathcal{E}$ and $\mathcal{I}$ are stable under pushouts:
   Given a $\mathcal{E}$-$\mathcal{I}$-pushout as (**1**) above, then we have $i' \in \mathcal{I}$ and $e' \in \mathcal{E}$ as well. ◇

Accordingly, we have to require for a component that the import and export connection are of the right class of morphisms.

Definition 3.2 (Component [36]).
A component $C = (IMP, EXP, BOD, imp, exp)$ is given by objects IMP, EXP, and BOD in **Cat** and by morphisms $exp : EXP \to BOD$ and $imp : IMP \to BOD$, so that $exp \in \mathcal{E}$ and $imp \in \mathcal{I}$. ◇

Subsequently we consider the basic operation that allows composing components C_1 and C_2 hierarchically. As it provides a connection $h : IMP_1 \to EXP_2$ from the import interface IMP_1 of C_1 to the export interface EXP_2 of C_2, we are now able to define the composition $C_3 = C_1 \circ_h C_2$ as follows.

Definition 3.3 (Composition [36]).
Given components $C_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i)$ for $i \in \{1, 2\}$ and a morphism $h : IMP_1 \to EXP_2$ in $\mathcal{E}$ the composition C_3 of C_1 and C_2 via h is defined by $C_3 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3)$ with $imp_3 = imp'_1 \circ imp_2$ and $exp_3 = h' \circ exp_1$ as depicted below,



where (**1**) is pushout diagram in the category **Cat**.
The composition is denoted by $C_3 = C_1 \circ_h C_2$. ◇

4 The Adhesive HLR System for Generic Components

In this section we sketch how the two theories can be integrated. The benefit is that the notions of rules and transformations as well as the corresponding results can be carried over to components. The basic idea is that the transformation of each part of the component, i.e. the export, the import, and the body, is a transformation in the underlying specification category. Naturally, the specification category has to be weakly adhesive. As the definition of components involves different classes of morphisms these need to be taken into consideration. The difficulty to establish transformations of components is directly dependent on the class of refinement morphisms. So first we need to investigate involved morphism classes. As the morphisms needed for the adhesive HLR system are usually simpler than the refinement morphisms, we have defined the class of refinement morphisms forming a supercategory of the morphisms used for the construction of rules and transformations. Therefore we have to extend the approach in [36] by relating the morphism classes used for the transformation and the components. This leads to the adhesive HLR framework for generic components.

Definition 4.1 (Adhesive HLR framework for generic components).
The adhesive HLR framework for generic components $\mathcal{A} = (\mathbf{Cat_p}, \mathbf{Cat_r}, \mathcal{M})$ consists of

1. $\mathbf{Cat_p}$ the category of specifications with plain morphisms

2. $\mathbf{Cat_r}$ the category of specifications with refinement morphisms is a super-category of $\mathbf{Cat_p}$ as the functor $\mathsf{Inc} : \mathbf{Cat_p} \to \mathbf{Cat_r}$ is an inclusion in the sense that $Obj_{\mathbf{Cat_p}} = Obj_{\mathbf{Cat_r}}$.

3. $(\mathbf{Cat_p}, \mathcal{M})$ is a weak adhesive HLR category

4. $\mathbf{Cat_r}$ has pushouts where at least one morphism is in $\mathsf{Inc}(Mor_{\mathbf{Cat_p}})$.

5. the inclusion functor $\mathsf{Inc} : \mathbf{Cat_p} \to \mathbf{Cat_r}$ preserves pushouts where at least one morphism is in $\mathcal{M}$.

6. the inclusion functor $\mathsf{Inc} : \mathbf{Cat_p} \to \mathbf{Cat_r}$ preserves pullbacks where at least one morphism is in $\mathcal{M}$.

◇

We first relate the adhesive HLR framework for generic components to the generic framework given in definition 3.1. Then we choose the import morphisms to be plain morphisms, i.e. $\mathcal{I} = \mathsf{Inc}(Mor_{\mathbf{Cat_p}})$, and the export morphisms to be refinement morphisms, i.e. $\mathcal{E} = Mor_{\mathbf{Cat_r}}$.

Fact 4.2 (Relation of Frameworks).
Given an adhesive HLR framework for generic components $(\mathbf{Cat_p}, \mathbf{Cat_r}, \mathcal{M})$, there is the framework for generic components $\mathcal{T} = (\mathbf{Cat_r}, Inc(Mor_{\mathbf{Cat_p}}), Mor_{\mathbf{Cat_r}})$. ◇

Due to item 4 of definition 4.1.

So, we have components as in definition 3.2 and hierarchical composition as in definition 3.3. Next we define the category of components **Comp**, for the definition of component morphisms we use plain morphisms, that have to be compatible with the corresponding import and export morphisms.

Definition 4.3 (Component Category).
For components $C_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i)$, $i = 1, 2$, the component morphisms are defined by $comp : C_1 \to C_2$ with $comp = (comp_I, comp_B, comp_B)$ s.t.

$comp_I : IMP_1 \to IMP_2$
$comp_B : BOD_1 \to BOD_2$
$comp_E : EXP_1 \to EXP_2$

where $comp_I, comp_B, comp_B \in Mor(\mathbf{Cat_p})$

1. $comp_B \circ imp_1 = imp_2 \circ comp_I$

2. $comp_B \circ exp_1 = exp_2 \circ comp_E$

Components and component morphisms constitute the category of components **Comp**. ◇

Subsequently we show that the category of components **Comp** together with $\mathcal{M}_C = \{comp = (comp_I, comp_B, comp_E) | comp_I, comp_B, comp_E \in \mathcal{M}\}$ is an adhesive HLR category. The following facts (see [37]) are important for the proof of Theorem 4.4: There are pushouts with at least one $\mathcal{M}_C$-morphism in **Comp** and there are pullbacks with at least one $\mathcal{M}_C$-morphism in **Comp**.

Theorem 4.4 ((Comp, $\mathcal{M}_C$) is weak adhesive HLR category).
Given the adhesive HLR framework for generic components $\mathcal{A} = (\mathbf{Cat_p}, \mathbf{Cat_r}, \mathcal{M})$ then (**Comp**, $\mathcal{M}$) is weak adhesive HLR category with
$\mathcal{M}_C = \{comp = (comp_I, comp_B, comp_E) | comp_I, comp_B, comp_E \in \mathcal{M}\}$. ◇

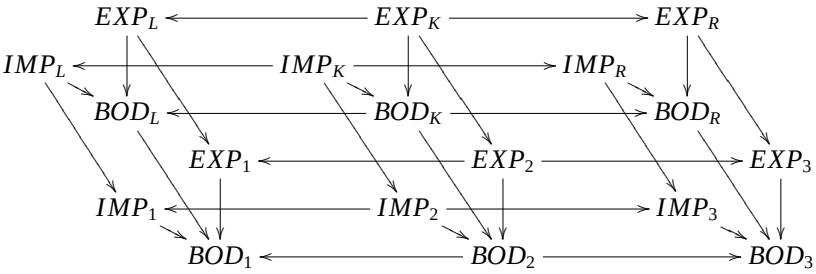
For the proof see [37].

So we can define an adhesive HLR system for components and have the important following results for the transformation of components.

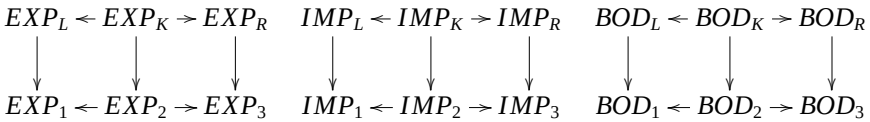
Results 4.5 (for an weak adhesive HLR system of components).
Given the adhesive HLR framework for generic components $\mathcal{A} = (\mathbf{Cat_p}, \mathbf{Cat_r}, \mathcal{M})$ then we have the weak adhesive HLR system of components (**Comp**, $\mathcal{M}_C, P$) where P is a set of rules and we have the results as given in 4.5:
Parallelism results,
Concurrency and pair factorization and
Embedding and local confluence.

Up to now, the instantiations of the HLR theory have been specification techniques as various types of graph transformations, Petri nets, algebraic specifications, etc. Now we instantiate with a generic component construction upon these specification techniques. Hence it is obvious that new questions of compatibility arise. especially the question, whether the operations at the level of components are compatible with the transformation concept. Subsequently we show the conditions under which transformations and hierarchical composition are compatible. The proofs are quite lengthy and in principle they are based on the compatibility of colimit constructions with commutative diagrams, so we omit them here and refer to [37].

Definition 4.6 (Rules and Transformations).
Given a rule in **Comp** with $r = (C_L \leftarrow C_K \rightarrow C_R)$ then the application of r yields the transformation $C_1 \overset{r}{\Longrightarrow} C_3$ given by the following diagram in $\mathbf{Cat_r}$:



with the following double pushouts in $\mathbf{Cat_p}$



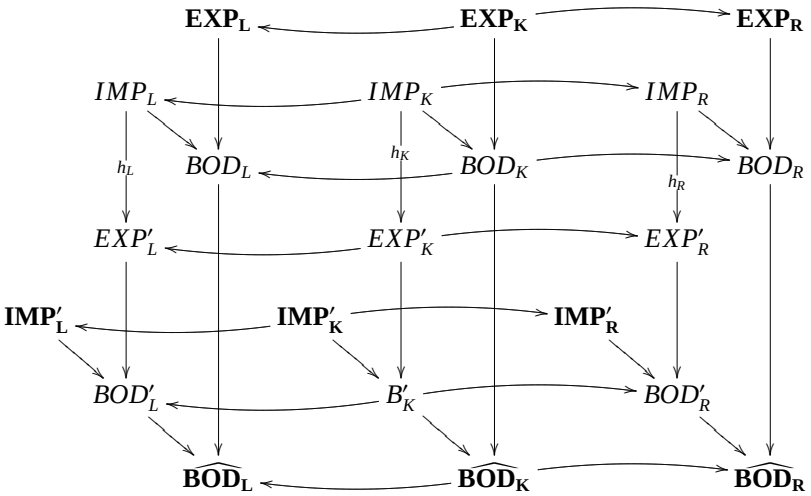
◇

Fact 4.7 (Hierarchical composition of rules).
Given the rules $r = (C_L \leftarrow C_K \rightarrow C_R)$ and $r' = (C'_L \leftarrow C'_K \rightarrow C'_R)$ in the category **Comp** for $h_L : IMP_L \rightarrow EXP'_L$, $h_K : IMP_K \rightarrow EXP'_K$, and $h_R : IMP_R \rightarrow EXP'_R$ so that

1. $IMP_K \rightarrow IMP_L \overset{h_L}{\rightarrow} EXP'_L = IMP_K \overset{h_K}{\rightarrow} EXP'_K \rightarrow EXP'_L$

2. $IMP_K \rightarrow IMP_R \overset{h_R}{\rightarrow} EXP'_R = IMP_K \overset{h_K}{\rightarrow} EXP'_K \rightarrow EXP'_R$

as depicted in the following diagram:



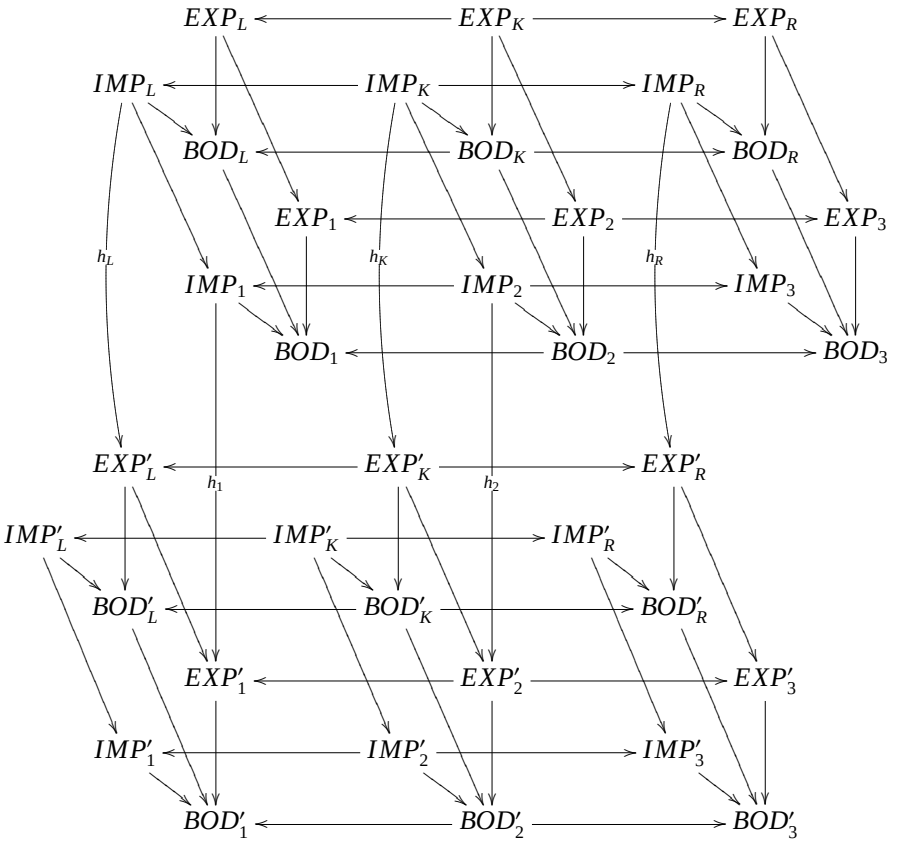
then we compose hierarchically $\widehat{r} := r \circ_h r' = (\widehat{C_L} \leftarrow \widehat{C_K} \rightarrow \widehat{C_R})$ with $h = (h_l, h_K, h_R)$ where
$\widehat{C_L} = (IMP'_L \rightarrow \widehat{BOD}_L \leftarrow EXP_L)$,
$\widehat{C_K} = (IMP'_K \rightarrow \widehat{BOD}_K \leftarrow EXP_K)$, and
$\widehat{C_R} = (IMP'_R \rightarrow \widehat{BOD}_R \leftarrow EXP_R)$ ◇

Due to adhesive HLR framework for generic components in definition 4.1. The definition below characterizes the conditions for the compatibility of component composition and component transformation: both rules are applicable and the corresponding connecting morphisms are compatible, i.e. we have commutative squares.

Definition 4.8 (Independence of transformation and composition).
Given the rules $r = (C_L \leftarrow C_K \rightarrow C_R)$ and $r' = (C'_L \leftarrow C'_K \rightarrow C'_R)$ with $r \circ_h r'$ with $h = (h_L, h_K, h_R)$ for $h_L : IMP_L \rightarrow EXP'_L$, $h_K : IMP_K \rightarrow EXP'_K$, and $h_R : IMP_R \rightarrow EXP'_R$ then the composition $C_1 \circ_{h_1} C'_1$ is independent from r and r' if there is $h_2 : IMP_2 \rightarrow EXP'_2$ s.t.

1. $IMP_2 \xrightarrow{h_2} EXP'_2 \rightarrow EXP'_1 = IMP_2 \rightarrow IMP_1 \xrightarrow{h_1} EXP'_1$
2. $IMP_L \rightarrow IMP_1 \xrightarrow{h_1} EXP'_1 = IMP_L \xrightarrow{h_L} EXP'_L \rightarrow EXP'_1$
3. $IMP_K \rightarrow IMP_2 \xrightarrow{h_2} EXP'_2 = IMP_K \xrightarrow{h_K} EXP'_K \rightarrow EXP'_2$

as in the diagram below:



◇

If there are two components and two rules to be applied we can either compose two components and then use a composed rule to transform the component or we transform the two components independently and compose the results of the composition.

The following theorem states that under independence both ways result in the same component (up to isomorphism).

Theorem 4.9 (Composition Theorem).
Let the rules $r = (C_L \leftarrow C_K \rightarrow C_R)$ and $r' = (C'_L \leftarrow C'_K \rightarrow C'_R)$ with $r \circ_h r'$ with $h = (h_l, h_K, h_R)$ for $h_L : IMP_L \rightarrow EXP'_L$, $h_K : IMP_K \rightarrow EXP'_K$, and $h_R : IMP_R \rightarrow EXP'_R$ be independ of the composition $C_1 \circ_{h_1} C'_1$, then we have

$C_1 \overset{r}{\Longrightarrow} C_3$ as well as $C_1' \overset{r'}{\Longrightarrow} C_3'$ and $C_1 \circ_{h_1} C_1' \overset{r \circ_h r'}{\Longrightarrow} C_3 \circ_{h_3} C_3'$
This can be illustrated in a diagram style by:

$$\begin{array}{ccccc} \widehat{C_1} & = & C_1 & \circ_{h_1} & C_1' \\ {\scriptstyle r \circ_h r'}\Downarrow & & \Downarrow{\scriptstyle r} & & \Downarrow{\scriptstyle r'} \\ \widehat{C_3} & = & C_3 & \circ_{h_3} & C_3' \end{array}$$

◇

For the proof see [37].

5 Instantiations with Automata and with Petri Nets

Automata Components The use of automata for the description of components and/or their interfaces is well established. In [10] the interfaces are modelled using input/output automata. The parallel composition of the interfaces is given and criteria for the compatibility are presented. But this approach concerns the interfaces only. In [1] input/output automata are used as well. There are three abstraction levels, the structural description of the architecture, the modeling of the component behavior and a data type description of the component. But the component is monolithic, consisting only of an interface.
Parameterized contracts [33, 38] are used for the adequate component composition and architecture evaluation in practice. Provides and requires interfaces a given using deterministic automata, and they are related – making parameterization of contracts possible – by so-called service effect automata (see [39]). These service effect automata represent the component specification and allow relating the interfaces. This component definition is the informal version of the definition of automata components below, where the provides interface corresponds to the export interface, the requires interface to the import interface and the service effect automata correspond to the body specification.
An automata component $AC = (IMP, EXP, BOD)$ consists of three deterministic automata, namely the import automaton IMP, the export automaton EXP and the body automaton BOD. Two morphisms $imp : IMP \to BOD$ and $exp : EXP \to BOD$ connect the interfaces to the body with $imp \in \mathcal{I}$ and $exp \in \mathcal{E}$. Automata component categories satisfy the adhesive HLR framework for generic components where $\mathcal{A}_{Aut}$ consists of the category of automata with plain morphisms, category of automata with refinement morphisms and $\mathcal{M}$ the class of plain, injective morphisms, see [37]. So, there are rules and transformation for automata components and we have the same results as in 4.5: Parallelism results, Concurrency and pair factorization and Embedding and local confluence.
And we have the Composition Theorem as in Theorem 4.9.

Petri net modules In the area of Petri nets various structuring concepts have been proposed during the last 40 years, some of these are even called modules or modular approach. There are hierarchical concepts (e.g. [29, 6, 28, 24]) as well as a variety of concepts for connector mechanisms as communication, coordination or cooperation (e.g. [8, 40, 12, 11]). In other approaches places and transitions of modules are merged by well-defined operations (e.g. [31, 3, 2, 5]). Most attempts to Petri net modules (among others [7, 13, 30]) do not provide Petri nets as interfaces. For a not really recent survey see [4]. There are either places or transitions, but no full Petri nets in the interface. When modeling software components these notions of Petri net modules are not powerful enough, since they do not allow specifying behavior in the interfaces. Object Coordination Nets [26, 27, 25] are somewhat more elaborate as they allow specifying simple protocols in their interfaces.
A Petri net (in the algebraic notion of [34]) is given by the set of transitions, the set of places and the pre- and post domain functions that map each transition to a multi-set of places. Plain morphisms are simple homomorphisms that are generated over the set of places. Substitution morphism are a generalization of plain morphisms, where transitions may be replaced by a subnet. They capture a very broad idea of refinement and hence are adequate for the relation between the export net and the body net. A Petri net module $MOD = (IMP, EXP, BOD)$ consists of three Petri nets, namely the import net IMP, the export net EXP and the body net BOD. Two Petri net morphisms $m : IMP \to BOD$ and $r : EXP \to BOD$ connect the interfaces to the body. Petri net modules [35] are an instantiation of the adhesive HLR framework for generic components where $\mathcal{A}_{PN}$ consists of the category of place/transition nets with plain morphisms, the category of place transition nets with substitution morphisms and the class of plain, injective morphisms, see [37]. So, there are rules and transformation for Petri net modules. For Petri net modules we have again the results 4.5: Parallelism results, Concurrency and pair factorization and Embedding and local confluence.
And we have the Composition Theorem as in Theorem 4.9.

6 Conclusion

In this paper we use the generic concept of components in a categorical frame, where a component consists of an import, an export and the body. High-level replacement systems are a categorical generalization of the algebraic approach to graph transformation systems with double pushouts. They allow formulating the same notions as for graph transformation systems, but not only for graphs but for objects of arbitrary categories.
The main result of this paper is the integration of both theories and a new re-

sult for compatibility of hierarchical composition and transformation. In order to achieve transformations of components we have to make the approach in [36] more concrete, by relating the morphism classes used for adhesive HLR systems and generic components leading to an adhesive HLR framework for generic components. Further instantiations of this approach can easily be obtained if the corresponding specification technique is shown to be a weak adhesive HLR category and if it is an instantiation of the generic component concept. These are various graph transformation systems, as the transfer of algebraic specification modules as defined by [18] to process description techniques has been started in [41], where modules for graph transformation systems and local action systems have been investigated. Algebraic specifications are an adhesive HLR category [23] and clearly conform with the generic framework for componets.

References

[1] M. C. Bastarrica, S. F. Ochoa, and P. O. Rossel. Integrated notation for software architecture specification. In *Proc. of the XXIV International Conference of the SCCC*, 2004.

[2] E. Battiston, F. De Cindio, and G. Mauri. OBJSA Nets: A Class of High-Level Nets Having Objects as Domains. In Rozenberg/Jensen, editor, *Advances in Petri Nets*. Springer, 1991.

[3] E. Battiston, F. De Cindio, G. Mauri, and L. Rapanotti. Morphisms and Minimal Models for OBJSA Nets. In *12^{t}h Int. Conference on Application and Theory of Petri Nets*, pages 455–476, 1991.

[4] L. Bernadinello and F. De Cindio. A survey of basic net models and modular net classes. *Advances in Petri Nets*, Lecture Notes in Computer Science 609, 1992.

[5] M. Broy and T. Streicher. Modular functional modelling of Petri nets with individual tokens. *Advances in Petri Nets*, Lecture Notes in Computer Science 609, 1992.

[6] P. Buchholz. Hierarchical high level Petri nets for complex system analysis. In *Application and Theory of Petri Nets*, Lecture Notes in Computer Science 815, pages 119–138. Springer, 1994.

[7] S. Christensen and L. Petrucci. Modular analysis of Petri nets. *Computer Journal*, 43(3):224–242, 2000.

[8] S. Christinsen and N.D. Hansen. Coloured Petri nets extended with channels for synchronous communication. In *Application and Theory of Petri Nets*, Lecture Notes in Computer Science 815, pages 159–178. Springer, 1994.

[9] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic approaches to graph transformation I : Basic Concepts and Double Pushout

Approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations*, chapter 3. World Scientific, 1997.

[10] L. de Alfaro and T.A Henzinger. Interface automata. In *ESEC/FSE 01: Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2001.

[11] W. Deiters and V. Gruhn. The FunSoft Net Approach to Software Process Management. *International Journal on Software Engineering and Knowledge Engineering*, 4(2):229–256, June 1994.

[12] J. Desel, G. Juhás, and R. Lorenz. Process semantics of Petri nets over partial algebra. In M. Nielsen and D. Simpson, editors, International Conference on Applications and Theory of Petri Nets, Lecture Notes in Computer Science 1825, pages 146–165. Springer, 2000.

[13] J. Desel, G. Juhás, and R. Lorenz. Petri Nets over Partial Algebras. In H. Ehrig, G. Juhás, J. Padberg, and G. Rozenberg, editors, *Advances in Petri Nets: Unifying Petri Nets*, Lecture Notes in Computer Science 2128. Springer, 2001.

[14] H. Ehrig, B. Braatz, M. Klein, F. Orejas, S. Pérez, and E. Pino. Object-oriented connector-component architectures. In *Proc. FESCA*, 2005.

[15] H. Ehrig, M. Gajewsky, and F. Parisi-Presicce. High-Level Replacement Systems with Applications to Algebraic Specifications and Petri Nets. In G. Rozenberg, U. Montanari, H. Ehrig, and H.-J. Kreowski, editors, *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 3: Concurrency, Parallelism, and Distribution*, chapter 6, pages 341–400. World Scientific, 1999.

[16] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Math. Struct. in Comp. Science*, 1:361–404, 1991.

[17] H. Ehrig, A. Habel, J. Padberg, and U. Prange. Adhesive high-level replacement categories and systems. In F. Parisi-Presicce, P. Bottoni, and G. Engels, editors, *Proc. ICGT'04*, Lecture Notes in Computer Science 3256, pages 144–160, 2004. Springer.

[18] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, Berlin, 1990.

[19] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A Generic Component Concept for System Modeling. In *Proc. FASE 2002: Formal Aspects of Software Engineering*, Lecture Notes in Computer Science 2306, pages 32–48. Springer, 2002.

[20] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A Transformation-Based Component Framework for a Generic Integrated Modeling Technique. *Journal of Integrated Design and Process Science*, 6(4):78–104, June 2003.

[21] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A component framework for system modeling based on high-level replacement systems. *Software and Systems Modeling*, pages 114–134, 3 2004.

[22] Hartmut Ehrig, Julia Padberg, Benjamin Braatz, Markus Klein, Fernando Orejas, Sonia Pérez, and Elvira Pino. A generic framework for connector architectures based on components and transformations. In *Proc. FESCA'04, satellite of ETAPS'04, Barcelona, ENTCS*, volume 108, pages 53–67, 2004.

[23] K. Ehrig, H.and Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Springer, 2005. in preparation.

[24] R. Fehling. A concept of hierarchical Petri nets with building blocks. In *Advances in Petri Nets'93*, Lecture Notes in Computer Science 674, pages 148–168. Springer, 1993.

[25] H. Giese. Object Coordination Nets 3.0: Reference Guide. Technical Report 1/01-I, University Münster, Computer Science, Distributed Systems Group, 2001.

[26] H. Giese, J. Graf, and G. Wirtz. Closing the Gap Between Object-Oriented Modeling of Structure and Behaviour. In R. France and B. Rumpe, editors, *UML'99 - The Second Int. Conference on The Unified Modeling Language*, Lecture Notes in Computer Science 1723, pages 534 – 549. Springer, 1999.

[27] H. Giese and G. Wirtz. The OCoN Approach for Object-Oriented Distributed Software Systems Modeling. *Computer Systems Science & Engineering*, 16(3):157–172, 2001.

[28] X. He. A Formal Definition of Hierarchical Predicate Transition Nets. In *Application and Theory of Petri Nets*, Lecture Notes in Computer Science 1091, pages 212–229. Springer, 1996.

[29] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1: Basic Concepts. Springer Verlag, EATCS Monographs in Theoretical Computer Science edition, 1992.

[30] G. Juhás and R. Lorenz. Modelling with Petri modules. In B. Caillaud, X. Xie, and L. Darondeau, Ph.and Lavagno, editors, *Synthesis and Control of Discrete Event Systems*, pages 125–138. Kluwer Academic Publishers, 2002.

[31] E. Kindler. *Modularer Entwurf verteilter Systeme mit Petrinetzen*. PhD thesis, Technische Universität München, Institut für Informatik, 1995.

[32] S. Lack and P. Sobociński. Adhesive Categories. In *Proc. FOSSACS 2004*, Lecture Notes in Computer Science 2987, pages 273–288. Springer, 2004.

[33] J. Matevska-Meyer, W. Hasselbring, and R. Reussner. Software architecture description supporting component deployment and system runtime reconfiguration. In *Proceedings of Workshop on Component-Oriented Programming (WCOP 2004)*, 2004.

[34] J. Meseguer and U. Montanari. Petri Nets are Monoids. *Information and Computation*, 88(2):105–155, 1990.

[35] J. Padberg. Petri net modules. *Journal on Integrated Design and Process Technology*, 6(4):121–137, 2002.

[36] J. Padberg and H. Ehrig. Petri net modules in the transformation-based component framework. *Journal of Logic and Algebraic Programming*, page 35, 2005. accepted.

[37] Julia Padberg. High-level replacement systems for software components. Technical Report 2005-07, Technische Universität Berlin, 2005.

[38] R. Reussner and H. W. Schmidt. Using Parameterised Contracts to Predict Properties of Component-Based Software Architectures. In I. Crnkovic, S. Larsson, and J. Stafford, editors, *Workshop on Component-Based Software Engineering*, 2002.

[39] R. Reussner. *Parametrisierte Verträge zur Protokolladaption bei Software-Komponenten*. PhD thesis, Universität Karlsruhe (Technische Hochschule), 2001.

[40] C. Sibertin-Blanc. Cooperative Nets. In *Application and Theory of Petri Nets'94*, pages 471–490. Springer Lecture Notes in Computer Science 815, 1994.

[41] M. Simeoni. *A Categorical Approach to Modularization of Graph Transformation Systems using Refinements*. PhD thesis, Università Roma *La Sapienza*, 1999.

The Logic in Computer Science Column

by

Yuri Gurevich

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

First-order topological properties

Jan Van den Bussche
Universiteit Hasselt

Author: Oh, hi, you're Quisani, Yuri Gurevich's student, right?

Quisani: That's me allright; can I help you?

A: Well, yes, Yuri asked me if I wanted to write a logic column, and suggested to talk to you for some inspiration.

Q: Sure, we can talk. Do you already have an idea what you will write about?

A: I was thinking, perhaps I could write about first-order topological properties.

Q: I suppose that is first-order as in "first-order logic", but what's about the topology? I'm afraid I do not know more about topology than what you find about it in a dictionary. It's supposed to deal with the shapes of objects; things that do not change when you deform an object, right?

A: That's quite right. Topology is basically that part of geometry that deals with properties that remain invariant under continuous transformations.

Q: But what has that to do with logic?

A: In itself not much. But it is a legitimate question to ask which properties of geometrical objects are at the same time topological and expressible in first-order logic.

Q: Why would anyone care about that?

A: Well, the original motivation comes from the theory of spatial databases.

Q: Do you mean databases the NASA folks use?

A: They probably do use spatial databases at NASA, but the general term does not specifically refer to space in that sense. A spatial database is simply any database that contains data with a geometrical interpretation [24]. Typical application areas are Geographical Information Systems (GIS) and robotics [29, 8].

Q: I see. I suppose topology could indeed be relevant in such applications.

A: You bet. For certain applications, only the topology of the data is important, and not the more specific geometrical aspects like precise locations or rotations. Applications concerned with dimensionality, or with connectivity, fall in this category.

Q: Could you be a little bit more concrete?

A: Sure, take the Paris metro map for example. It gives you full information about the topology of the subway network: how stations are connected. But it is very unreliable in other aspects, such as precise distances.

Q: And people are fine with that, because anyway they use the map only to check how they can get from point A to point B, and are happy with using the number of intermediate stations on the route as a purely topological approximation of distance. OK, I'm with you. But where does the first-order logic come in the picture?

A: We're getting there. Let's first agree that we are interested in the topology of some geometrical object, which we formalize in the standard mathematical way as a set A of points in n-dimensional real space $\mathbb{R}^n$.

A: By using Cartesian coordinates, we can view A as an n-ary relation on $\mathbb{R}$.

Q: Excellent remark, because that's exactly what we will do. This allows us to use first-order logic sentences φ over the vocabulary $(<, 0, 1, +, \times, S)$, evaluated over $\mathbb{R}$, to express properties about geometric objects. Here, $<$, 0, 1, $+$ and $\times$ are the obvious predicates and functions over $\mathbb{R}$, and S is an n-ary relation symbol interpreted by the set A we want to talk about.

Q: OK, let's see... if I want to express that $S \subseteq \mathbb{R}^2$ is a straight line, I can write

$$\exists a \exists b \forall x \forall y (S\,xy \leftrightarrow a \times x + b = y).$$

A: Very good. It is not a topological property, but it is a first-order expressible property allright. To give you an example of a first-order property that is topological, consider the property that $S \subseteq \mathbb{R}^2$ contains a two-dimensional subset. For that we can write

$$\exists x_0 \exists y_0\, \exists r > 0\, \forall x \forall y ((x - x_0)^2 + (y - y_0)^2 < r \rightarrow S\,xy).$$

Q: OK. Now let me try to express that S is topologically connected. For $n = 1$ that is easy, because a subset of the real line is connected if and only if it is an open, half-open, or closed interval, and we can easily express that in first-order logic. But I do not immediately see how to do it in $\mathbb{R}^2$.

A: You will never see it, because it is impossible to do. This was shown by the combined results of Grumbach and Su [15] and Benedikt, Dong, Libkin and Wong [3].

Q: But doesn't that kill your whole story? I thought connectivity questions provided the motivation to study this stuff in the first place.

A: I didn't say that. I said that connectivity is a typical example of a topological question, but there are many others. First-order logic is the theoretical foundation for all database query languages [1], including spatial database query languages, and therefore it is important that we understand precisely which topological properties are expressible in first-order logic, even if we already know connectivity is not one of them.

Q: Fair enough. I see you are eager to tell me what those expressible properties are, but please allow me one further question first. Now that you tell me you are considering first-order logic to be a query language, I wonder how can one evaluate first-order logic sentences over arbitrary, typically infinite subsets of $\mathbb{R}^n$ in an effective manner? Clearly, we want our query language to be implementable on a computer, don't we?.

A: Excellent point; I have almost forgotten to tell you about that. You know about the decidability of the first-order theory of the reals?

Q: Sure, this is an old theorem of Tarski [27]; he showed that the truth of any first-order sentence about the reals can be effectively determined. In the setting we are talking about, these would be the sentences without the extra predicate S. The example they always give of a true sentence over the reals is the solution of quadratic equations:

$$\forall a \forall b \forall c \exists x (ax^2 + bx + c = 0 \leftrightarrow b^2 - 4ac \geq 0)$$

A: Very good. In view of this decidability, we will restrict attention to sets A that are first-order definable over $\mathbb{R}$. That allows us to effectively evaluate a query φ on a set A, simply by plugging in the formula that defines A at all places in φ where the predicate symbol S is used.

Q: But are the sets in $\mathbb{R}^n$ that are explicitly definable by a first-order logic formula over $\mathbb{R}$ interesting enough in practice?

A: They sure are. They include all the geometric objects one encounters in elementary geometry, and form a well-studied class of sets known as the "semi-algebraic sets". They are sufficient for robotics applications [25] and are surely

enough for GIS and computer graphics applications, where all the data objects are typically built up from straight line segments.

Q: OK, OK, I see; descriptions of geometrical objects in Cartesian coordinates using polynomial equations or inequalities, such as lines, arcs, circles, cubes, ellipsoids, cylinders, and so on, all fall in the realm of first-order logic formulas. And then we can also take projections (through existential quantification), unions, intersections, complements.

But, isn't going through the first-order theory of the reals an enormously inefficient way of implementing your query language?

A: Of course, specific operations on specific geometric objects can be implemented much more efficiently using specific algorithms from computational geometry. It is the typical trade-off between specificity and generality. Note, however, that algorithmic progress on decision procedures for the reals has been ongoing ever since Collins's cylindrical algebraic decomposition method [2, 7, 16]. And if nothing else, this whole idea of "plug-in evaluation" remains a nice theoretical framework. It was first proposed by Kanellakis, Kuper and Revesz in the form of "constraint databases" [17, 21]. Peter Revesz and his students have implemented quite a few constraint database systems, for different logical theories.

Q: Very interesting. But perhaps we should move on.

A: Yes, let's move on. What I wanted to show you is a characterisation of the first-order topological properties of closed semi-algebraic sets in the plane.

Q: So we're focusing here on subsets of the plane, $\mathbb{R}^2$, but what does "closed" mean?

A: It's a standard topological concept. It means that the set includes all its border. For example, the set of points inside the unit circle, $x^2 + y^2 < 1$, is not closed, but the points inside together with the circle itself, $x^2 + y^2 \leq 1$, is closed. Actually, you can express that S is closed in first-order:

$$\forall x \forall y (\mathrm{Acc}(x, y) \rightarrow S\, xy),$$

where $\mathrm{Acc}(x, y)$ stands for the formula

$$\exists \varepsilon > 0\, \forall \delta \in\,]0, \varepsilon[\, \exists x_0 \exists y_0 (S\, x_0 y_0 \wedge 0 < (x - x_0)^2 + (y - y_0)^2 < \delta)$$

Q: Got it; $\mathrm{Acc}(x, y)$ expresses that (x, y) is an accumulation point. Anyway, from a practical standpoint, restricting to closed sets does not seem too harmful. I wouldn't know how to draw a set on a piece of paper without drawing its borders as well!

A: You could use different colors, e.g., draw the borders that are not part of the set in red and the rest in green. But for a drawing in one color you're right.

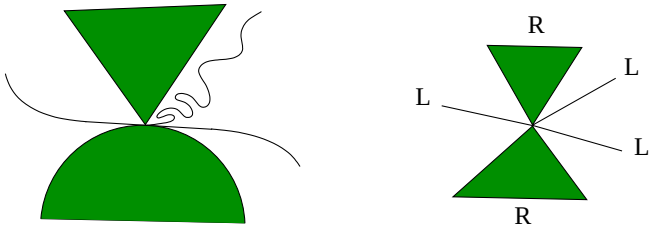


Figure 1: A set and the cone of one of its points.

Now locally around each of its points, a closed semi-algebraic set in the plane looks "conical", in the sense that if you travel around the point in a small enough radius, you will always encounter the same circular list of lines and regions. We call that circular list of L's and R's the "cone" of the point. Let me illustrate it with a drawing (Figure 1).

Q: I see. Is this so because the set is closed?

A: Not really, it is rather because the set is semi-algebraic. Semi-algebraicity rules out "wild" topologies [28]. If the set is not closed, the cones may be a bit more complicated than the ones we have here.

Q: Let me look at some special cases to make sure I understand your definition of cone. If I have a point lying one a line, then its cone is (*LL*), because we see the line to the left and the right of the point. For an endpoint of a line, the cone is simply (*L*). And for a point in the interior of the geometrical object, the cone is (*R*), right?

A: You're right about the points on a line, and about the endpoints, but a point with cone (*R*) is a point on the border of a region. Actually, an interior point is completely surrounded by a region, something we do not have a notation for yet. So let's introduce one and indicate the cone of an interior point by the special letter *F* (for "full").

Q: OK. So in your drawing (Figure 1), all the different cones that we can see, apart from (*RLLRL*) for the central point, are three times (*L*) for the three endpoints of lines; and infinitely many (*LL*)'s, (*R*)'s, and *F*'s, for all the points on the lines, on the borders of the regions, and inside the regions, respectively.

Now that I think about it: the cones (*LL*), (*R*) and *F* always occur infinitely often if they occur at all. Moreover, cones *F* occur if and only if at least one of the cones has an *R*. So, *F* is in a sense redundant.

A: Good observation, unless the set consists of the entire plane, but let's forget about that special case. Now I can present you with a first theorem [20]: two sets in which precisely the same cones appear, with precisely the same multiplicities,

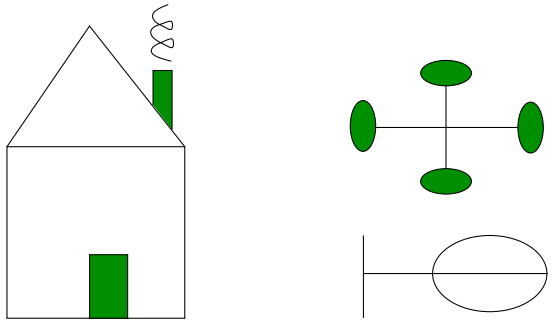


Figure 2: The set on the left has precisely the same cones, with precisely the same multiplicities, as the set on the right. Hence the two sets are indistinguishable by topological first-order sentences.

are indistinguishable by topological first-order sentences. Here's an illustration (Figure 2).

Q: Awesome! How can such a thing be proved?

A: You should look at the paper [20], but in brief, a number of elementary transformations are introduced by which two sets with the same cones can be transformed into each other. These transformations are shown to be indistinguishable by topological first-order sentences. The proof technique involves a reduction to finite structures [15], and an fundamental tool is provided by the collapse theorems for embedded finite models [3, 5, 22].

Q: I'll add the chapter on embedded finite models of Libkin's book to my bedtime reading list.

A: Sweet dreams! Anyway, this indistinguishability theorem really paves the road for a full characterisation of the first-order topological properties.

Q: Hate to stop you now, but one quick remark: I suppose the indistinguishability theorem is actually an if and only if, right? I mean, it seems a doable exercise to express in first-order that a point has a given cone, so if two sets do not agree on their cones then they are distinguishable by a topological first-order sentence.

A: You got it. Now our second theorem lifts the indistinguishability theorem to the global level of properties, and says that the first-order topological properties are precisely the properties that can be expressed in first-order logic when looking only at the cones.

Q: To make sense of that statement we need some kind of logical interpretation of cones.

A: Precisely. But that is very natural. Recall that a cone is a circular list of L's and R's. We can represent such a list as a finite structure $\{1, 2, \ldots, n\}$, where n is the length of the list, equipped with the following relations: L and R are unary relations containing the positions that are L and R, respectively; B is the ternary relation consisting of the triples (i, j, k) of distinct positions such that j comes before k in the sequence

$$i, i+1, \ldots, n, 1, 2, \ldots, i-1.$$

Q: So B is a "circular" order relation; I assume 'B' stands for 'between'?

A: You got it. We can now use first-order logic sentences over the vocabulary (L, R, B) to express properties of cones.

Q: Let me try. To express that there are two consecutive L's in the cone, I could write

$$\exists x \exists y \neg \exists z (Bxzy \wedge Lx \wedge Ly \wedge x \neq y)$$

A: Correct! We are now ready to define "cone logic sentences". These are simply boolean combinations of basic sentences of the form $|\gamma| \geq k$, where γ is a first-order sentence over (L, R, B) as above, and k is a natural number. The meaning of such a basic sentence is simply that there are at least k points in the set whose cone satisfies γ.

Q: I see. So, to express that the set is a bunch of non-intersecting lines and loops, i.e., that the only cones that can occur are (L) and (LL), we could write

$$\neg(|\gamma| \geq 1)$$

where γ is

$$\exists x \exists y \exists z (x \neq y \wedge x \neq z \wedge y \neq z) \vee \exists x\, Rx$$

A: The formal theorem now is that the first-order topological properties of closed semi-algebraic sets in the real plane are precisely those that can be expressed as cone logic sentences [6]. Not surprisingly, it is undecidable whether a given first-order sentence is topological. So, the theorem gives us an alternative, syntactic characterisation of an undecidable class of sentences.

Q: I can sympathize with your enthusiasm. It is a beautiful result. How is it proved?

A: You should again look in the paper (a full version is available from me), but in brief, sets are put in a normal form consisting of drawings of cones. Sets in this normal form can be represented by abstract finite structures, which we call "codes". We then show how to rewrite a topological first-order sentence φ into a first-order sentence ψ about codes. The major technical hurdle left after that is

that the codes contain information about how cones are linked together by lines. Because we know that this linking information is not captured by φ, we can remove it also from ψ, but this requires a few complicated invariance arguments. The resulting ψ sans linking information then easily yields a cone logic sentence.

Q: Sounds like a nice achievement; congratulations!

A: It was certainly not my achievement alone; a lot of credit goes to my collaborators Michael Benedikt, Bart Kuijpers, Christof Löding, and Thomas Wilke.

Q: So, what does the future hold?

A: A lot of remaining open questions. First of all, we now have a characterisation of the first-order topological properties, but this is for closed semi-algebraic sets in the plane only. What about non-closed sets? A semi-algebraic set can always be written as a boolean combination of closed semi-algebraic sets. Hence, we can ask more generally, what about properties not of a single set, but of a collection of sets, even closed sets? Grohe and Segoufin [14] have shown that already the indistinguishability theorem fails for non-closed sets, even within a class of very simple sets. For that class they do give a new indistinguishability theorem, however.

Further, what in higher dimensions? And what about properties of more general, not necessarily semi-algebraic, sets? On the one hand, the problem becomes more difficult, because we lose the tame topology of semi-algebraic sets. On the other hand, the problem becomes easier, because less first-order sentences will be topological now.

Q: That's quite a research program you have there!

A: I must admit, though, that only a handful of people have worked on this topic. Apart from the people whose papers I've already cited, there are Papadimitriou, Suciu and Vianu [23] who investigated logics over the plane graph representation of the topology of the set, and Segoufin and Vianu [26], who continued that line of research with some very interesting results. As far as I know, currently none of these people is very much occupied with trying to continue the topic. We need new blood!

Q: We're not going to end our conversation on a pessimistic note, are we?

A: No, we can't do that, can we! At any rate, it is quite clear already now that first-order logic is much too weak to be a useful topological query language. So, rather than invest effort in understanding better exactly how weak it is, that effort is perhaps better spent on the investigation of extensions of the language. There has already been interesting work in that direction [4, 13, 10, 18, 19, 11, 12]. Interestingly, a good understanding of extensions of the first-order language can necessitate further work on the first-order language itself [9].

Q: Whew. I have plenty to read and think about now.

A: You've been a great help to me in the writing of this column.

Q: You're welcome.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2003.

[3] M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational expressive power of constraint query languages. *Journal of the ACM*, 45(1):1–34, 1998.

[4] M. Benedikt, M. Grohe, L. Libkin, and L. Segoufin. Reachability and connectivity queries in constraint databases. *Journal of Computer and System Sciences*, 66(1):169–206, 2003.

[5] M. Benedikt and L. Libkin. Relational queries over interpreted structures. *Journal of the ACM*, 47(4):644–680, 2000.

[6] M. Benedikt, C. Löding, J. Van den Bussche, and T. Wilke. A characterization of first-order topological properties of planar spatial data (extended abstract). In *Proceedings 23th ACM Symposium on Principles of Database Systems*, pages 107–114. ACM Press, 2004.

[7] B.F. Caviness and J.R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer, 1998.

[8] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer, 2000.

[9] F. Geerts. Expressing the box cone radius in the relational calculus with real polynomial constraints. *Discrete and Computational Geometry*, 30(4):607–622, 2003.

[10] F. Geerts and B. Kuijpers. Expressing topological connectivity of spatial databases. In R.C.H. Connor and A.O. Mendelzon, editors, *Research Issues in Structured and Semistructured Database Programming*, volume 1949 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2000.

[11] F. Geerts, B. Kuijpers, and J. Van den Bussche. Linearization and completeness results for terminating transitive closure queries on spatial databases. *SIAM Journal on Computing*. To appear.

[12] F. Geerts, L. Smits, and J. Van den Bussche. N- versus (N+1)-dimensional connectivity testing of first-order queries to semi-algebraic sets. *Acta Informatica*. To appear.

[13] C. Giannella and D. Van Gucht. Adding a path connectedness operator to FO+poly. *Acta Informatica*, 38(9):621–648, 2002.

[14] M. Grohe and L. Segoufin. On first-order topological queries. *ACM Transactions on Computational Logic*, 3(3):336–358, 2002.

[15] S. Grumbach and J. Su. Queries with arithmetical constraints. *Theoretical Computer Science*, 173(1):151–181, 1997.

[16] J. Heintz and B. Kuijpers. Constraint databases, data structures, and efficient query evaluation. In B. Kuijpers and P. Revesz, editors, *Constraint Databases—Proceedings CDB 2004*, volume 3074 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2004.

[17] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, August 1995.

[18] S. Kreutzer. Fixed-point query languages for linear constraint databases. In *Proceedings 19th ACM Symposium on Principles of Database Systems*, pages 116–125. ACM Press, 2000.

[19] S. Kreutzer. Operational semantics for fixed-point logics on constraint databases. In R. Nieuwenhuis and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning—Proceedings LPAR 2001*, volume 2250 of *Lecture Notes in Computer Science*, pages 470–484, 2001.

[20] B. Kuijpers, J. Paredaens, and J. Van den Bussche. On topological elementary equivalence of closed semi-algebraic sets in the plane. *Journal of Symbolic Logic*, 65(4):1530–1555, 2000.

[21] G. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases*. Springer, 2000.

[22] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[23] C.H. Papadimitriou, D. Suciu, and V. Vianu. Topological queries in spatial databases. *Journal of Computer and System Sciences*, 58(1):29–53, 1999.

[24] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases*. Morgan Kaufmann, 2001.

[25] J.T. Schwartz, M. Sharir, and J. Hopcroft, editors. *Planning, Geometry, and Complexity of Robot Motion*. Ablex Publishing Corporation, Norwood, New Jersey, 1987.

[26] L. Segoufin and V. Vianu. Querying spatial databases via topological invariants. *Journal of Computer and System Sciences*, 61(2):270–301, 2000.

[27] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[28] L. van den Dries. *Tame Topology and O-Minimal Structures*. Cambridge University Press, 1998.

[29] M. Worboys and M. Duckham. *GIS: A Computing Perspective*. CRC Press, 2004.

The Natural Computing Column

by

Grzegorz Rozenberg

Leiden University, Leiden Center for Natural Computing
Niels Bohrweig 1, 2333 CA Leiden, The Netherlands
rozenber@liacs.nl

Development of a Bacteria Computer: From *in silico* finite automata to *in vitro* and *in vivo*

Yasubumi Sakakibara
Keio University, Department of Biosciences and Informatics
yasu@bio.keio.ac.jp

Abstract

We overview a series of our research on implementing finite automata *in vitro* and *in vivo* in the framework of DNA-based computing [4, 5]. First, we employ the length-encoding technique proposed and presented in [8, 7] to implement finite automata in test tube. In the length-encoding method, the states and state transition functions of a target finite automaton are effectively encoded into DNA sequences, a computation (accepting) process of finite automata is accomplished by self-assembly of encoded complementary DNA strands, and the acceptance of an input string is determined by the detection of a completely hybridized double-strand DNA. Second, We report our intensive *in vitro* experiments in which we have implemented and executed several finite-state automata in test tube. We have designed and developed practical laboratory protocols which combine several *in vitro* operations such as annealing, ligation, PCR, and streptavidin-biotin bonding to execute *in vitro* finite automata based on the length-encoding technique. We have carried laboratory experiments on various finite automata of from 2 states to 6 states for several input strings. Third, we present a novel framework to develop a programmable and autonomous *in vivo* computer using

Escherichia coli (E. coli), and implement *in vivo* finite-state automata based on the framework by employing the protein-synthesis mechanism of *E. coli*. Our fundamental idea to develop a programmable and autonomous finite-state automata on *E. coli* is that we first encode an input string into one plasmid, encode state-transition functions into the other plasmid, and introduce those two plasmids into an *E. coli* cell by electroporation. Fourth, we execute a protein-synthesis process in *E. coli* combined with four-base codon techniques to simulate a computation (accepting) process of finite automata, which has been proposed for *in vitro* translation-based computations in [7]. This approach enables us to develop a programmable *in vivo* computer by simply replacing a plasmid encoding a state-transition function with others. Further, our *in vivo* finite automata are autonomous because the protein-synthesis process is autonomously executed in the living *E. coli* cell. We show some successful experiments to run an *in vivo* finite-state automaton on *E. coli*.

1 Introduction

Biological molecules such as DNA, RNA and proteins are natural devices to store information, activate (chemical) functions and communicate between systems (such as cells). DNA computer study utilizes these biological devices to make a computer. One of the ultimate goals is to make an autonomous cell-based turing machine and apply to genetic and life engineering. Our attempts to make a bacteria-based computer make progress toward this goal.

The finite-state automata (machines) are the most basic computational model in Chomsky hierarchy and are the start point to build universal DNA computers. Several works have attempted to develop finite automata *in vitro*. However, there have been no experimental research works which attempt to build a finite automaton *in vivo*. Benenson et al. [1] have successfully implemented the two state finite automata by the sophisticated use of the restriction enzyme (actually, *Fok*I) which cut outside of its recognition site in a double-stranded DNA. However, their method has some limitations for extending to more than 2 states. Yokomori et al. [8] have proposed a theoretical framework using length-encoding technique to implement finite automata on DNA molecules. Theoretically, the length-encoding technique has no limitations to implement finite automata of any larger states.

In our first research work [4], we have attempted to implement and execute finite automata of a larger number of states *in vitro*, and carry intensive laboratory experiments on various finite automata of from 2 states to 6 states for several input strings.

On the other hand, in our next research work [5], we have previously proposed

a method using the protein-synthesis mechanism combined with four-base codon techniques to simulate a computation (accepting) process of finite automata *in vitro* [7] (a codon is normally a triplet of base, and different base triplets encode different amino acids in protein). The proposed method is quite promising and has several advanced features such as the protein-synthesis process is very accurate and overcomes mis-hybridization problem in the self-assembly computation and further offers an autonomous computation. Our aim was to extend this novel principle into a living system, by employing the *in vivo* protein-synthesis mechanism of *Escherichia coli (E. coli)*. (Escherichia coli is a typical bacteria living inside our body, large intestine.) This *in vivo* computation possesses the following two novel features, not found in any previous biomolecular computer. First, an *in vivo* finite automaton is implemented in a living *E. coli* cell; it does not mean that it is executed simply by an incubation at a certain temperature. Second, this automaton increases in number very rapidly according to the bacterial growth; one bacterial cell can multiply to over a million cells overnight. The present study explores the feasibility of *in vivo* computation.

The main feature of our *in vivo* computer based on *E. coli* is that we first encode an input string into one plasmid, encode state-transition functions into the other plasmid, and transform *E. coli* cells with these two plasmids by electroporation. Second, we execute a protein-synthesis process in *E. coli* combined with four-base codon techniques to simulate a computation (accepting) process of finite automata, which has been proposed for *in vitro* translation-based computations in [7]. The successful computations are detected by observing the expressions of a reporter gene linked to mRNA encoding an input data. Therefore, when an encoded finite automaton accepts an encoded input string, the reporter gene, *lacZ*, is expressed and hence we observe a blue color. When the automaton rejects the input string, the reporter gene is not expressed and hence we observe no blue color. Our *in vivo* computer system based on *E. coli* is illustrated in Fig. 1.

Thus, our *E. coli*-based computer enables us to develop a programmable and autonomous computer. To our knowledge, this is the first experimental development of *in vivo* computer and has succeeded to execute an finite-state automaton on *E. coli*.

2 Methods

2.1 Length-encoding method to implement finite-state automata

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a (deterministic) finite automaton, where Q is a finite set of states numbered from 0 to k, Σ is an alphabet of input symbols, δ is a state-

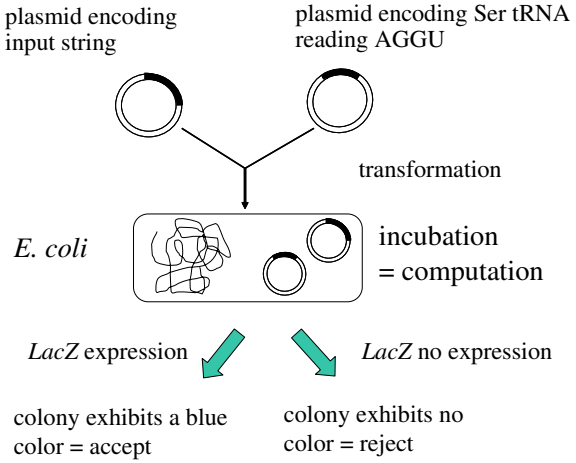


Figure 1: The framework of our *in vivo* computer system based on *E. coli*.

transition function such that $\delta : Q \times \Sigma \longrightarrow Q$, q_0 is the initial state, and F is a set of final states. We adopt the length-encoding technique [8] to encode each state in Q by the length of DNA subsequences.

For the alphabet Σ, we encode each symbol a in Σ into a single-strand DNA subsequence, denoted $e(a)$, of fixed length. For an input string w on Σ, we encode $w = x_1 x_2 \cdots x_m$ into the following single-strand DNA subsequence, denoted $e(w)$:

$$5'\text{-}\ e(x_1)\underbrace{\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_k}_{k\text{ times}}e(x_2)\underbrace{\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_k}_{k\text{ times}}\cdots e(x_m)\underbrace{\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_k}_{k\text{ times}}\ \text{-}3',$$

where $\mathtt{X}_i$ is one of four nucleotides $\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}$, and the subsequences $\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_k$ are used to encode $k+1$ states of the finite automaton M. For example, when we encode a symbol '1' into a ssDNA subsequence GCGC and a symbol '0' into GGCC, and encode three states into TT, a string "1101" is encoded into the following ssDNA sequence:

$$5'\text{-}\quad \overbrace{\mathtt{GCGC}}^{1}\,\mathtt{TT}\,\overbrace{\mathtt{GCGC}}^{1}\,\mathtt{TT}\,\overbrace{\mathtt{GGCC}}^{0}\,\mathtt{TT}\,\overbrace{\mathtt{GCGC}}^{1}\,\mathtt{TT}\quad \text{-}3'$$

In addition, we append two supplementary subsequences at both ends for PCR and probes for affinity purifications with magnetic beads which will be used in laboratory protocol:

$$5'\text{-}\ \underbrace{\mathtt{S}_1\mathtt{S}_2\cdots\mathtt{S}_s}_{\text{PCR primer}}e(x_1)\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_k\cdots e(x_m)\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_k\underbrace{\mathtt{Y}_1\mathtt{Y}_2\cdots\mathtt{Y}_t}_{\text{probe}}\underbrace{\mathtt{R}_1\mathtt{R}_2\cdots\mathtt{R}_u}_{\text{PCR primer}}\ \text{-}3'.$$

For a state-transition function from state q_i to state q_j with input symbol $a \in \Sigma$, we encode the state-transition function $\delta(q_i, a) = q_j$ into the following complementary single-strand DNA subsequence:

$$3'\text{-}\ \underbrace{\overline{\mathtt{X}}_{i+1}\overline{\mathtt{X}}_{i+2}\cdots\overline{\mathtt{X}}_{k}}_{k-i \text{ times}}\,\overline{e(a)}\,\underbrace{\overline{\mathtt{X}}_{1}\overline{\mathtt{X}}_{2}\cdots\overline{\mathtt{X}}_{j}}_{j \text{ times}}\ \text{-}5'$$

where $\overline{\mathtt{X}}_i$ denotes the complementary nucleotide of $\mathtt{X}_i$, and $\bar{y}$ denotes the complementary sequence of y. Further, we put two more complementary ssDNA sequences for the supplementary subsequences at both ends:

$$3'\text{-}\ \overline{\mathtt{S}}_1\overline{\mathtt{S}}_2\cdots\overline{\mathtt{S}}_s\ \text{-}5', \qquad 3'\text{-}\ \underbrace{\overline{\mathtt{Y}}_1\overline{\mathtt{Y}}_2\cdots\overline{\mathtt{Y}}_t\overline{\mathtt{R}}_1\overline{\mathtt{R}}_2\cdots\overline{\mathtt{R}}_u}_{\text{biotinylated}}\ \text{-}5',$$

where the second ssDNA is biotinylated for streptavidin-biotin bonding.
Now, we put all those ssDNAs encoding an input string w and encoding state transition functions and the supplementary subsequences of probes and PCR primers. Then, a computation (accepting) process of the finite automata M is accomplished by self-assembly among those complementary ssDNAs, and the acceptance of an input string w is determined by the detection of a completely hybridized double-strand DNA.

The main idea of length-encoding technique is explained as follows. Two consecutive valid transitions $\delta(h, a_n) = i$ and $\delta(i, a_{n+1}) = j$ are implemented by concatenating two corresponding encoded ssDNAs, that is,

$$3'\text{-}\ \underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{k-h}\,\overline{e(a_n)}\,\underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{i}\ \text{-}5',$$

and

$$3'\text{-}\ \underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{k-i}\,\overline{e(a_{n+1})}\,\underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{j}\ \text{-}5'$$

together make

$$3'\text{-}\ \underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{k-h}\,\overline{e(a_n)}\,\underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{k}\,\overline{e(a_{n+1})}\,\underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{j}\ \text{-}5'.$$

Thus, the subsequence $\underbrace{\mathtt{AAA}\cdots\mathtt{A}}_{k}$ plays a role of "joint" between two consecutive state-transitions and it guarantees for the two transitions to be valid in M.

2.2 Designing laboratory protocols to execute finite automata in test tube

In order to practically execute the laboratory experiments for the method described in the previous section, we design the following experimental laboratory protocol, which is also illustrated in Fig. 2:

0. Encoding: Encode an input string into a long ssDNA, and state transition functions and supplementary sequences into short pieces of complementary ssDNAs.

1. Hybridization: Put all those encoded ssDNAs together into one test tube, and anneal those complementary ssDNAs to be hybridized.

2. Ligation: Put DNA "ligase" into the test tube and invoke ligations at temperature of 37 degree. When two ssDNAs encoding two consecutive valid state-transitions $\delta(h, a_n) = i$ and $\delta(i, a_{n+1}) = j$ are hybridized at adjacent positions on the ssDNA of the input string, these two ssDNAs are ligased and concatenated.

3. Denature and extraction by affinity purification: Denature double-stranded DNAs to ssDNAs and extract concatenated ssDNAs containing biotinylated probe subsequence by streptavidin-biotin bonding with magnetic beads.

4. Amplification by PCR: Amplify the extracted ssDNAs with PCR primers.

5. Detection by gel-electrophoresis: Separate the PCR products by length using gel-electrophoresis and detect a particular band of the full-length. If the full-length band is detected, that means a completely hybridized double-strand DNA is formed, and hence the finite automaton "accepts" the input string. Otherwise, it "rejects" the input string. In our laboratory experiments, we have used a "capillary" electrophoresis microchip-based system, called Bioanalyser 2100 (Agilent Technologies), in place of conventional gel-electrophoresis. The capillary electrophoresis is of higher resolution and more accurate than gel electrophoresis such as agarose gel.

3 In vitro experiments

We have carried laboratory *in vitro* experiments on various finite automata for several input strings.

3.1 4-states automaton with three input strings

Our first experiment attempts 4-states automaton shown in Fig. 3 (upper left) for the three input strings (a) 1101, (b) 1110, and (c) 1010. This 4-states automaton accepts the language $(1(0 \cup 1)1)^* \cup (1(0 \cup 1)1)^*0$, and hence it accepts 1110 and 1010 and rejects 1101.

The results are shown in Fig. 3 (upper right) in gel-like image. As in the first experiment, the full-length DNA is of 190 bps (mer). Bands at position of 190

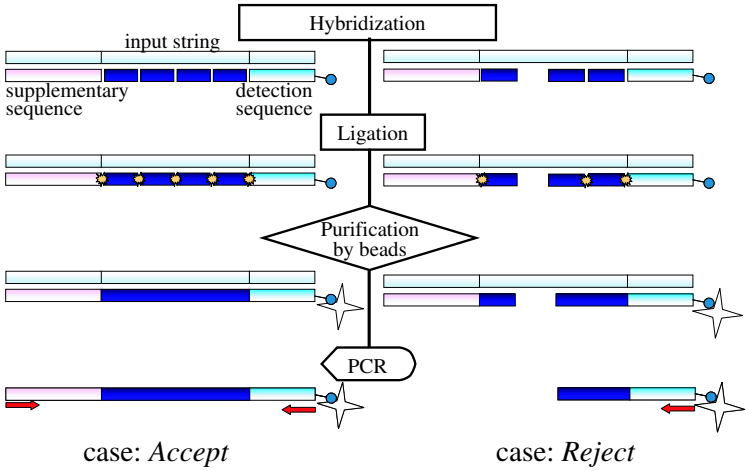


Figure 2: The flowchart of laboratory protocol to execute *in vitro* finite automata which consists of five steps: hybridization, ligation, denature and extraction by affinity purification, amplification by PCR, and detection by gel-electrophoresis. The acceptance of the input string by the automata is the left case, and the rejection is the right case.

Figure 3: (Left:) A 4-states automaton used for this experiment. (Right:) The results of electrophoresis are displayed in gel-like image. Lane (a) is for the input string 1101, lane (b) for 1110, and lane (c) for 1010. Since the full-length band (190 mer) is detected in lane (b) and (c), we determine the automaton accepts two input strings (b) 1110 and (c) 1010.

mer is detected in lane (b) and lane (c). Hence, our *in vitro* experiments have successfully detected that the automaton accepts two input string (b) 1110 and (c) 1010.

3.2 From 2-states to 6-states automata with one input string "111111" of length 6

Our second experiments are 5 different automata of from 2 states to 6 states shown in Fig. 4 (upper) for one input string "111111" of length 6. The automaton (2) accepts the language $(11)^*$, that is, strings with even numbers of symbol '1', (3) accepts the language $(111)^*$, strings repeating three times of 1s, (4) accepts the language $(1111)^*$, strings repeating four times of 1s, (5) accepts the language $(11111)^*$, strings repeating five times of 1s, (6) accepts the language $(111111)^*$, strings repeating six times of 1s. Since 6 is a multiple of 2, 3 and 6, the automata (2), (3) and (6) accept the input string 111111 of length 6.

The results are shown in Fig. 4 (lower) in gel-like image. For the input string 111111, the full-length DNA is of 240 bps (mer). Bands at position of 240 mer are detected in lanes (2), (3) and (6) in Fig. 4. Hence, in our *in vitro* experiments, the automaton (2), (3) and (6) have correctly accepted the input string 111111 and the automaton (4) and (5) have correctly rejected 111111.

3.3 Simulating computation process of finite automata using four-base codons and protein-synthesis mechanism

Sakakibara and Hohsaka [7] have proposed a method using the protein-synthesis mechanism combined with four-base codon techniques to simulate a computation (accepting) process of finite automata. Our approach to make *in vivo* computer is to execute the proposed method on *E. coli* in order to improve the efficiency of the method and further develop a programmable *in vivo* computer. We describe the proposed method using an example of simple finite automaton, illustrated in Fig. 5, which is of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepts input strings with even numbers of symbol 1 and rejects input strings with odd numbers of 1s.

The input symbol '1' is encoded to the four-base subsequence `AGGU` and an input string is encoded into an mRNA by concatenating `AGGU` and `A` alternately and adding `AAUAAC` at the 3'-end. This one-nucleotide `A` in between `AGGU` is used to encode two states $\{s_0, s_1\}$, which is a same technique presented in [8]. For example, a string "111" is encoded into an mRNA:

$$\underbrace{\texttt{AGGU}}_{1}\,\texttt{A}\,\underbrace{\texttt{AGGU}}_{1}\,\texttt{A}\,\underbrace{\texttt{AGGU}}_{1}\,\texttt{AAAUAAC}.$$

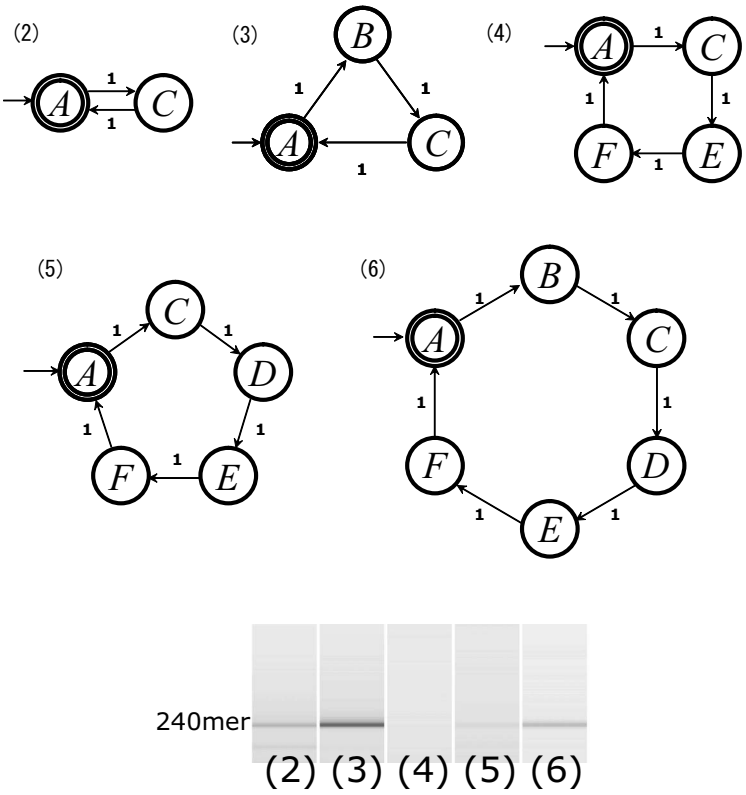


Figure 4: (Upper:) Five different automata of from 2 states to 6 states used for this experiment. (Lower:) The results of electrophoresis are displayed in gel-like image. Lane (2) is for the automaton (2), (3) for (3), (4) for (4), (5) for (5), and (6) for (6). Since the full-length bands (240 mer) are detected in lane (2), (3) and (6), we determine the automata (2), (3) and (6) accepts the input string 111111.

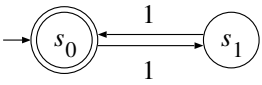


Figure 5: A simple finite automaton of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepting input strings with even numbers of symbol 1 and rejecting input strings with odd numbers of 1s.

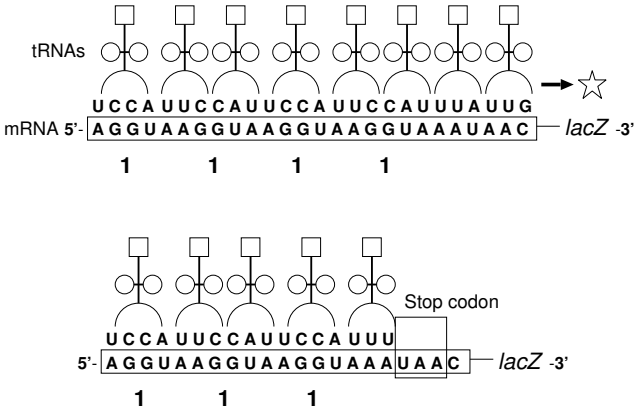


Figure 6: Examples of accepting processes: (Upper) For an mRNA encoding a string "1111", the translation successfully goes through the mRNA and translates the reporter gene of *lacZ* emitting the blue signal. (Lower) For an mRNA encoding a string "111", the translation stops at the stop codon UAG, does not reach to the *lacZ* region and produces no blue signal.

The four-base anticodon (3')UCCA(5') of tRNA encodes the transition rule $s_0 \xrightarrow{1} s_1$, that is a transition from state s_0 to state s_1 with input symbol 1, and the combination of two three-base anticodons (3')UUC(5') and (3')CAU(5') encodes the rule $s_1 \xrightarrow{1} s_0$. Further, the encoding mRNA is linked to *lacZ*-coding RNA subsequence as a reporter gene for the detection of successful computations. Together with these encodings and tRNAs containing four-base anticodon (3')UCCA(5'), if a given mRNA encodes an input string with odd numbers of symbol 1 , an execution of the *in vivo* protein-synthesis system stops at the stop codon, which implies that the finite automaton does not accept the input string, and if a given mRNA encodes even numbers of 1s, the translation goes through the entire mRNA and the detection of acceptance is found by the *blue* signal of *lacZ*. Examples of accepting processes are shown in Fig. 6: (Upper) For an mRNA encoding a string "1111", the translation successfully goes through the entire mRNA and translates the reporter gene of *lacZ* which emits the blue signal. (Lower) For an mRNA encoding a string "111", the translation stops at the stop codon UAA, does not reach to the *lacZ* region and produces no blue signal.

If the competitive three-base anticodon (3')UCC(5') comes faster than the four-base anticodon (3')UCCA(5'), the incorrect translation (computation) immediately stops at the following stop codon UAA.

4 In vivo experiments

We have done some laboratory experiments by following the laboratory protocols presented in [5] to execute the finite automaton shown in Fig. 5, which is of two states $\{s_0, s_1\}$, defined on one symbol '1', and accepts input strings with even numbers of symbol 1 and rejects input strings with odd numbers of 1s.

We tested our method for six input strings, "1", "11", "111", "1111", "11111", and "111111", to see whether the method correctly accepts the input string "11", "1111", "111111", and rejects the strings "1", "111", "11111".

The results are shown in Fig. 7. Blue-colored colonies which indicates the expression of *lacZ* reporter gene have been observed only in the plates for the input strings 11, 1111, and 111111. Therefore, our *in vivo* finite automaton has succeeded to correctly compute the six input strings, that is, it correctly accepts the input strings 11, 1111, 111111 of even numbers of symbol '1' and correctly rejects 1, 111, 11111 of odd number of 1s. To our knowledge, this is the first experimental development of *in vivo* computer and has succeeded to execute an finite-state automaton on *E. coli*.

4.1 A framework of programmable and autonomous in vivo computer on E. coli

Two important issues on developing DNA-based computers are *programmable* and *autonomous*. We realize these two mechanisms by using the main features of our *in vivo* computer based on *E. coli*.

4.1.1 Programmable:

The programmable means that a program is stored as a data (i.e., stored program computer) and any computation can be accomplished by just choosing a stored program. In DNA-based computers, it requires that a program is encoded into a molecule different from the main and fixed units of DNA computer, a molecule encoding programs can be stored and changed, and a change of molecules encoding programs accomplishes any computations.

The main features of our *in vivo* computer enable us to develop a programmable *in vivo* computer. We simply replace a plasmid encoding a state-transition function with other plasmid encoding a different state-transition function, and the *E. coli* cell transformed a new plasmid computes a different finite automaton.

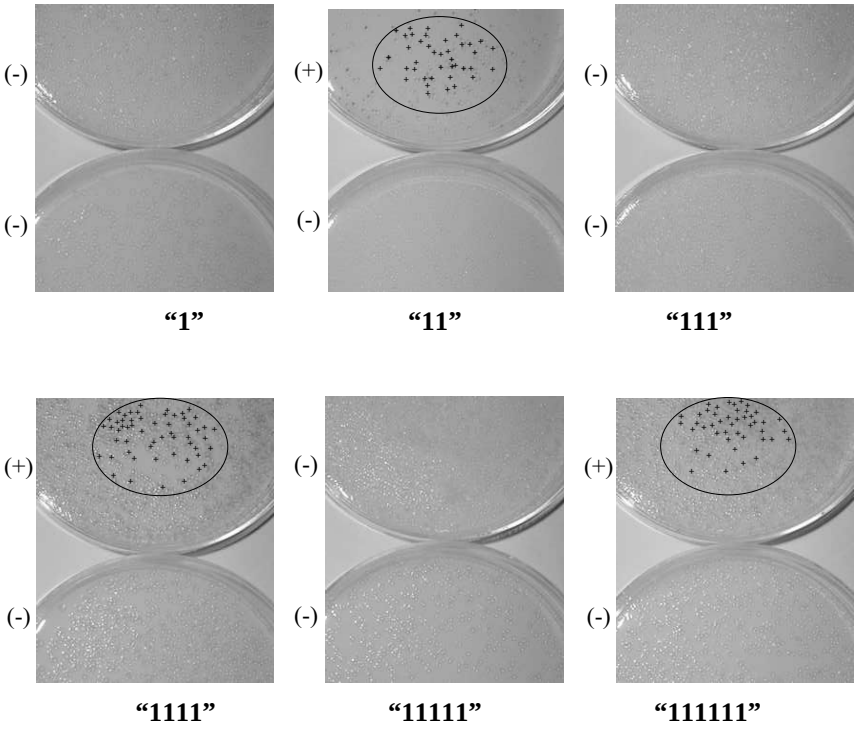


Figure 7: Computation by the *E. coli* cells with plasmids of the input strings: 1, 11, 111, 1111, 11111, 111111. In each panel, the upper plate (part of a LB plate) shows the result in the presence of the suppressor tRNA with UCCU anticodon in the cell, while the lower plate shows the result of control experiment with no suppressor tRNA expressed. The signs (+) and (-) indicate the theoretical values about the expressions of *lacZ* reporter gene: (+) means that the cultured *E. coli* cells must express *lacZ* theoretically, and (-) means it must not express. Circles indicate the blue-colored colony expressing *lacZ*. Therefore, our *in vivo* finite automaton has correctly computed the six input strings, that is, it correctly accepts the input strings 11, 1111, 111111 of even numbers of symbol '1' and correctly rejects 1, 111, 11111 of odd number of 1s.

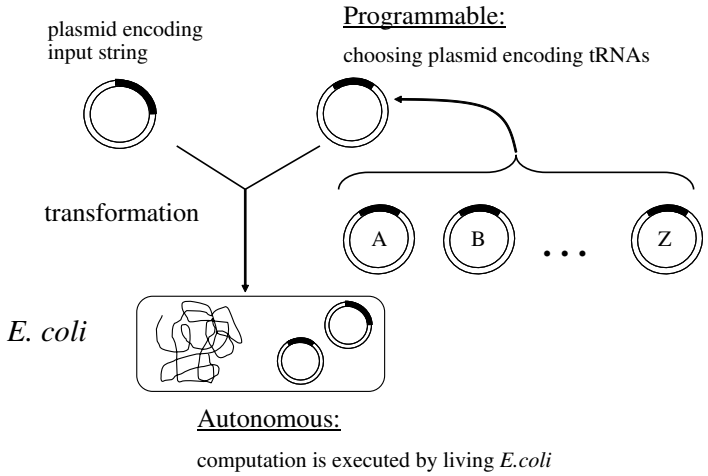


Figure 8: A programmable and autonomous *in vivo* computer system based on *E. coli*.

4.1.2 Autonomous:

The autonomous DNA computers mean that once we set a program and an input data and start a computation, the entire computational process is carried out without any operations from the outside. Our *in vivo* finite automata are autonomous in the sense that the protein-synthesis process which corresponds to a computation (accepting) process of an encoded finite automata is autonomously executed in a living *E. coli* cell and require no laboratory operations from the outside.

5 Discussions

The presented experiments of our *in vivo* finite automata based on *E. coli* propose a kind of population computations in the following two senses: (1) While a computation by one single *E. coli* cell is not effective and accurate, a colony consisting of a large number of *E. coli* cells provides a reliable computation. (2) Since one bacterial cell can multiply to over a million cells overnight, our *in vivo* computation framework offers a massively parallel computation. Further, our *in vivo* finite automata have a quite distinguished feature that an *in vivo* finite automaton is implemented in a living *E. coli* cell; it is not implemented simply by an incubation at a certain temperature.

Acknowledgements

This work is supported in part by Grant-in-Aid for Scientific Research on Priority Area No. 14085205. This work was also performed in part through Special Coordination Funds for Promoting Science and Technology from the Ministry of Education, Culture, Sports, Science and Technology, the Japanese Government, and a grant of Keio Leading-edge Laboratory of Science and Technology (KLL) specified research projects.

References

[1] Benenson, Y., T. Paz-Ellzur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro. Programmable and autonomous computing machine made of biomolecules. *Nature*, 414, 430–434, 2001.

[2] Hohsaka, T., Y. Ashizuka, H. Taira, H. Murakami, M. Sisido. Incorporation of nonnatural amino acids into proteins by using various four-base codons in an *Escherichia coli* in vitro translation system. *Biochemistry*, 40, 11060–11064, 2001.

[3] Hohsaka, T., Y. Ashizuka, H. Murakami, M. Sisido. Five-base codons for incorporation of nonnatural amino acids into proteins. *Nucleic Acids Research*, 29, 3646–3651, 2001.

[4] Kuramochi, J. and Y. Sakakibara. Intensive in vitro experiments of implementing and executing finite automata in test tube. *Proceedings of 11th International Meeting on DNA Based Computers*, London, Ontario, 59–67, 2005.

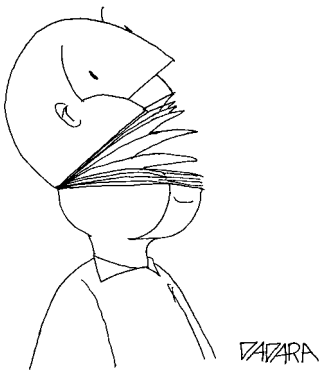
[5] Nakagawa, H., K. Sakamoto, and Y. Sakakibara. Development of an in vivo computer based on Escherichia coli. *Proceedings of 11th International Meeting on DNA Based Computers*, London, Ontario, 68–77, 2005.

[6] Păun, Gh., G. Rozenberg, and A. Salomaa. *DNA Computing*. Springer-Verlag, Heidelberg, 1998.

[7] Sakakibara, Y. and T. Hohsaka. In vitro translation-based computations. *Proceedings of 9th International Meeting on DNA Based Computers*, Madison, Wisconsin, 175–179, 2003.

[8] Yokomori, T., Y. Sakakibara, and S. Kobayashi. A Magic Pot : Self-assembly computation revisited. *Formal and Natural Computing*, LNCS 2300, Springer-Verlag, 418–429, 2002.

TECHNICAL CONTRIBUTIONS



A NOTE ABOUT MERGIBLE STATES IN LARGE NFA*

Pedro García† Manuel Vázquez de Parga‡

Abstract

We propose a new solution to the problem of the existence of a threshold for a regular language L such that every AFN accepting L must contain mergible states. The proposed solution is based in a known property of the universal automaton.

1 Introduction

We face in this work the problem of the possibility of the existence, for a given regular language L, of nondeterministic finite automata of any size without mergible states. Câmpeanu et al [2] have recently enounced this problem in a precise way as follows:

Let L be an arbitrary regular language, and $k \geq 2$ an arbitrary integer. Does it exist (and if "yes", effectively construct it) a constant $E_{L,k}$ such that any NFA of size at least $E_{L,k}$ has at least k mergible states?

The solution proposed in [2] is based in the definition, for every state of a NFA, of two equivalence relations, one of them related to Nerode's right congruence and the other to the syntactic congruence. The threshold obtained in this way is:

$$E_{k,L} = (k-1)P(H_L)N_L P(N_L) + 1$$

where N_L is the index of the Nerode's right congruence (size of the minimal DFA that accepts L), H_L is the index of the syntactic congruence and $P(n)$ is the number of all possible partitions of the set $\{1, ..., n\}$. Note that H_L is bounded above by $N_L^{N_L}$.

That bound seems too high to the authors of the cited work and they propose to obtain a tighter one as future work. We propose in this work a smaller threshold using the definition of Universal Automaton of a language L [1, 3] and its property of representing a canonical automaton for L.

*Work partially supported by the Spanish CICYT under contract TIC2003-09319-C03-02

†Universidad Politécnica de Valencia, pgarcia@dsic.upv.es

‡Universidad Politécnica de Valencia, mvazquez@dsic.upv.es

2 Preliminaries and notation

The free monoid over the alphabet A in denoted as A^*. The empty word is denoted as 1. For any word u of and a language $L \subseteq A^*$, $u^{-1}L$ denotes the residual $\{w \in A^* \mid uw \in L\}$. A non-deterministic automaton (NFA) is denoted $\mathcal{A} = (Q, A, \delta, I, F)$ where I, F are subsets of Q, (initial and final states), and the transition function is $\delta : Q \times A^* \to 2^Q$. $L(\mathcal{A})$ denotes the language accepted by $\mathcal{A}$ and $L_q^{\mathcal{A}} = \{w \in A^* \mid q \in \delta(I, w)\}$, $I_q^{\mathcal{A}} = \{w \in A^* \mid q \in \delta(q, w)\}$, $R_q^{\mathcal{A}} = \{w \in A^* \mid \delta(q, w) \bigcap F \neq \emptyset\}$ denote respectively the left, inner and right languages of the state q. The states $p_1, ..., p_k$ in $\mathcal{A}$ are mergible if and only if [2]:

$$\left(\bigcup_{i=1}^{k} L_{p_i}^{\mathcal{A}}\right)\left(\bigcup_{i=1}^{k} I_{p_i}^{\mathcal{A}}\right)^*\left(\bigcup_{i=1}^{k} R_{p_i}^{\mathcal{A}}\right) \subseteq L$$

Nerode's right congruente will be denoted as $\simeq_L$. The index of $\simeq_L$ is the size of the minimal DFA accepting L and $D = \{u^{-1}L \mid u \in A^*\}$ is the set of states of such a DFA. The equivalente class of the word u is denoted as $[u]_{\simeq_L}$.

The *universal automaton* of the regular language L is defined [4],[5] as $\mathcal{U} = (U, A, \delta, I, F)$ with $U = \{u_1^{-1}L \bigcap ... \bigcap u_k^{-1}L \mid k \geq 0, u_1, ..., u_k \in A^*\}$, $I = \{q \in U \mid q \subseteq L\}$, $I = \{q \in U \mid 1 \in q\}$ and for the states p and q in U and a in A, $q \in \delta(p, a)$ if $q \subseteq a^{-1}p$. The size of U is bounded above by 2^{N_L}.

3 Mergible states and universal automaton

The universal automaton of a language L is a canonical automaton as it is shown in the following proposition:

Proposition 1 ([1]). *Let $\mathcal{U}$ be the universal automaton of the regular language $L \subseteq A^*$. Then,*

1. *$\mathcal{U}$ accepts L.*
2. *For any NFA $\mathcal{A} = (Q, A, \delta, I, F)$ accepting a subset of L, the function φ that assigns to every q in Q, $\varphi(q) = \bigcap_{u \in L_q^{\mathcal{U}}} u^{-1}L$ is an homomorphism that maps $\mathcal{A}$ in $\mathcal{U}$.*

Remark. For every q in U, let us suppose that $\{u_1, ..., u_k\}$ is a maximal set of words over A such that, for $i, j \in \{1, ..., k\}$, if $i \neq j$, then $u_i^{-1}L \neq u_j^{-1}L$ and $q = \bigcap_{i=1}^{k} u_i^{-1}L$. It is easily seen that any word of $[u_i]_{\simeq_L}$ reaches the state $u_i^{-1}L$ and from the definition of the transition function of $\mathcal{U}$, it reaches any state of U which is a subset of $u_i^{-1}L$. Therefore $L_q^{\mathcal{U}} = \bigcup_{i=1}^{k} [u_i]_{\simeq_L}$

Proposition 2. *Let $\mathcal{A}$ be a NFA accepting L and let $\mathcal{U}$ be the universal automaton of L. Let φ be the homomorphism that maps $\mathcal{A}$ in $\mathcal{U}$. Then:*

1. $L_q^{\mathcal{A}} \subseteq L_{\varphi(q)}^{\mathcal{U}}$
2. $R_q^{\mathcal{A}} \subseteq \varphi(q) = R_{\varphi(q)}^{\mathcal{U}}$
3. $I_q^{\mathcal{A}} \subseteq I_{\varphi(q)}^{\mathcal{U}}$

Proof. The statement 1 directly follows from *Remark 1*. The inclusion of the right languages comes from the fact that if state q is reached by words belonging to different classes of $\simeq_L$, their suffixes in L must belong to the intersection of the residuals. Finally, 3 follows from the fact that φ is an automata morphism.

□

Proposition 3. *Let $\mathcal{A}$ be a NFA accepting L and let $\mathcal{U}$ be the universal automaton of L. Let φ the morphism that maps $\mathcal{A}$ in $\mathcal{U}$. Then, if there exist k states of $\mathcal{A}$, $q_1, ..., q_k$ such that $\varphi(q_1) = ... = \varphi(q_k)$, then the states $q_1, ..., q_k$ are mergibles.*

Proof. By the previous proposition

$$\left(\bigcup_{i=1}^{k} L_{q_i}^{\mathcal{A}}\right)\left(\bigcup_{i=1}^{k} I_{q_i}^{\mathcal{A}}\right)^{*}\left(\bigcup_{i=1}^{k} R_{q_i}^{\mathcal{A}}\right) \subseteq L_{\varphi(q_1)}^{\mathcal{U}}(I_{\varphi(q_1)}^{\mathcal{U}})^{*}R_{\varphi(q_1)}^{\mathcal{U}} \subseteq L$$

□

From this result it is possible to obtain a threshold such that automata of greater size must contain mergible states.

Proposition 4. *Let $\mathcal{A}$ be a NFA accepting L being its set of states Q. If $|Q| \geq (k-1)2^{N_L}$, the automaton $\mathcal{A}$ has at least k mergible states.*

Proof. The number of states of the universal automaton is bounded above by 2^{N_L}. Using pigeon-hole principle, if $|Q| \geq (k-1)2^{N_L}$, at least k will have the same image under φ. □

The above threshold can not be reduced. To prove this fact it is enough to define, for every alphabet A a family of minimal DFA such that all the intersections of their residuals are different. For $n \geq 2$, le tus consider $\mathcal{A}(n) = (\mathbb{Z}_n, A, \delta, 0, \mathbb{Z}_n - \{0\})$ With the transition function defined as follows:

For i in $\mathbb{Z}_n$ and $a \in A$, $\delta(i, a) = i + 1 \mod n$.

References

[1] A. Arnold, A. Dicky, M. Nivat. *A note about minimal non-deterministic automata*. Bull. EATCS 47, 166-169,1970.

[2] C. Câmpeanu, N. Sântean, S. Yu. *Mergible states in large NFA* Theoretical Computer Science, 330, 23-34, 2005 .

[3] C. Carrez. *On the minimalization of Non-deterministic Automaton*. Laboratoire de Calcul de la Faculté des Sciences de LUniversité de Lille, 1970.

[4] S. Lombardy. *Approche structurelle de quelques problèmes de la théorie des automates*. Thèse, Ecole Doctorale d'Informatique, Téléconuuications et Electronique de Paris, 2001.

[5] L. Polák *Minimalizations of NFA using the universal automaton* LNCS 3317, 325-326 2005 .

On decision problems for timed automata

Olivier Finkel
Equipe de Logique Mathématique,
U.F.R. de Mathématiques, Université Paris 7
2 Place Jussieu 75251 Paris cedex 05, France.
finkel@logique.jussieu.fr

Abstract

We solve some decision problems for timed automata which were raised by Tripakis in [9] and by Asarin in [3]. In particular, we show that one cannot decide whether a given timed automaton is determinizable or whether the complement of a timed regular language is timed regular.

1 Introduction

We assume the reader to be familiar with the basic theory of timed languages and timed automata (TA) [1].
The set of positive reals will be denoted $\mathcal{R}$. A (finite length) timed word over a finite alphabet Σ is in the form $t_1.a_1.t_2.a_2 \ldots t_n.a_n$, where, for all integers $i \in [1, n]$, $t_i \in \mathcal{R}$ and $a_i \in \Sigma$. It may be seen as a *time-event sequence*, where the $t_i \in \mathcal{R}$ represent time lapses between events and the letters $a_i \in \Sigma$ represent events. The set of all (finite length) timed words over a finite alphabet Σ is the set $(\mathcal{R} \times \Sigma)^\star$. A timed language is a subset of $(\mathcal{R} \times \Sigma)^\star$. The complement (in $(\mathcal{R} \times \Sigma)^\star$) of a timed language $L \subseteq (\mathcal{R} \times \Sigma)^\star$ is $(\mathcal{R} \times \Sigma)^\star - L$ denoted L^c.
We consider a basic model of timed automaton, as introduced in [1]. A timed automaton $\mathcal{A}$ has a finite set of states and a finite set of transitions. Each transition is labelled with a letter of a finite input alphabet Σ. We assume that each transition of $\mathcal{A}$ has a set of clocks to reset to zero and only *diagonal-free* clock guard [1]. As usual, we denote $L(\mathcal{A})$ the timed language accepted (by final states) by the timed automaton $\mathcal{A}$. A timed language $L \subseteq (\mathcal{R} \times \Sigma)^\star$ is said to be timed regular iff there is a timed automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.
Many decision problems for timed automata have been studied and solved partially, see [2] for a survey of these results. Some decision problems were recently

raised by Tripakis in [9] and by Asarin in [3]. We give in this paper the answer to several questions of [9, 3]. In particular, we show that one cannot decide whether a given timed automaton is determinizable or whether the complement of a timed regular language is timed regular.
For that purpose we use a method which is very similar to that one used in [4] to prove undecidability results about infinitary rational relations.

2 Complementability and determinizability

We first state our main result about the undecidability of determinizability or regular complementability for timed regular languages.

Theorem 2.1. *It is undecidable to determine, for a given TA $\mathcal{A}$, whether*

1. *$L(\mathcal{A})$ is accepted by a deterministic TA.*
2. *$L(\mathcal{A})^c$ is accepted by a TA.*

Proof. It is well known that the class of timed regular languages is not closed under complementation. Let Σ be a finite alphabet and let $a \in \Sigma$. Let A be the set of timed words in the form $t_1.a.t_2.a \ldots t_n.a$, where, for all integers $i \in [1, n]$, $t_i \in \mathcal{R}$ and there is a pair of integers (i, j) such that $i, j \in [1, n]$, $i < j$, and $t_{i+1} + t_{i+2} + \ldots + t_j = 1$. The timed language A is formed by timed words containing only letters a and such that there is a pair of a's which are separated by a time distance 1. The timed language A is regular but its complement can not be accepted by any timed automaton because otherwise this timed automaton should have an unbounded number of clocks to check that no pair of a's is separated by a time distance 1, [1].

We shall use the undecidability of the universality problem for timed regular languages: one cannot decide, for a given timed automaton $\mathcal{A}$ with input alphabet Σ, whether $L(\mathcal{A}) = (\mathcal{R} \times \Sigma)^\star$.

Let c be an additional letter not in Σ. For a given timed regular language $L \subseteq (\mathcal{R} \times \Sigma)^\star$, we are going to construct another timed language $\mathcal{L}$ over the alphabet $\Gamma = \Sigma \cup \{c\}$ defined as the union of the following three languages.

- $\mathcal{L}_1 = L.(\mathcal{R} \times \{c\}).(\mathcal{R} \times \Sigma)^\star$
- $\mathcal{L}_2$ is the set of timed words over Γ having not any letters c or having at least two letters c.
- $\mathcal{L}_3 = (\mathcal{R} \times \Sigma)^\star.(\mathcal{R} \times \{c\}).A$, where A is the above defined timed regular language over the alphabet Σ.

The timed language $\mathcal{L}$ is regular because L and A are regular timed languages. There are now two cases.

(1) **First case.** $L = (\mathcal{R} \times \Sigma)^\star$. Then $\mathcal{L} = (\mathcal{R} \times (\Sigma \cup \{c\}))^\star$. Therefore $\mathcal{L}$ has the minimum possible complexity. $\mathcal{L}$ is of course accepted by a deterministic timed automaton (without any clock). Moreover its complement $\mathcal{L}^c$ is empty thus it is also accepted by a deterministic timed automaton (without any clock).

(2) **Second case.** L is strictly included into $(\mathcal{R}\times\Sigma)^\star$. Then there is a timed word $u = t_1.a_1.t_2.a_2 \ldots t_n.a_n \in (\mathcal{R}\times\Sigma)^\star$ which does not belong to L. Consider now a timed word $x \in (\mathcal{R}\times\Sigma)^\star$. It holds that $u.1.c.x \in \mathcal{L}$ iff $x \in A$. Then we have also : $u.1.c.x \in \mathcal{L}^c$ iff $x \in A^c$.
We are going to show that $\mathcal{L}^c$ is not timed regular. Assume on the contrary that there is a timed automaton $\mathcal{A}$ such that $\mathcal{L}^c = L(\mathcal{A})$. There are only finitely many possible global states (including the clock values) of $\mathcal{A}$ after the reading of the initial segment $u.1.c$. It is clearly not possible that the timed automaton $\mathcal{A}$, from these global states, accept all timed words in A^c and only these ones, for the same reasons which imply that A^c is not timed regular. Thus $\mathcal{L}^c$ is not timed regular. This implies that $\mathcal{L}$ is not accepted by any deterministic timed automaton because the class of deterministic regular timed languages is closed under complement.

In the first case $\mathcal{L}$ is accepted by a deterministic timed automaton and $\mathcal{L}^c$ is timed regular. In the second case $\mathcal{L}$ is not accepted by any deterministic timed automaton and $\mathcal{L}^c$ is not timed regular. But one cannot decide which case holds because of the undecidability of the universality problem for timed regular languages. □

Below $TA(n, K)$ denotes the class of timed automata having at most n clocks and where constants are at most K. In [9], Tripakis stated the following problems which are similar to the above ones but with “bounded resources".

Problem 10 of [9]. Given a TA $\mathcal{A}$ and non-negative integers n, K, does there exist a TA $\mathcal{B} \in TA(n, K)$ such that $L(\mathcal{B})^c = L(\mathcal{A})$? If so, construct such a $\mathcal{B}$.

Problem 11 of [9]. Given a TA $\mathcal{A}$ and non-negative integers n, K, does there exist a deterministic TA $\mathcal{B} \in TA(n, K)$ with $L(\mathcal{B}) = L(\mathcal{A})$? If so, construct such a $\mathcal{B}$.

Tripakis showed that these problems are not algorithmically solvable. He asked also whether these bounded-resource versions of previous problems remain undecidable if we do not require the construction of the witness $\mathcal{B}$, i.e. if we omit the

sentence "If so construct such a $\mathcal{B}$" in the statement of Problems 10 and 11.
It is easy to see, from the proof of preceding Theorem, that this is actually the case because we have seen that, in the first case, $\mathcal{L}$ and $\mathcal{L}^c$ are accepted by deterministic timed automata *without any clock*.

3 Minimization of the number of clocks

The following problem was shown to be undecidable by Tripakis in [9].

Problem 5 of [9]. Given a TA $\mathcal{A}$ with n clocks, does there exists a TA $\mathcal{B}$ with $n-1$ clocks, such that $L(\mathcal{B}) = L(\mathcal{A})$? If so, construct such a $\mathcal{B}$.

The corresponding decision problem, where we require only a Yes / No answer but no witness in the case of a positive answer, was left open in [9].
Using a very similar reasoning as in the preceding section, we can prove that this problem is also undecidable.

Theorem 3.1. *Let $n \geq 2$ be a positive integer. It is undecidable to determine, for a given TA $\mathcal{A}$ with n clocks, whether there exists a TA $\mathcal{B}$ with $n-1$ clocks, such that $L(\mathcal{B}) = L(\mathcal{A})$*

Proof. Let Σ be a finite alphabet and let $a \in \Sigma$. Let $n \geq 2$ be a positive integer, and A_n be the set of timed words in the form $t_1.a.t_2.a \ldots t_k.a$, where, for all integers $i \in [1,k]$, $t_i \in \mathcal{R}$ and there are n pairs of integers (i,j) such that $i,j \in [1,k]$, $i < j$, and $t_{i+1} + t_{i+2} + \ldots + t_j = 1$. The timed language A_n is formed by timed words containing only letters a and such that there are n pairs of a's which are separated by a time distance 1. A_n is a timed regular language but it can not be accepted by any timed automaton with less than n clocks.

Let c be an additional letter not in Σ. For a given timed regular language $L \subseteq (\mathcal{R} \times \Sigma)^\star$, we construct another timed language $\mathcal{V}_n$ over the alphabet $\Gamma = \Sigma \cup \{c\}$ defined as the union of the following three languages.

- $\mathcal{V}_{n,1} = L.(\mathcal{R} \times \{c\}).(\mathcal{R} \times \Sigma)^\star$
- $\mathcal{V}_{n,2}$ is the set of timed words over Γ having not any letters c or having at least two letters c.
- $\mathcal{V}_{n,3} = (\mathcal{R} \times \Sigma)^\star.(\mathcal{R} \times \{c\}).A_n$.

The timed language $\mathcal{V}_n$ is regular because L and A_n are regular timed languages. There are now two cases.

(1) **First case.** $L = (\mathcal{R} \times \Sigma)^\star$. Then $\mathcal{V}_n = (\mathcal{R} \times (\Sigma \cup \{c\}))^\star$, thus $\mathcal{V}_n$ is accepted by a (deterministic) timed automaton *without any clock.*

(2) **Second case.** L is strictly included into $(\mathcal{R}\times\Sigma)^\star$. Then there is a timed word $u = t_1.a_1.t_2.a_2 \ldots t_k.a_k \in (\mathcal{R}\times\Sigma)^\star$ which does not belong to L. Consider now a timed word $x \in (\mathcal{R} \times \Sigma)^\star$. It holds that $u.1.c.x \in \mathcal{V}_n$ iff $x \in A_n$.
Towards a contradiction, assume that $\mathcal{V}_n$ is accepted by a timed automaton $\mathcal{B}$ with at most $n-1$ clocks. There are only finitely many possible global states (including the clock values) of $\mathcal{B}$ after the reading of the initial segment $u.1.c$. It is clearly not possible that the timed automaton $\mathcal{B}$, from these global states, accept all timed words in A_n and only these ones, because it has less than n clocks.

But one cannot decide which case holds because of the undecidability of the universality problem for timed regular languages accepted by timed automata with n clocks, where $n \geq 2$. □

Remark 3.2. *For timed automata with only one clock, the inclusion problem, hence also the universality problem, have recently been shown to be decidable by Ouaknine and Worrell [8]. Then the above method can not be applied. It is easy to see that it is decidable whether a timed regular language accepted by a timed automaton with only one clock is also accepted by a timed automaton without any clock.*

4 Concluding remarks

We have restricted here the study to the case of *finite* timed words as in [9, 3]. However the above results can be easily extended to the case of timed regular ω-languages accepted by Büchi timed automata.
The simple idea behind the proofs was already used in [4] and relies heavily on the undecidability of the universality problem.
It could be easily used in other contexts, for instance to study the notion of ambiguity for context-free languages. Ginsburg and Ullian proved in [5] that one cannot decide whether a given context-free language is non-ambiguous or inherently ambiguous. We know that the class of inherently ambiguous context-free languages can be partitioned in an infinite hierarchy by considering the degree of ambiguity of a context-free language [6]. Moreover in recent works of Wich and Naji the context-free languages which are inherently ambiguous of infinite degrees can also be distinguished by considering the growth-rate of their ambiguity with respect to the length of the words [7, 10]. We are not aware of published results about the decidability of membership to subclasses of context-free languages

defined with these notions of degrees of ambiguity.
Using the undecidability of the universality problem for context-free languages and a similar method as in this paper, we can easily prove results like: one cannot decide whether a given context-free language has a degree of ambiguity which is smaller than k, where $k \geq 2$ is a positive integer, or which is smaller than “exponentially ambiguous" (in the sense of Naji and Wich).

References

[1] R. Alur and D. Dill, A Theory of Timed Automata, Theoretical Computer Science, Volume 126, p. 183-235, 1994.

[2] R. Alur and P. Madhusudan, Decision Problems for Timed Automata: A Survey, in Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Revised Lectures. Lecture Notes in Computer Science, Volume 3185, Springer, 2004, p. 1-24.

[3] E. Asarin, Challenges in Timed Languages, From Applied Theory to Basic Theory, Bulletin of the European Association for Theoretical Computer Science, Volume 83, p. 106-120, June 2004.

[4] O. Finkel, Undecidability of Topological and Arithmetical Properties of Infinitary Rational Relations, RAIRO-Theoretical Informatics and Applications, Volume 37 (2), 2003, p. 115-126.

[5] S. Ginsburg and J.S. Ulllian, Ambiguity in Context Free Languages, JACM 13 (1), 1966, p. 62-89.

[6] H. A. Maurer, The Existence of Context Free Languages which are Inherently Ambiguous of any Degree, Dept. of Mathematics, Research Series, University of Calgary, 1968.

[7] M. Naji, Grad der Mehrdeutigkeit Kontextfreier Grammatiken und Sprachen, Diplomarbeit, FB Informatik, Johann-Wolfgang-Goethe-Universität, Frankfurt am Main, 1998.

[8] J. Ouaknine and J. Worrell, On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap, in the Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science , LICS 2004, Turku, Finland, IEEE Computer Society, 2004, p. 54-63.

[9] S. Tripakis, Folk Theorems on the Determinization and Minimization of Timed Automata, in the Proceedings of FORMATS’2003, Lecture Notes in Computer Science, Volume 2791, p. 182-188, 2004.

[10] K. Wich, Exponential Ambiguity of Context-free Grammars, Proc. of 4th Int. Conf. on Developments in Language Theory 1999, World Scientific, Singapore.

THE LANGUAGE OF PRIMITIVE WORDS IS NOT REGULAR: TWO SIMPLE PROOFS*

Pál Dömösi† Géza Horváth ‡

To the memory of Professor Alexandru Mateescu

Abstract

In this paper we give two simple proofs to show that the language Q of all primitive words over a nontrivial alphabet is not regular. We also give two simple proofs to show the non-regularity of well-known sublanguages of Q.

1 Introduction

A word is called primitive if it is not a repetition of another word. Otherwise we speak about a nonprimitive word. It is a widely known (in)famous problem in theoretical computer science whether or not the language Q of all primitive words over a nontrivial alphabet (having at least two letters) is context-free [2, 3]. It is strongly conjectured that this language is not context-free. The problem seems to be simple but nobody could solve it by this time. It is a much more simpler question whether or not Q is regular. Actually, this question has already answered indirectly in [2] and [3]: It was shown that the complementer language of Q is not context-free.

By well-known results (see, for example, [4]), this implies that Q is not deterministic context-free, i.e. Q is not regular. Later it was shown in [6] that Q is not linear which directly implies that Q is not regular.

* This paper was supported by a grant of the Japan-Hungary joint research project given by Japan Society for Promotion of Science and Hungarian Academy of Sciences. It was also supported by a grant of the Hungarian National Foundation for Scientific Research (OTKA T049409). The first author of this work was also supported by a grant from Japan Society for Promotion of Science (No. L04710) and Xerox Foundation UAC grant (1478-2004), U.S.A. The second author of this work was also supported by a grant from Japan Society for Promotion of Science (No. P04028)

†Faculty of Informatics, Debrecen University, Debrecen, Egyetem tér 1., H-4032, Hungary, domosi@inf.unideb.hu

‡Faculty of Informatics, Debrecen University, Debrecen, Egyetem tér 1., H-4032, Hungary, geza@inf.unideb.hu

All the well-known definitions, notions and notations of the formal language theory and the automata theory are omitted. (See, for example, [4] and [5] for the details.)

Let $\mathcal{L}_{DFA}$ denote the class of languages which can be accepted by deterministic finite automata. It is well-known (see, for example, [4, 5]) that $\mathcal{L}_{DFA}$ and the class of regular languages coincide.

The following statement is well-known.

Theorem 1. *(Classical Pumping/Iteration Lemma for Regular Languages)* [1] *Let R be a regular language over Σ. Then there is a constant k, depending on R, such that for each $w \in R$ with $|w| \geq k$ there exist words $x, y, z \in \Sigma^*$ such that $w = xyz$ and $|xy| \leq k, |y| > 0, xy^tz \in R$ for all $t \geq 0$.*

□

It is also well-known (see, for example, [4]) that for every context-free language L and regular language R, $L \cap R$ is also context-free. Therefore, Q is not context-free if there exists a regular language R for which $Q \cap R$ is not context-free. The unsuccessful efforts to find a regular language R having this property are summarized in the following statement.

Theorem 2. [7, 8] *$R \cap Q$ is context-free if $R = (ab^*)^n, n = {p_1}^{f_1} \ldots {p_k}^{f_k}$, where $p_1, \ldots, p_k$ are distinct prime numbers and either $k \leq 4$ or $1/p_1 + \ldots + 1/p_k \leq 4/5$.*

□

The above investigations lead to the next unsolved problem.

Problem 1. [7, 8] *Is $Q \cap (ab^*)^n$ context-free for every positive integer n?*

2 Primitive words and regular languages

Theorem 3. *Q is not regular.*

Proof. Suppose the opposite. Then, applying Theorem 1, there is a constant k, depending on Q, such that for each $w \in Q$ with $|w| \geq k$ there exist words $x, y, z \in \Sigma^*$ such that $w = xyz$ and $|xy| \leq k, |y| > 0, xy^tz \in Q$ for all $t \geq 0$.

It is obvious that for every fixed positive integer k there exists a positive integer m such that the diophantic equation system $(k - \ell)x_\ell + \ell = m,\ \ell = 0, \ldots, k-1$ has a nontrivial solution with appropriate positive integers $x_1, \ldots, x_\ell > 1$. (Then $m = k + ij$, where i is the least common divisor of $2, \ldots, k$ and j is an arbitrary positive integer.) Consider the word $a^k ba^m b$ with $a, b \in \Sigma, a \neq b$. Then for every $\ell = 0, \ldots, k-1$, there exists a positive integer $x_\ell > 1$ such that $a^{(k-\ell)x_\ell + \ell} ba^m b =$

$(a^m b)^2 \notin Q$. Hence, for any $\ell = 0, \ldots, k-1, u = 0, \ldots, \ell$, $xy^t z \notin Q$, whenever $x = a^u, y = a^{k-\ell}, z = a^{\ell-u} b a^m b, t = x_\ell$. On the other hand, because of $k < m$, $xyz = a^k b a^m b \in Q$, a contradiction. □

Next we give another proof of this statement using automata theoretic methods.

Second proof : Suppose to the contrary, there exists a finite, deterministic automaton A, which accepts the language Q. We denote the number of states of A by i. Let a, b be letters of the input alphabet of A, and let $p = (ab^m)^2$ be a word, where $m > i$. The automaton A rejects the word p, because p is a non-primitive word. Since $i < m$, there exist distinct positive integers j, k with $1 \leq j < k \leq m$ such that the automaton is in the same state after reading ab^j and ab^k. From this, we can see that the automaton is in the same state after reading the word ab^m, and the word $ab^{m-(k-j)}$. Finally, the automaton is in the same state after reading the word $p = (ab^m)^2 = ab^m ab^m$ and the word $ab^{m-(k-j)} ab^m$, so the automaton will reject both words, but the word $ab^{m-(k-j)} ab^m$ is primitive, which is a contradiction.

□

Now we give two easy proofs for a partial solution of Problem 1.

Theorem 4. *For every integer $n \geq 2$, $Q \cap (ab^*)^n$ is not regular.*

Proof. Suppose the opposite again. Then, applying Theorem 1, there is a constant k, depending on Q, such that for each $w \in Q$ with $|w| \geq k$ there exist words $x, y, z \in \Sigma^*$ such that $w = xyz$ and $|xy| \leq k, |y| > 0, xy^t z \in Q$ for all $t \geq 0$.

We shall use the fact again that for every fixed positive integer k there exists a positive integer m such that the diophantic equation system $(k-\ell)x_\ell + \ell = m$, $\ell = 0, \ldots, k-1$ has a nontrivial solution with appropriate positive integers $x_1, \ldots, x_\ell > 1$. Consider the word $ab^{k-1}(ab^{m-1})^{n-1}$. Because of $k < m, ab^{k-1}(ab^{m-1})^{n-1} \in Q \cap (ab^*)^n$. If $x = \lambda, y = ab^r, z = b^{k-r-1}(ab^{m-1})^{n-1}$ with $r \leq k-1$ then, clearly, $xy^t z \notin (ab^*)^n, t \neq 1$ and thus $xy^t z \notin Q \cap (ab^*)^n, t \neq 1$. Now let $x = ab^u, y = b^{(k-\ell)}, z = b^{\ell-u-1}(ab^{m-1})^{n-1}$ for some ℓ and u with $\ell = 1, \ldots, k-1, u = 0, \ldots, \ell-1$. Then $xy^{x_\ell} z \notin Q$, i.e. $xy^t z \notin Q \cap (ab^*)^n$ with $t = x_\ell$. Therefore, there are no words $x, y, z \in \Sigma^*$ such that $ab^{k-1}(ab^{m-1})^{n-1} = xyz (\in Q \cap (ab^*)^n)$ and $|xy| \leq k, |y| > 0, xy^t z \in Q \cap (ab^*)^n$ for all $t \geq 0$, a contradiction. □

Now we also show another proof of this statement using automata theoretic methods.

Second proof : Suppose to the contrary, there exists a finite, deterministic automaton A, which accepts the language $Q \cap (ab^*)^n$. We denote the number of states of A by i. Let $p = (ab^m)^n$ be a word, where $m > i$. The automaton A rejects

the word p, because p is a non-primitive word. Since $i < m$, there exist distinct positive integers j, k, with $1 \leq j < k \leq m$ such that the automaton is in the same state after reading ab^j and ab^k. From this, we can see that the automaton is in the same state after reading the word ab^m, and the word $ab^{m-(k-j)}$. Finally, the automaton is in the same state after reading the word $p = (ab^m)^n = ab^m(ab^m)^{n-1}$ and the word $ab^{m-(k-j)}(ab^m)^{n-1}$, so the automaton will reject both words, but the word $ab^{m-(k-j)}(ab^m)^{n-1}$ is a primitive word, which is a contradiction.

□

References

[1] Bar-Hillel,Y., Perles, M., Shamir, E. : *On formal properties of simple phrase structure grammars.* Zeitschrift für Phonetik, Sprachwissenschaft, und Kommunikationsforschung, **14** (1961), 143-172.

[2] Dömösi, P., Horváth, S., Ito, M.: On the connection between formal languages and primitive words. *In: Proc. First Session on Scientific Communication, Univ. of Oradea,* Oradea, Romania, 6-8 June, 1991, *Anns. Univ. of Oradea, Fasc. Mat.,* 1991, 59 – 67.

[3] Dömösi, P., Horváth, S., Ito, M., Formal languages and primitive words, *Publ. Math., Debrecen,* **20** (1993), 315 – 321.

[4] Harrison, M. A. : *Introduction to Formal Language Theory.* Addison-Wesley Publishing Company, Reading, Massachusetts, Menlo Park, California, London, Amsterdam, Don Mils, Ontario, Sidney, 1978.

[5] J.E. Hopcroft, J.E., Ullmann, J.D. : *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, Menlo Park, California, London, Amsterdam, Don Mils, Ontario, Sidney, 1979.

[6] Horváth, S.: Strong interchangeability and nonlinearity of primitive words. *Proc. Worksh. AMiLP'95 (Algebraic Methods in Language Processing, 1995),* Univ. of Twente, Enschede, the Netherlands, 6-8 Dec., 1995, Univ. Twente Service Centrum, 1995, 173-178.

[7] Kászonyi, L., Katsura, M., On the context-freeness of a class of primitive words, *Publ. Math., Debrecen,* bf 51 (1997), 1 – 11.

[8] Kászonyi, L., Katsura, M., Some new results on the context-freeness of languages $Q \cap (ab^*)^n$, *Publ. Math., Debrecen,* **54** (1999), 877 – 884.

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.

72 Coffee or Milk ?

You have two cups, one with coffee, the other one with milk.

Take a spoon. Take a spoonful of coffee from the coffee cup, and poor it in the milk. Stir. Take a spoonful of milk (which will include a little coffee) and poor it in the coffee.

Is there more milk in the coffee or more coffee in the milk ?

73 Bags of coins

You are likely to already know that problem, but look still at the last question.

You have N bags of coins, with K coins each, K being large. A bag contains either only valid coins, or only fake coins. A valid coin weighs 10 grams. A fake one weighs 11 grams. You have a scale that tells you the exact weigh of what you put on the scale.

First, you know that exactly one bag contains fake coins. Find which one with one weigh. How big do you need K to be ? Prove that this value of K_{min} of optimal.

Second, every bag may contain fake coins (possibly none of them, possibly all of them). Find again which ones with one weigh. How big do you need K to be ? Is your K_{min} optimal ? In particular, is there a solution with 10 bags of 500 coins ? What about 10 bags of 400 coins ?

Solutions to Previous Puzzles

70 The 50 prisoners

50 prisoners are gathered and are told the following:

There is a room with a lamp, and nobody but a prisoner can switch it on or off. From time to time, a prisoner is brought into the room. He will see if the lamp is on or off, and is allowed to switch it off or on. At any time, a prisoner is allowed to claim that every prisoner has been at least once in the room. If he is right, every prisoner is freed. If he is wrong, every prisoner is beheaded. Right now, the prisoners do not know whether the lamp is on or off. The choice of who is brought in the room in which order is unknown to the prisoners, it is only known that if no prisoner ever makes a claim, then each prisoner will be brought in the room infinitely often. Right now, the prisoners can talk, so they can decide for a protocol, but then, they will be separated and will not be able to talk or see each other any more.

You are one of the prisoners. Suggest a protocol.

Solution

Choose a prisoner that we will call the leader. If you are a non-leader: when you are brought into the room, and that the light is off, switch it on. Do so twice, then don't do anything. If you are the leader: when you are brought into the room, and that the light is on, then switch it off. Do so 97 times and when you find the light off for the 98^{th} time, claim that every prisoner has been at least once in the room.

There will eventually be a claim, which will be correct:

If the light is initially off, then the 49 non-leaders will switch the light on twice before the leader makes his claim.

If the light is initially on, then 48 non-leaders will switch the light on twice and one will switch it on once, before the leader makes his claim.

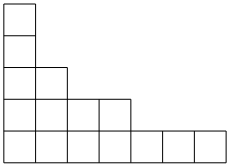
Note that, if we knew that the light is initially off, a faster protocol is

If you are a non-leader: when you are brought into the room, and that the light is off, switch it on. Do so ONCE. If you are the leader: when you are brought into the room, and that the light is on, then switch it off. Do so 48 times and when you find the light off for the 49^{th} time, claim that every prisoner has been at least once in the room.

71 About a few partitions of integers

A partition of an integer n is a decreasing sequence of non-negative integers whose sum is n. For example, (5,3,2,2,1,1,1) is a partition of 15.

A partition can be represented by a "Young table" which is the histogram of the sequence. For example, here is the Young table of (5,3,2,2,1,1,1):



A partition is *All-Different* iff the integers are different. It is *Slowly-Decreasing* iff two successive integers differ by 0 or 1, and the last integer is 1. It is *Odd-Numbered* iff every integer is odd.

Let $AD(n)$, $SD(n)$, $ON(n)$, be the number of *All-Different*, respectively *Slowly-Decreasing*, resp. *Odd-Numbered* partitions. For example, $AD(6) = SD(6) = ON(6) = 4$.[1]

Show that for every n, $AD(n) = SD(n) = ON(n)$

Solution

Transposition (i.e. inverting the x-axis and the y-axis, i.e. applying a symmetry with respect to the $x = y$ axis) is a bijection between Young tables of *All-Different* partitions and Young tables of *Slowly-Decreasing* partitions.

To transform an *All-Different* partition into an *Odd-Numbered* partition, write every number n occurring in the partition as $2^k * p$ where $k \geq 0$ and p is odd, and replace n by 2^k occurrences of p.

This is a bijection. To do the converse transformation, count the number α of occurrences of the odd number p, write α as a sum $2^{k_1} + ... + 2^{k_s}$ of powers of 2, and replace the p's by $2^{k_1} * p$, ..., $2^{k_s} * p$.

[1] The corresponding sets of partitions are:
$\{(6), (5,1), (4,2), (3,2,1)\}$ respectively $\{(3,2,1), (2,2,1,1), (2,1,1,1,1), (1,1,1,1,1,1,1)\}$ respectively $\{(5,1), (3,3), (3,1,1,1), (1,1,1,1,1,1)\}$.

Reports from Conferences

Report on ICALP 2005 / PPDP 2005

32nd Int. Colloquium on Automata, Languages and Programming
and
7th Conference on Principles and Practice of Declarative Programming
July 11-15, 2005, Lisboa, Portugal

Manfred Kudlek

ICALP'05, the 32^{th} (2^5-th) in this series of conferences on Theoretical Computer Science, took place from July 11-15, 2005, together with the workshops from July 10-17, 2005, at **Lisboa**, for the first time in Portugal. It was co-located with PPDP'05 (The 7^{th} ACM/SIGPLAN Conference on *Principles and Practice of Declarative Programming*) which was held from July 11-13, 2005. There was also an *International Conference on Semigroups and Languages (CSL'2005)* in Honour of the 65th birthday of Donald B. McAlister which took place from July 12 to 15, 2005 at Centro de Algebra da Universidade de Lisboa. Conference site for ICALP and PPDP was *Gulbenkian Foundation*, and for the workshops Instituto Superior Técnico.

ICALP'05 was organized by Departamento de Informática e de Centro Informática e Tecnologias da Informação, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa. The Organizing Committee consisted of Luís Caires (co-chair), Margarida Mamede (finance chair), Luís Monteiro (co-chair), António Ravara (workshops co-chair), Vasco Vasconcelos (workshops co-chair), João Seco, José Pacheco, and Anabela Duarte, Sandra Rainha, Filipa Reis, Pedro Adão, Mi- guel Durão, Paula Mangas, Francisco Martins, João Rebelo, Ricardo Silva, Carlos Tanulonis, Hugo Vieira.

PPDP'05 was organized by Centro de Inteligência Artificial, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa. The organizing committee consisted of Pedro Barahona, Francisco Azevedo, and Jorge Cruz.

ICALP'05 was sponsored by EATCS, CITI (Centro de Informática e Tecnologias da Informação, FCT UNL), CLC (Centre for Logic and Computation, IST), Elsevier, FCT (Fundação para a Ciência e Tecnologia), Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Springer-Verlag, Associação de Turismo de Lisboa, Caixa Geral de Depósitos, Microsoft, and Fundação Calouste Gulbenkian. PPDP'05 was sponsored by ACM, FCT (Fundação para a Ciência e Tecnologia), CENTRIA (Centro de Inteligência Artificial), and Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa.

ICALP'05 was accompanied by satellite 8 workshops. Dates, number of invited talks and accepted papers, as well as total numbers of participants (P) and of

such for the workshops only (W) are given below.

WS	D	I	A	P	W
COSMICAH	7.10.	1	7	14	10
DCM	7.10.	1	13	26	18
PDMC	7.10.	1	9	15	9
SOS	7.10.	2	7	23	10
ARSPA	7.16.	2	7	24	9
SD	7.16.	2	14	28	23
PCC	7.16.-17.		14	23	16
WSA	7.16.	1	12	46	33

ICALP'05 itself was attended by 255 participants (including 11 students) from 29 countries, together with satellite workshops by 384 participants from 32 countries. PPDP'05 was attended by 60 participants, 28 of them also attended ICALP such that both conferences together had 416 participants. For ICALP this is a new record. Details are given below where W and P indicate participants only for workshops, and PPDP, respectively.

C	I	W	P	C	I	W	P	C	I	W	P	C	I	W	P
BE	3		1	EE	2		2	IT	18	6	3	RU	2	3	2
BR		1		ES	3	1	2	JP	11		4	SE	9		1
CA	13	2	1	FI	4	1	1	KR	3			SG	1		
CH	6	3	1	FR	30	20	13	NG	1		6	SK	1		
CN		1		GR	2			NL	8	5	2	TW	1		
CZ	10	1		HK	1			NO	1	1		UA	1		
DE	23	24	1	IL	6			PL	4	2	1	UK	10	25	3
DK	5	4		IN	5			PT	27	27	8	US	45	2	8

ICALP'05 covered the following fields in 3 tracks. (A): Algorithms, Automata, Complexity and Games, (B): Logic, Semantics, and Theory of Programming, (C): Security and Cryptography Foundations.

The scientific program consisted of 6 invited lectures, 2 special lectures, and 113 contributions selected from 407 submitted papers from 39 countries, the highest number of submissions for ICALP so far. Statistical details on invited lectures, submitted, and accepted papers are given in the tables below. The scientific program of PPDP consisted of 3 invited lectures (2 joint with ICALP) and 20 contributions. The program of ICALP'05, and those of the workshops, can be found at `http://icalp05.di.fct.unl.pt/`, and that of PPDP'05 at `http://centria.di.fct.unl.pt/`.

	A		B		C		Σ	
	S	A	S	A	S	A	S	A
1	73	9	11	3	10	6	94	18
2	100	26	30	7	33	13	163	46
3	50	20	25	10	19	3	94	33
4	28	8	6	4	11	2	45	14
5	5	2	3				8	2
6	2				1		3	
	258	65	75	24	74	24	407	113

	I	A		B		C		Σ	
		S	A	S	A	S	A	S	A
AU		$1\frac{1}{4}$		$\frac{1}{2}$				$1\frac{3}{4}$	
BE				$\frac{2}{3}$		$2\frac{1}{4}$	$1\frac{1}{4}$	$2\frac{11}{12}$	$1\frac{1}{4}$
BG						$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
BR		$2\frac{1}{3}$						$2\frac{1}{3}$	
CA	1	$16\frac{41}{60}$	$9\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{4}$		$17\frac{14}{15}$	$9\frac{3}{4}$
CH		$6\frac{1}{6}$	$1\frac{1}{2}$	$1\frac{11}{30}$	$\frac{1}{3}$	$5\frac{1}{2}$	1	$13\frac{1}{30}$	$2\frac{5}{6}$
CL		1						1	
CM				$\frac{1}{3}$	$\frac{1}{3}$			$\frac{1}{3}$	$\frac{1}{3}$
CN		$5\frac{2}{3}$		5		$3\frac{1}{2}$		$14\frac{1}{6}$	
CZ		$5\frac{1}{3}$	1					$5\frac{1}{3}$	1
DE	1	$24\frac{5}{12}$	$8\frac{1}{2}$	$7\frac{11}{30}$	$1\frac{5}{6}$	$3\frac{3}{4}$	1	$35\frac{8}{15}$	$11\frac{1}{3}$
DK		$4\frac{13}{30}$	$3\frac{13}{30}$	$1\frac{1}{2}$	$\frac{1}{2}$			$5\frac{14}{15}$	$3\frac{14}{15}$
EE						$1\frac{1}{3}$	$\frac{1}{3}$	$1\frac{1}{3}$	$\frac{1}{3}$
ES		$4\frac{3}{4}$		$1\frac{1}{3}$	$\frac{1}{3}$	2		$8\frac{1}{12}$	$\frac{1}{3}$
FI		$2\frac{1}{6}$	$\frac{5}{6}$					$2\frac{1}{6}$	$\frac{5}{6}$
FR	$2\frac{1}{5}$	$15\frac{2}{5}$	$3\frac{11}{15}$	$9\frac{7}{15}$	3	$5\frac{1}{2}$	$2\frac{1}{2}$	$30\frac{11}{30}$	$9\frac{7}{30}$
GR		$3\frac{3}{4}$	1			2		$5\frac{3}{4}$	1
HK		$4\frac{5}{6}$	1					$4\frac{5}{6}$	1
HR				1				1	
HU		$\frac{1}{3}$		1				$1\frac{1}{3}$	
IE		1				1		2	
IL	1	$21\frac{41}{60}$	$3\frac{43}{60}$	1		1	1	$23\frac{41}{60}$	$4\frac{43}{60}$
IN		$4\frac{1}{3}$	$2\frac{1}{3}$	$\frac{1}{2}$		3	1	$7\frac{5}{6}$	$3\frac{1}{3}$
IT		9	3	9	$3\frac{2}{3}$	$3\frac{1}{2}$	2	$21\frac{1}{2}$	$8\frac{2}{3}$
JP		$6\frac{1}{6}$	$\frac{1}{2}$	$3\frac{1}{6}$		3	1	$12\frac{1}{3}$	$1\frac{1}{2}$
KR		2				8		10	

LV		1						1	
NL		4	$1\frac{3}{4}$	$2\frac{7}{10}$	2	2	1	$8\frac{7}{10}$	$4\frac{3}{4}$
NO		$1\frac{5}{6}$	$\frac{1}{3}$					$1\frac{5}{6}$	$\frac{1}{3}$
PL		$1\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{2}$	$1\frac{1}{2}$		$3\frac{1}{4}$	$1\frac{1}{4}$
PT		1		$\frac{1}{4}$	$\frac{1}{4}$	1		$2\frac{1}{4}$	$\frac{1}{4}$
RU		$2\frac{1}{2}$						$2\frac{1}{2}$	
SE		$8\frac{1}{6}$	$4\frac{1}{2}$	$1\frac{1}{3}$	$1\frac{1}{3}$	1	1	$10\frac{1}{2}$	$6\frac{5}{6}$
SG		$1\frac{1}{3}$		$1\frac{1}{6}$		$1\frac{11}{12}$	$\frac{2}{3}$	$4\frac{5}{12}$	$\frac{2}{3}$
TR		1						1	
TW		1		$\frac{1}{4}$	$\frac{1}{4}$	$1\frac{1}{4}$		$2\frac{1}{2}$	$\frac{1}{4}$
UK	1	$5\frac{1}{5}$	$1\frac{1}{6}$	$9\frac{1}{2}$	$3\frac{2}{3}$	$\frac{1}{4}$		$14\frac{19}{20}$	$4\frac{5}{6}$
US	$1\frac{4}{5}$	$86\frac{1}{60}$	$16\frac{7}{10}$	$15\frac{3}{5}$	$5\frac{1}{2}$	$18\frac{3}{4}$	10	$120\frac{11}{30}$	$32\frac{1}{5}$
YU	1							1	
	7	258	65	75	24	74	24	407	113

The conference site is a centre for cultural, educational and human activities, hosting the fine art collection of the Armenian millionaire and philanthropist Calouste Sarkis Gulbenkian, born in 1869, who came to Portugal in 1942 and died there in 1955, dedicating a great amount of his property to the foundation.

ICALP'05 consisted of 34 sessions (20 in A, 7 in B, 7 in C), held in up to 3 (4 with PPDP'05 on Tuesday) parallel sessions, such that there were some decision problems where to go, in particular for track A which 7 times was split into 2. All contributions were presented, that of Hoeteck Wee by Henry Lin.

ICALP'05 was opened on Monday morning by Luís Monteiro who gave details on the conference and thanks to all authors, PC members, organizers, invited speakers and sponsors. Luís Caires gave hints on Lisboa and explained the ICALP logo, showing *Ponte 25 de Abril*, the bridge over the *Tejo*.

The first invited lecture was *'Cryptography - State of the Science'* by Adi Shamir. It was his second one on ICALP's, an excellent and very interesting, also personal, overview on the history of and recent results in the field. He talked on cryptograpy as a tool, security as a goal, the security paradox, privacy (Clinton in 1989 : *'Enough is enough'*), the lack of theoretical basis in 1980, the development of provable security, and the first crypto conferences (CRYPTO'81 with first chair Ron Rivest and first speaker himself), the progress between 1980 and 1990, on differential and linear cryptoanalysis, the darkest secret (NSA, FBI, CIA, hackers), that there is nothing fundamental on public key systems since 1990, new crypto systems, and some predictions, also asking and answering *'Break RSA by listening to sound of PC? Yes!'*.

The second one, given by Leslie G. Valiant, (already his third invited talk on ICALP's), on *'Holographic Circuits'*, was also an excellent and interesting

summary of complexity theory, complexity classes (*'At* 1000*th* ICALP*: how will it look like?'*), application of linear superpositions to classify algorithms, reductions, relations to physical problems, holographic circuits, and finishing with the question if the existence of a holographic algorithm *'either is true or the most obviously false statement in Computer Science'*.

John C. Mitchell (joint with PPDP) (co-authors Anupam Datta, Ante Derek, Vitaly Shmatikov, Mathieu Turuani) presented the third, also very good one, with *'Security Analysis of Network Protocols: Logical and Computational Methods'* (in the proceedings 'Probabilistic Polynomial-Time Semantics for a Protocol Security Logic'), on protocols, symbolic analysis of protocol security, computational analysis, combination of both, asking also *'If proceedings exist, there is a 2 page list of references. Does anyone have seen them?'*, symbolic and computational models and their combination, using temporal logic, and presenting CPCL (computational protocol compositional logic). He finished with *'Science is a special process'*.

The fourth one by Giuseppe Castagna (joint with PPDP) (co-author Alain Frisch) on *'A Gentle Introduction to Semantic Subtyping'* was a very good and clear introduction on activations and goals (models, circularity (*'the dog bites itself'*, bootstrap (*'dog with shoe'*)), semantic subtyping (*'we need the model to state how types are related'*), subtyping algorithms, applications to a language, and extensions. His conclusion was *'La morale d'histoire est ...'*.

The fifth invited lecture by Burkhard Monien (co-authors Martin Gairing, Thomas Lücking, Karsten Tiemann) on *'Nash Equilibria, the Price of Anarchy and the Fully Mixed Nash Equilibrium Conjecture'* was a good and interesting, well illustrated talk on a *love affair with game theory*, in particular its fields of applications like the WWW and related ones, routing games, weighted games, equilibria and existence of such, and the price of anarchy. At the end he showed his 7 co-authors (including family members!).

Leonid Libkin, with the sixth invited lecture *'Logics for Unranked Trees: An Overview'*, gave a good and fast survey on relation between ranked, unranked, unordered trees and logics (MSO, LTL), efficient tree logic, navigation in trees, and applications (XML). He started with *'Most works have been published other than at ICALP'*, and also mentioned an example of a possible query result *'Titles of all books co-authored by W. G. Bush'*.

There first invited lecture in PPDP'05 was given by Manuel Hermenegildo on *'Abstraction Carrying Code and Resource Awareness'*.

To mention are also the very good and interesting talks by the award winners of best (student) papers. Marius Zimand, solving a long open problem in cryptography and complexity, although that looks very simple, Neeraj Kayal on the complexity of solving equations on finite fields, Damien Pous on weak bisimu-

lation, Mihai Pătraşcu on dynamic data structures (referring also to *Χρονος* and Yggdrasill), and Nicole Schweikardt on communication complexity.

The good talks by Henry Lin on public-coin argument systems, by Omer Horvitz on lower bounds on efficiency of crypto systems, by Nikos Triandopoulos, starting with *'This gives me 45 minutes'*, on hierarchical data processing, and by Klaus Kursawe on protocols for atomic broadcasting, mentioning also the system SINTRA (*'useful topic you can work a lot of time on it'*), should not be omitted, either.

Good and interesting presentations were also given by Tal Moran, with *Anthropo Cryptography* and good illustrations, on cryptographic protocols, by Nicolas Hopper, showing nice illustrations with Alice and Bob in prison trying to escape, on steganography, and by An Braeken on classification of Boolean functions with relation to cryptography as well.

Very nice talks gave Georg Schnitger on new results on finite automata, Sylvain Lombardy on equivalence of $\mathbb{Z}$-automata, Eugen Czeizler on inverse cellular automata, and François Lemieux on groupoids recognizing regular languages. A colourful talk, with pink background, was given by Mathieu Baudet, and a very good and interesting one, also in pink, by Matín Abadi on password encryption. Nice and interesting talks presented also Eric Badouel on Petri algebras, Wan Fokkink on failure semantics, Krishnendu Chatterjee, speaking very fast, with good illustrations, on complexity of stochastic Rabin and Streett games, and Mihalis Yannakakis on probabilistic and stochastic models for decision problems.

Other good and interesting talks presented Scott Diehl on the complexity of SAT, Viraj Kumar on visibly pushdown languages, Jeff Ford on multiparty communication complexity, and Nathan Segerlind on propositional proof complexity. Nice good presentations were given by Douglas Wikström, his first non-crypto talk, on GCD-algorithm in rings of integers, by Robert Špalek on query complexity, by Frédéric Magniez, starting with *'No quantum theory needed'*, on quantum complexity of group commutativity, and by Martin Grohe on extension preservation in finite structures.

The proceedings, edited by Luís Caires, Giuseppe Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung (the highest number of editors so far, and with XXVII+1480 pages the biggest ICALP volume so far), have been published as Springer LNCS 3580. They contain all contributions, and 5 of the invited lectures (that of Giuseppe Castagna only as an extended abstract, and unfortunately not that of Adi Shamir). The proceedings of PPDP'05, edited by Emy Feity and containing all invited lectures and contributions, have been published as a report of *ACM/SIGPLAN*.

The *EATCS* General Assembly was held on Tuesday evening. On it there is a separate report. Branislav Rovan distributed presents to the organizers and PC

Jean-Eric Pin	$10\frac{1}{2}$	**ICALP Contributors**	
Kurt Mehlhorn	$10\frac{1}{6}$		
Juhani Karhumäki	$8\frac{47}{60}$	Michael Rabin	5
Zvi Galil	8	Arnold Schönhage	5
Philippe Flajolet	$7\frac{1}{4}$		
Amir Pnueli	$7\frac{1}{6}$	Burkhard Monien	$4\frac{59}{60}$
Grzegorz Rozenberg	7	Dominique Perrin	$4\frac{5}{6}$
Paul Vitányi	$6\frac{11}{12}$	Zohar Manna	$4\frac{5}{6}$
Mihalis Yannakakis	$6\frac{11}{12}$	Thomas Henzinger	$4\frac{3}{4}$
Claus-Peter Schnorr	$6\frac{1}{2}$	Juraj Hromkovič	$4\frac{7}{10}$
Torben Hagerup	$6\frac{1}{2}$	Denis Thérien	$4\frac{7}{12}$
Karel Čulik II	6	Manfred Droste	$4\frac{1}{2}$
Géraud Sénizergues	6	Robin Milner	$4\frac{1}{2}$
John Reif	$5\frac{3}{4}$	Ming Li	$4\frac{5}{12}$
Walter Vogler	$5\frac{1}{2}$	Moshe Vardi	$4\frac{1}{3}$
Joost Engelfriet	$5\frac{1}{2}$	Maurice Nivat	$4\frac{1}{4}$
Matthew Hennessy	$5\frac{1}{2}$	Moti Yung	$4\frac{1}{4}$
Arto Salomaa	$5\frac{1}{2}$	Volker Diekert	$4\frac{1}{6}$
Juris Hartmanis	$5\frac{1}{3}$	Piotr Berman	$4\frac{1}{6}$
Andrzej Lingas	$5\frac{1}{3}$	Christophe Reutenauer	4
Ronald Book	$5\frac{1}{4}$	Marcel Paul Schützenberger	4
Christos Papadimitriou	$5\frac{1}{4}$	Davide Sangiorgi	4
Christian Choffrut	5	Leslie Valiant	4

chairs. At the end the author of this report gave *EATCS* buttons to those having reached 5 or more full papers on ICALP's (Mihalis Yannakakis, Arto Salomaa, Christos Papadimitriou who due to an error in updating didn't get them earlier, and Andrzej Lingas), as well as to the 5 editors of the proceeedings (see above). The current list of active contributors is given in the table above.

On Thursday afternoon several special events took place.

The first one was the *EATCS* Award ceremony. After Mogens Nielsen gave a short introduction, Jan van Leeuwen explained the decision of the award committee (Mariangiola Dezan-Ciancaglini, Wolfgang Thomas, and himself) to present the award to Robin Milner for his outstanding contributions to the development of mathematical theory of computation. After that he offered the prize to the honoured. In the following Robin Milner gave an excellent talk (it was his third in-

vited or special lecture on ICALP's) *'Software Science from Virtual to Reality'* on 30 years history of machines, programming languages, semantics, modelling of systems, interactive processes (*'No intrinsic mathematical justification for models, lack of dignity of pure science'*). He finished with *'Much more chance than ever before to apply models in practice. Informatics has a long time to go'*.

After that Giorgio Ausiello informed us on the Gödel Prize 2005 which was given on STOC'2005 to Noga Alon, Yossi Matias, Mario Szegedy for their article *'The Space Complexity of Approximating the Frequency Moments'* (JCSS 58, pp 137-147 (1999)). It was already the second one for Mario Szegedy.

Following that was the presentation of best paper (BPA) and best student paper awards (BSPA). Because of 8 equivalent high quality papers there was no BPA in A. The BSPA was given by Giuseppe Italiano to Mihai Pătraşcu (the co-author Corina E. Pătraşcu was not present) for the paper *'On Dynamic Bit-probe Complexity'*. The BPA in B was given by Catuscia Palamidessi to Martin Grohe, and Nicole Schweikardt who gave the presentation (the 3rd author Christoph Koch was not present) for their paper *'Tight Lower Bounds for Query Processing on Streaming and External Memory Data'*, and the BSPA to Damien Pous for the contribution *'Up-to Tecniques for Weak Bisimulation'*. For track C Moti Yung gave the BPA to Marius Zimand for the paper *'Simple Extractors via Constructions of Cryptographic Pseudo-random Generators'*, and the BSPA to Neeraj Kayal for his paper *'Solvability of a System of Bivariate Polynomial Equations over a Finite Field'*. All their presentation were very good and interesting, with a number of nice new results.

The next highlight was the *Elsevier TCS* 30^{th} *Anniversary Ceremony*. Maurice Nivat informed us on the history of *EATCS* and *TCS*. Next Don Sannella explained the reasons for the prize which was given to the author of the mostly cited paper in *TCS* ($\geq$ 475 citations), *'Linear Logic'* by Jean-Yves Girard (*TCS 50*(1), pp 1-102 (1987)). It is an extraordinary paper for its short title, length, and its influence on many other fields. Furthermore, it is the only paper never refereed.

In the following Jean-Yves Girard gave an interesting and very vivid talk on the history of his works, how he came to logics, that it was his first work in Computer Science, and on influence on other fields like linguistics. Adressing Maurice Nivat he said *'Maurice, don't do it for a second time!'*

The social program started on Monday evening 18h with a visit of *Oceanario de Lisboa*. After exhaustive views on oceanic life a reception was held. After 21h we left that interesting place. The second event, starting at 14h on Wednesday, was the excursion to *Sintra*, where we visited *Palácio da Pena* on top, and the village at the foot of *Serra de Sintra* where there was some time for shopping. After that we crossed the mountains to visit the cliffs and light house of *Cabo da Roca*, the western most point of the continent. Along the coast, passing through

Cascais, we returned to Lisboa, arriving there by 20h.

The last event was the conference dinner on Thursday evening at *Estufa Real*, a Royal green house and hunting place, near *Palácio Nacional de Ajuda* in the western parts of Lisboa. It was well after 23h when we left that place. The menu consisted of

Seafood Vol-au-Vent
Grilled Tenderloin Estufa Real
Tulip of Exotic Fruits with Hazelnut Ice Cream
Coffee and Petits Fours
Wines - white and red Adega Vila Bucelas
Water

Thus this anniversary ICALP was successful again, of a high scientific level, very well organized and in a relaxed atmosphere. *Muito abrigado*. Next ICALP will be held in *Venezia*, from July 10 to 14, 2006, together with workshops from July 9 to 16, 2006.

Adeus Lisboa and **Benvenuto a Venezia**.

Report on APC25

Algebraic Process Calculi: The First Twenty Five Years and Beyond
August 1–5, 2005
University of Bologna Residential Center
Bertinoro (Forlì), Italy

Luca Aceto

Now that the dust has settled, and I am back in Reykjavík, I feel that it is appropriate for me to produce a short report on the workshop "Algebraic Process Calculi: The First Twenty Five Years and Beyond" that I co-organized in Bertinoro together with Mario Bravetti, Jim Davies, Wan Fokkink, Andrew D. Gordon, Joost-Pieter Katoen, Faron Moller and Steve Schneider. The event in part was driven by the idea of a "CONCUR re-union" endorsed by Jos Baeten, Jan Bergstra, Tony Hoare, Robin Milner, and Jan Willem Klop. It was sponsored by the Bertinoro International Center for Informatics (BICI), BRICS and Microsoft Research. Unesco offered some scholarships to support the participation of PhD students.

Being one of the organizers makes me extremely biased in the evaluation of the event, but I do believe that the workshop was a great success, both scientifically and socially. First of all, we had about eighty participants, most of whom attended the whole workshop. Secondly, the quality of the presentations and of the ensuing discussions was very high, and pleasant academic chat continued well into the night at the local restaurants and wine bars. Echoing Robin Milner's words at the closing of the workshop, I feel that I have learned much from the event, despite having to miss parts of talks in order to take care of some organizational issues, and that the workshop has given a very good snapshot of the status of research in concurrency theory within the process calculi community.

Since many of the participants wished to contribute talks, we decided to divide the presentations into three main categories: five one hour keynote addresses (one per day), thirty minute "standard talks", and fifteen minute short presentations.

The keynote addresses were delivered to a consistently large audience by Martín Abadi, Rob van Glabbeek, Tony Hoare, Jan Willem Klop and Robin Milner at the beginning of the programme for each day of the workshop.

Tony Hoare kicked off the workshop on Monday, 1 August, by presenting some work of his that aims at using the notion of retraction to unify different theories of concurrency. More precisely, he argued that theories of concurrency can be distinguished by their choice of pre-ordering relation, used to compare processes and to prove their correctness. For example, theories based on CCS are often pre-ordered by some notion of simulation or bisimulation, whereas theories

based on CSP choose some notion of observational refinement (inclusion) as their pre-order. Hoare's strategy for unification of two theories is based on the definition of a function L (for link), which maps the processes of the source theory onto those of the target theory. The ordering relation of the target theory is a composition of the link L with the ordering relation of the source theory. In his talk, Hoare showed how to use SOS transition rules of a structured operational semantics to define a series of such functions, and that their composition is a retraction.

Robin Milner's talk reported on his long term efforts to base a broad theory of processes on the way that placing, or locality, interacts with linking or connectivity. This theme has been around in many forms—for instance, in the ambient calculus of Cardelli and Gordon, and in Parrow's net-like variants of the π-calculus. In his lecture, Milner introduced bigraphs, and traced, by means of examples, the ways that placing and linking collaborate to yield many aspects of process activity: remote interaction, suppression or enablement of activity, parametric reaction regimes, and binding (static or dynamic).

Jan Willem Klop's seminar presented a lovely historical reflection on his development of the theory of ACP together with Jan Bergstra (who, unfortunately, could not be with us in Bertinoro because of previous engagements). He showed us photos of Jan and him as young men, copies of slides used in early talks reporting on their ACP work, and delighted us with a description of some on-going work with Clemens Grabmayer and Bas Luttik on the development of a geometry of processes.

Jan Willem Klop's presentation was followed by a forty five minute address by Jos Baeten, one of the prime movers behind the CONCUR conference series and the early CONCUR projects funded by the European Union. Jos presented some recent joint work with Mario Bravetti that aims at contributing to the unification of work done in the three classic process algebras CCS, CSP and ACP. He defined a generic process algebra where each basic mechanism of CCS, CSP and ACP is expressed by an operator, and which can be used as an underlying common language. He showed an example of the advantages of adopting such a language instead of one of the three more specialized algebras: producing a complete axiomatization of finite-state behaviours.

Martín Abadi delivered his address the morning after the conference excursion to Ravenna, where we had the chance to admire some of its world famous and beautiful mosaics, and the conference dinner in Cesenatico. Despite the late return to Bertinoro, a packed room listened to his well paced lecture on the use of process calculi to model and analyze security mechanisms. He focused on security protocols and their study through dialects of the π-calculus, and reviewed the development of those dialects and their applications. He also presented some of his recent progress in providing complexity-theoretic justifications for symbolic models of cryptography.

In the last of the invited addresses, Rob van Glabbeek raised the question of how to specify timeouts in process algebra, and argued that the basic formalisms fall short in this task—at least if we consider generative rather than reactive machines. The lecture was "vintage Rob van Glabbeek", and generated a fair amount of discussion.

The thirty minute and the short presentations offered a bird's eye view of the field of algebraic process calculi and other topics in concurrency theory. There were so many interesting talks that I cannot hope to do justice to all that was said at the workshop in the short space of this report. Here, it suffices to say that the audience was treated to a collection of talks

- putting the history of the development of the theory of some process calculi into perspective—e.g., talks by Stephen Brookes and Joel Ouaknine on the development of the theory of CSP, by Rocco De Nicola on Klaim, by Davide Sangiorgi on the history of the notion of bisimulation, and by Kim G. Larsen on the notion of quotienting—,

- presenting applications of ideas of process calculi in other fields of (computer) scientific research—e.g., talks by Samson Abramsky on quantum computation, Corrado Priami on the use of process calculi in biology, by Ed Brinksma on embedded systems and by Gianluigi Zavattaro on web services—,

- describing the use of process calculi in the description and analysis of non-trivial computing systems and discussing the strengths and weaknesses of these formalisms in applications—e.g., talks by Hubert Garavel on interfacing process calculi with the real world, by Jan Friso Groote on the applications of μCRL as well as suitably thought provoking addresses by Uwe Nestmann and Peter Sewell—or

- reporting on the development of variations on classic process calculi to account for the description of phenomena in reactive computation like real-time or stochastic behaviour, or issues related to security in computation—e.g., talks by Cedric Fournet, Catuscia Palamidessi, Holger Hermanns and Christel Baier.

The complete list of the talks that were delivered at the workshop is available at `http://www.cs.auc.dk/~luca/BICI/PA-05/abstracts.html`, and the organizers plan to make the slides for all of the talks available there soon. This will allow the readers of this piece to experience, at least in part, some of the exciting presentations that we were treated with in Bertinoro. For the time being, I encourage all of you to browse through volume NS-05-3 of the BRICS Notes Series (edited by Andrew D. Gordon and myself) that contains a collection of short

essays that we commissioned to several members of the research community on algebraic process calculi. The organizing committee decided from the very start that the volume should *not* consist of a collection of long technical articles. (After all, there are already plenty of standard outlets for those contributions.) Rather, we decided to solicit from the participants at the workshop, and other selected members of our community, short essays on the theme of algebraic process calculi. Some ideas for papers that we proposed to potential contributors were: a reminiscence about the early days; a prospectus for future research; a statement of challenges or open problems; a history of a thread of research; a critical assessment of an idea or a project; a review of a seminal paper and its impact; or even a self-contained technical observation. The response from the colleagues we contacted was overwhelmingly positive, and beyond our most optimistic expectations. I trust that you will enjoy reading their varied and interesting contributions. As we did not seek scientific articles in the usual sense, the contributions in the volume are unrefereed.

Apart from our sponsors, on behalf of the organizing committee, I thank Elena Della Godenza (University Residential Centre of Bertinoro) for her tireless organizational and secretarial assistance at all times, and Uffe Engberg (BRICS) for his work in the production of the aforementioned volume of essays on algebraic process calculi.

Overall, I feel that this workshop gave an excellent testimony of the vitality of research in the field of algebraic process calculi, and offered further confirmation of the suitability of the facilities in Bertinoro for the hosting of research workshops, schools and other high quality events in Computer Science. I encourage the readers of the Bulletin interested in organizing workshops on all aspects of Theoretical Computer Science to consider the University Residential Centre of Bertinoro as a possible location for their events. I, for one, might be tempted to organize a third workshop on process calculi there within a couple of years.

Pictures from APC 25 (by L. Aceto)

The participants

REPORT ON CPM'2005

16th Annual Symposium on Combinatorial Pattern Matching
Jeju Island, Korea, June 19 – 22, 2005

Shiri Dori

The 16th annual Symposium on Combinatorial Pattern Matching was held 19–22 June 2005 in the picturesque Jeju Island in Korea. The venue was the Ramada Hotel in Jeju City, along the coast.

The conference was organized by Dong Kyue Kim, Yoo-Jin Chung, Sung-Ryul Kim, Heejin Park, Jeong Seop Sim and Jin Wook Kim, and led by the venerable Kunsoo Park. Upon arrival, foreign participants were debriefed regarding Korean family names, so that they should not wonder whether the organizers are all related to each other (they are not).

Two invited talks were presented. The first by Ming Li titled *Super patterns and their applications in bioinformatics and finance*, and the second by Esko Ukkonen - *In the search of motifs (and other hidden structures)*. There were 37 interesting talks, and over 80 participants from fifteen countries attended. Further details of the program can be found on the conference web site, `http://theory.snu.ac.kr/cpm2005/`.

Social highlights included a formal banquet and two excursions. The banquet consisted of a formal dinner in Western style. The dinner was followed by an enchanting show of Korean traditional music, presented by the lovely Sukhie Moon, Kunsoo Park's wife, and performed by four students. In the first excursion, participants toured through Eastern Jeju, taking a cruise around some smaller islands and viewing sunrise peak; the tour continued to a beach, and wrapped up with a visit to a fun maze park. The second excursion was spontaneously organized, due to requests of many participants to climb Mount Halla, an extinct volcano and the largest mountain in South Korea. This excursion occurred on June 23, after the conference officially ended.

CPM 2006 will be hosted in Barcelona, Spain in July 5–7, as announced by Gabriel Valiente, who will be co-chair. Surely, next year's conference will be as successful as CPM 2005.

List of Talks

- *Sharper Upper and Lower Bounds for an Approximation Scheme for CONSENSUS-PATTERN* by Brona Brejova, Daniel G. Brown, Ian M. Harrower, Alejandro Lopez-Ortiz, and Tomas Vinar.

- *On the Longest Common Rigid Subsequence Problem* by Bin Ma and Kaizhong Zhang.
- *Text Indexing with Errors* by Moritz G. Maass and Johannes Nowak.
- *A New Compressed Suffix Tree Supporting Fast Search and its Construction Algorithm Using Optimal Working Space* by Dong Kyue Kim and Heejin Park.
- *Succinct Suffix Arrays based on Run-Length Encoding* by Veli Makinen and Gonzalo Navarro.
- *Linear-time Construction of Compressed Suffix Arrays Using $O(n \log^e n)$-Bit Working Space for Large Alphabets* by Joong Chae Na.
- *Faster algorithms for delta, gamma-matching and related problems* by Peter Clifford, Raphael Clifford, and Costas Iliopoulos.
- *A fast algorithm for approximate string matching on gene sequences* by Zheng Liu, Xin Chen, James Borneman, and Tao Jiang.
- *Approximate Matching in the L_1 Metric* by Amihood Amir, Ohad Lipsky, Ely Porat, and Julia Umanski.
- *An Efficient Algorithm for Generating Super Condensed Neighborhoods* by Luis M. S. Russo and Arlindo L. Oliveira.
- *The median problem for the reversal distance in circular bacterial genomes* by E. Ohlebusch, M.I. Abouelhoda, K. Hockel, and J. Stallkamp.
- *Using PQ Trees for Comparative Genomics* by Gad M. Landau, Laxmi Parida, and Oren Weimann.
- *Hardness of Optimal Spaced Seed Design* by Francois Nicolas and Eric Rivals.
- *Weighted Directed Word Graph* by Meng Zhang and Yi Zhang.
- *Construction of Aho Corasick Automaton in Linear Time for Integer Alphabets* by Shiri Dori and Gad M. Landau.
- *An extension of the Burrows Wheeler Transform and Applications to Sequence Comparison and Data Compression* by Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, and Marinella Sciortino.
- *DNA Compression Challenge Revisited : A Dynamic Programming Approach* by Behshad Behzadi and Fabrice Le Fessant.
- *On the Complexity of Sparse Exon Assembly* by Carmel Kent, Gad M. Landau, and Michal Ziv-Ukelson.
- *An Upper Bound on the Hardness of Exact Matrix Based Motif Discovery* by Paul Horton and Wataru Fujibuchi.
- *Incremental Inference of Relational Motifs with a Degenerate Alphabet* by

Nadia Pisanti, Henry Soldano, and Mathilde Carpentier.

- *Speeding up Parsing of Biological Context-Free Grammar* by D. Fredouille and C.H. Bryant.
- *A New Periodicity Lemma* by Kangmin Fan, W. F. Smyth, and R. J. Simpson.
- *Two Dimensional Parameterized Matching* by Carmit Hazay, Moshe Lewenstein, and Dekel Tsur.
- *An Optimal Algorithm for Online Square Detection* by Gen-Huey Chen, Jin-Ju Hong, and Hsueh-I Lu.
- *A Simple Fast Hybrid Pattern-Matching Algorithm* by Frantisek Franek, Christopher G. Jennings, and W. F. Smyth.
- *Prefix-Free Regular-Expression Matching* by Yo-Sub Han, Yajun Wang and Derick Wood.
- *Reducing the size of NFAs by using equivalences and preorders* by Lucian Ilie, Roberto Solis-Oba, and Sheng Yu.
- *Regular expression constrained sequence alignment* by Abdulla N. Arslan.
- *A linear tree edit distance algorithm for similar ordered trees* by Helene Touzet.
- *A Polynomial Time Matching Algorithm of Ordered Tree Patterns having Height-Constrained Variables* by Kazuhide Aikou, Yusuke Suzuki, Takayoshi Shoudai, Tomoyuki Uchida, and Tetsunhiro Miyahara.
- *Assessing the significance of Sets of Words* by Valentina Boeva, Julien Clement, Mireille Regnier, and Mathias Vandenbogaert.
- *Inferring a Graph from Path Frequency* by Tatsuya Akutsu and Daiji Fukagawa.
- *Exact and Approximation Algorithms for DNA Tag Set Design* by Ion I. Mandoiu and Dragos Trinca.
- *Parametric Analysis for Ungapped Markov Models of Evolution* by David Fernandez-Baca and Balaji Venkatacha.
- *Linear Programming for Phylogenetic Reconstruction Based on Gene Rearrangements* by Jijun Tang and Bernard M.E. Moret.
- *Identifying similar surface patches on proteins using a spin-image surface representation* by M. E. Bock, G. M. Cortelazzo, C. Ferrari, and C. Guerra.
- *Mass Spectra Alignments and their Significance* by Sebastian Bocker and Hans-Michael Kaltenbach.

Report on WG 2005

The 31st International Workshop on Graph Theoretic Concepts in Computer Science

Hans L. Bodlaender

From June 23 – 25, 2005, the **31st International Workshop on Graph-Theoretic Concepts in Computer Science**, WG 2005, was held in the city of Metz. The number of submitted papers was an all-time record of 125 (of which one was withdrawn). From these, 38 papers were accepted for presentation. In addition to these 38 regular lectures, there were two invited lectures. There were over 90 participants, from all over the world.

The meeting started with a welcome reception which gave the participants an excellent taste of the French cuisine on the evening of June 22. Thursday, June 23 started with a welcome word. Ludek Kucera then spoke words to remember Ondrej Sykora, who died May 12 this year from an illness.

There were two invited lectures. The meeting started with the first invited lecture by Georg Gottlob: *Hypertree decompositions: structure, algorithms, and applications*. Applications, e.g., from constraint satisfaction and database query optimisation, motivate the concepts of hypertree width and hypertree decomposition: these are related to treewidth and tree decomposition, but use hypergraphs instead of graphs. The second invited lecture was given Friday afternoon, by Gregory Kucherov: *Combinatorial search on graphs motivated by bioinformatics applications: a case study and generalizations*; the talk reviewed several results on combinatorial problems motivated from applications from bioinformatics.

Each of the 38 accepted papers was presented at the meeting by one of the authors. The talks showed a variety of topics concerning graphs, with many of their aspects in relation to computer science. Many talks gave new or better algorithms for graph problems, some with a theoretical formulation, and some with an application. The overall quality of the presented results and the presentations was high. These talks and the two invited lectures made that WG 2005 had an excellent scientific program.

The participants of WG 2005 received not one, but two excellent conference dinners, in two different restaurants, one at Thurday evening, and one at Friday evening. On Friday evening there was also a very interesting guided tour through the charming city of Metz.

Many thanks are due to Dieter Kratsch and his crew for the excellent organisation of WG 2005. This Workshop on Graph Theoretic Concepts in Computer Science was (again) a very pleasant meeting, with a lot of interesting science, excellent food, and good interactions between the participants. WG 2006 will

be held near Bergen in Norway, and this is a meeting to look forward to. The scientific program of WG 2005 is given below.

Thursday, June 23, 2005

8:50 Opening of WG 2005

9:00-9:50 Invited Talk: Georg Gottlob (Vienna, Austria)
Hypertree Decompositions: Structure, Algorithms and Applications

9:50-10:15 Divesh Aggarwal, Shashank K. Mehta, Jitender S. Deogun
Domination Search on Graphs with Low Dominating-Target Number

10:15-10:40 Christophe Crespelle, Christophe Paul
Fully dynamic algorithm for modular decomposition and recognition of permutation graphs

11:00-11:25 Sang-il Oum
Approximating Rank-width and Clique-width Quickly

11:25-11:50 Omer Gimenez, Petr Hlineny, Marc Noy
Computing the Tutte Polynomial on Graphs of Bounded Clique-Width

11:50-12:15 Frank Gurski, Egon Wanke
Minimizing NLC-width is NP-complete

12:15-12:40 Frederic Havet, Jean-Sebastien Sereni
Channel assignment and improper choosability of graphs

14:15-14:40 Daniel Meister
Computing treewidth and minimum fill-in for permutation graphs in linear time

14:40-15:05 Mathieu Liedloff, Ton Kloks, Jiping Liu, Sheng-Lung Peng
Roman domination over some graphs classes

15:05-15:30 Jiri Fiala, Daniel Paulusma, Jan Arne Telle
Algorithms for comparability of matrices in partial orders imposed by graph homomorphisms

15:30-15:55 Zuzana Beerliova, Felix Eberhard, Thomas Erlebach, Alexander Hall, Michael Hoffmann, Matus Mihalak, L. Shankar Ram
Network Discovery and Verification

16:15-16:40 Emeric Gioan
Complete graph drawings up to triangle mutations

16:40-17:05 Derek G. Corneil, Feodor F. Dragan, Ekkehard Köhler, Chenyu Yan
Collective tree 1-spanners for interval graphs

17:05-17:30 Van Bang Le, Raffaele Mosca, Haiko Müller
On stable cutsets in claw-free graphs and planar graphs

Friday, June 24, 2005

9:00-9:25 Prosenjit Bose, Vida Dujmovic, David R. Wood
Induced Subgraphs of Bounded Degree and Bounded Treewidth
9:25-9:50 Pinar Heggernes, Daniel Lokshtanov
Optimal broadcast domination of arbitrary graphs in polynomial time
9:50-10:15 A. Berry, R. Krueger, G. Simonet
Ultimate generalizations of LexBFS and LEX M
10:15-10:40 Stavros D. Nikolopoulos, Leonidas Palios
Adding an Edge in a Cograph

11:00-11:25 Michael Gatto, Riko Jacob, Leon Peeters, Anita Schöbel
The Computational Complexity of Delay Management
11:25-11:50 Daniel Goncalves, Mickaël Montassier
Acyclic choosability of graphs with small maximum degree
11:50-12:15 Shin-ichi Nakano, Takeaki Uno
Generating Colored Trees
12:15-12:40 Ephraim Korach, Margarita Razgon
Optimal hypergraph tree-realization

14:15-15:05 Invited Talk: Gregory Kucherov (Nancy, France)
Combinatorial search on graphs motivated by bioinformatics applications: a case study and generalizations
15:05-15:30 Guillaume Blin, Guillaume Fertin, Danny Hermelin, Stephane Vialette
Fixed-parameter algorithms for protein similarity search under mRNA structure constraints
15:30-15:55 Peter Damaschke
On the Fixed-Parameter Enumerability of Cluster Editing

16:15-16:40 Manuel Bodirsky, Daniel Kral
Locally Consistent Constraint Satisfaction Problems with Binary Constraints
16:40-17:05 Robert Elsaesser, Thomas Sauerwald
On Randomized Broadcasting in Star Graphs
17:05-17:30 Torsten Tholey
Finding Disjoint Paths on Directed Acyclic Graphs

Saturday, June 25, 2005

9:00-9:25 Eric Angel, Evripidis Bampis, Laurent Gourves
Approximation algorithms for the bi-criteria weighted max cut problem

9:25-9:50 Akihisa Kako, Takao Ono, Tomio Hirata, Magnus M. Halldorsson
Approximation Algorithms for the Weighted Independent Set Problem

9:50-10:15 Erik Jan van Leeuwen
Approximation Algorithms for Unit Disk Graphs

10:15-10:40 P. Berthome, S. Lebresne, K. Nguyen
Computation of chromatic polynomials using triangulations and clique trees

11:00-11:25 Fedor Fomin, Frederic Mazoit, Ioan Todinca
Computing branchwidth via efficient triangulations and blocks

11:25-11:50 Joachim Kneis, Daniel Moelle, Stefan Richter, Peter Rossmanith
Algorithms Based on Treewidth of Sparse Graphs

11:50-12:15 Michael Dom, Jiong Guo, Falk Hüffner, Rolf Niedermeier
Extending the tractability border for Closest Leaf Powers

12:15-12:40 Joachim Giesen, Dieter Mitsche
Bounding the Misclassification Error in Spectral Partitioning in the Planted Partition Model

14:15-15:40 Ross M. McConnell, Fabien de Montgolfier
Algebraic Operations on PQ Trees and Modular Decomposition Trees

14:40-15:05 Yoshio Okamoto, Takeaki Uno, Ryuhei Uehara
Linear-Time Counting Algorithms for Independent Sets in Chordal Graphs

15:05-15:30 Anne Berry, Alain Sigayret, Jeremy Spinrad
Faster Dynamic Algorithms for Chordal Graphs, and an Application to Phylogeny

15:30-15:55 Stavros D. Nikolopoulos, Leonidas Palios
Recognizing HHDS-free Graphs

15:55-16:05 Closing Remarks

REPORT ON AFL 2005

11th International Conference of Automata and Formal Languages
May 17–20, 2005
Dobogókő, Hungary

Manfred Kudlek

AFL'05 was held from May 17-20, 2005, at Dobogókő, about 35 km northwest from Budapest, situated nicely in Visegrádi Heggység (Visegrád Mountains). It was the 11^{th} conference in this series, founded by István Peák in 1980. Conference site was Hotel Nimród at 700 m above sea level, where the majority of the participants stayed, some in Platán Panzió.

AFL'05 was organized by the *Institute of Informatics, University of Szeged*. The organizing committee consisted of ZOLTÁN ALEXIN, ZOLTÁN ÉSIK (chair), ZSOLT GAZDAG, ÉVA GOMBÁS, BALÁZS IMREH (chair), SZABOLCS IVÁN, ZSOLT KAKUK, LÓRÁND MUZAMEL, ZOLTÁN L. NÉMETH, and ANTAL PUKLER.

It was supported by *Institute of Informatics of University of Szeged, EATCS, Hungarian Academy of Sciences, Fund for Szeged*, and *Novadat Company*.

AFL'05 was attended by 59 participants from 16 countries, their distribution as follows :

HU	21	CZ	4	PL	2	AT	1	IN	1	SK	1
DE	8	FR	4	RO	2	ES	1	JP	1		
FI	6	CA	3	US	2	IN	1	RU	1		

The scientific program consisted of 7 invited lectures and 21 contributions, selected from 37 submissions.

C	I	A	C	I	A	C	I	A	C	I	A
AT		1	ES		$\frac{1}{2}$	IN	1		RO		1
CA		2	FI	1	3	IT	1		US	1	1
CZ		2 $\frac{2}{3}$	FR	1	2	JP		1			
DE	2	2 $\frac{1}{3}$	HU		3	PL		1			

All contributions were presented in 12 sessions by one of the authors. The program can be found at `http://www.inf.u-szeged.hu/afl05`. Session 7 and 8 were shifted from Thursday to Wednesday afternoon, and the Friday afternoon sessions were 30 minutes earlier.

The conference was opened on Tuesday afternoon by ZOLTÁN ÉSIK, talking on the history of AFL and its founder ISTVÁN PEÁK, who died in 1990.

In the first invited lecture *'Weighted Automata and Weighted Logics'* Manfred Droste gave a very good and interesting survey on the relation between automata and logic, on the effect of weights in automata and their correspondence in logic. Thomas Wilke (co-author Ralf Küsters) presented a nice and interesting second one with *'Automated Analysis of Cryptographic Protocols by Automata-theoretic Means'*, talking on rewriting and automata techniques, finite model checking, and logic programming for cryptographic protocols.

Another good and interesting third one was presented by Antonio Restivo (co-authors Sabrina Mantaci, Marinella Sciortino) with *'The Burrows-Wheeler Transform from Data Compression to Combinatorics on Words'*. He started with *'A very special technique in a very special application'*, talked on the relation of combinatorics of words to biological sequences, and when asked by Stephen Bloom *'What is the meaning of this picture ?'* he answered *'The Gorilla is very close to you I think'*. Kamal Lodaya also gave a nice presentation with the forth invited lecture *'Looking Back at Process Algebra'*, starting with *'Time reversal'*, *'A lunch talk'*, and *'Only a lot of questions !'*, and presenting a survey on several algebraic structures in the field. He also gave an example of a coffee machine with Indian currency.

An excellent and very interesting fifth invited talk, *'Automata on Linear Orderings : Complementation'*, presented Olivier Carton on automata on linear and scattered orderings, cuts of linear orderings, and closure of rational and recognizable sets under complementation.

Juhani Karhumäki, with *'Combinatorics on Words and Complexity'* gave a very good sixth invited talk on various complex phenomena as aperiodic tilings, the relation of complexity of words to fields like Mathematics, Physics, Biology, and in particular to Mathematical complexity. He started with thanks to Michal Kunc for the excellent excursion the day before, announced the conference on words in Montreal, and also mentioned the chimney in Turku with Fibonacci numbers (not Fibonacci words).

Also *'Some Remarks on Regular Words'* by Stephen L. Bloom was a good seventh one on categorial theories of words, axiomatization of regular operations, and two special theories of words. It started with *'Short talk on long words'*, with exercises, and the question *'What is the most general structure, associative and having fixed points ?'*.

Good and interesting presentations were given by Werner Kuich on various semirings, referring also to the talk of Manfred Droste, by Victor Mitrana on multiple patterns and decision problems related to them, and by Martin Plátek on relations between generalized contextual grammars and reset automata.

Zoltán L. Németh gave a nice talk on infinite bi-posets and bi-semigroups, Libor Polák an interesting one on conjunctive varieties of regular languages, and Krystyna Stawikowska another good one on closures of star-free languages.

Kai Salomaa gave a good presentation on sets and hypersets of trajectories, Paul Amblard an excellent one on the history of computing (*automaton* first appears in *Ilias*), especially on *William Stanley Jevons* (1835-1882) and his mechanical machine.

Pál Dömösi, having dedicated his talk to the memory of Alexandru Mateescu who had passed away on January 23, 2005, gave a nice and interesting presentation on products of primitive words, Masami Ito another good one on commutative closure of languages, and Michal Kunc offered an interesting talk on largest solutions of left-linear language inequalities.

Good and interesting presentations were given by Alexander Okhotin on a Boolean grammar for a programming language, by Tatjana Petković on quasi orders, languages and algebras, and by Bianca Truthe on picture languages.

The Friday morning sessions were entirely contributed by *Turku*, except for the first chairman Antonio Restivo who also advertised DLT'05.

AFL'05 was closed on Friday late afternoon by Zoltán Ésik, announcing high temperatures for the following days, thanking all contributors, participants and organizers, announcing a special issue of *TCS* and *Acta Cybernetica*, and also AFL'08 which will be organized by Erzsébet Csuhaj-Várju and take place at MTA/SZTAKI in Budapest.

The proceedings, edited by Zoltán Ésik, and Zoltán Fülöp, containing all invited talks, although that by Manfred Droste as short, and that of Olivier Carton as extended abstract, have been published by *Institute of Informatics, University of Szeged*, and been printed by *Novadat, Győr*.

Because of bad weather (fog with visibility under 50 m, rain, and temperatures below 15°) the excursion (two hiking tours, one of 15, the other one of 5 km), planned for Wednesday afternoon, was first postponed to Thursday, and eventually cancelled entirely. The social program started on Tuesday evening with a welcome party. Fruits, salads, and cakes were offered, as well as mineral water, juice, beer and wine (white *Tokaji Hárslevelű 2003, Nyárlörinci Cserszegi Fűszeres 2002*, rosé *Kékfrankos Rosé 2001*, and red *Egri Bikavér 2001*) as well as coffee. It was well after 22h when this party ended. The second event was the conference dinner on Wednesday evening, opened with champagne.

Thus AFL'05 was a successful conference on a high scientific level, in a nice atmosphere (except for the weather), and well organized.

Report on WSA 2005

Workshop on Semigroups and Automata
July 16, 2005, Lisbon, Portugal

Manfred Kudlek

WSA'05 was held as a joint satellite workshop of ICALP 2005 and CSL 2005 (*International Conference on Semigroups and Languages in Honour of the 65 th birthday of* Donald B. McAlister) in Lisboa on July 16, 2005, exactly one year after another workshop with identical name. Conference site was *Instituto Superior Técnico* (IST). The only entrance of this institute on the weekend was rather difficult to find, and some participants had problems.

WSA'05 was organized by Luís Caires, António Ravara, Vasco Vasconcelos, and Vítor Hugo Fernandes, Gracinda M. S. Gomes, Jean-Éric Pin, Mikhail V. Volkov. The workshop was supported by the AuthoMathA project of the *European Science Foundation (ESF)*. WSA'05 was attended by 43 participants from 11 countries, in details : CA 2, CN 1, CZ 1, DE 6, ES 1, FI 4, FR 2, PT 16, RU 3, UK 5, US 2. The scientific program consisted of an invited lecture and 12 contributions. The program can be found at `http://icalp05.di.fct.unl.pt/`.

Jarkko Kari, with *'Synchronization Problems in Eulerian Graphs'*, presented an excellent invited lecture on synchronization problems in Euler graphs, the Černy and road colouring conjectures, and on lengths of shortest synchronizing words. Interesting and good talks gave Philip Feinsilver on quaternionic and Lie algebras (*'Let us see what fun we had'*), Attila Egri-Nagy on decomposition of rings, reminding *'Remember ! If you have finite state automata then we can tell you how to understand them exactly'* with an unreadable footnote, and Libor Polák, as at AFL'05 on automata theory in practice. Jean-Éric Pin introduced ESF and AutoMathA, the project sponsoring the workshop.

The proceedings, edited by the last 4 organizers, have been published as a report of the *European Science Foundation*. Except for the invited talk they contain all contributions, and in addition another one by Tamara Shcherbak on *'Interval Rank of Monotonic Automata'*.

WSA'05 was an interesting and well organized workshop.

Report on

Natural Processes and Models of Computation
June 16–18, 2005, Bologna, Italy

Elena Calude

The Workshop “Natural Processes and Models of Computation", organised by Rossella Lupacchini, Giorgio Sandri and Guglielmo Tamburrini, and the Department of Philosophy of the University of Bologna was held in beautiful Bolgna from June 16 to 18, 2005.

A city of north-central Italy at the foot of the Apennines, Bologna was originally an Etruscan town and became a Roman colony in the second century B.C. Its famed university, the Europe’s oldest, was founded in 1088. The “Scuola Superiore de Studi Umanistici” of Umberto Eco hosted the workshop. The original fresco in the main lecture room represents the coronation of Charles V (1500-1558) at Bologna Basilica di San Petronio (1530) as Holy Roman emperor, the last imperial crowning by a pope.

The multidisciplinary workshop brought together people working in computer science, physics, philosophy, and the cognitive neurosciences with the aim of exploring and comparing computational paradigms inspired by nature.

We now give short description of the contributions presented to the workshop. In the first session J. Gruska gave a comprehensive overview of frontiers, bounds, driving forces and paradigms of information processing by nature. The talk was followed by M. Frixione (computationalism through levels), D. Mundici (an inspiring talk about mathematics, classical physics and computation), and C. Toffalori (the cost of computing). The second session included talks by C. Calude (computing at the speed of light), M. Rasetti (topological quantum computation), R. Giuntini (a very interesting completeness theorem in quantum computation), and A. Marzuoli (analog and discrete computing machines from quantum angular momenta). The third session started with W. Sieg's interesting proposal for an axiomatic treatment of computability bounds, D. Silva Graça's overview of theories of analog computation, and continued with A. Treves (frontal latching networks) and G. Trautteur (analog computation and digital virtuality). The last session included B. MacLennnan's overview of natural computation, E. Burattini (autonomous robotic systems) and G. Tamburrini (machine experiments and the theoretical modelling of adaptive behaviours).

The workshop ended with a panel discussion on the future of natural computing moderated by C. Calude. The discussion started with a record of David Hilbert's speech *Naturerkennen und Logik* (given on 8 September 1930 in Königsberg) which ended with the famous words (in English translation):

> For us there is no *ignorabimus*, and in my opinion none whatever in natural science. In opposition to the foolish *ignorabimus* I offer our answer: **We must know, we will know.**

A central part of the discussion was based around the presentation by G. Rozenberg of a comprehensive "programme" for the future of natural computing. The main points of discussion were: What is natural computing? Is computing the new science for this century? Is mathematics becoming more experimental? Limits: Church-Turing thesis, van Leeuwen-Wiedermann extended thesis, Sieg's axioms, Is super-Turing computing a myth?

A selection of the best papers presented at the workshop will appear in a special issue of the *Journal of Natural Computing*.

More workshop pictures (including snapshots of the fresco appearing on the walls of the lecture room) and drawings by Marcello Frixione are posted at

```
www.massey.ac.nz/~ecalude/BolognaImages/DigitalBologna
www.massey.ac.nz/~ecalude/BolognaImages/ArtistBologna
```

The workshop was very well organised, stimulating, and inspiring: a most enjoyable experience.

Pictures from DNA 11 (by N. Santean)

James Gimzewski

Eric Klavins

Eshel Ben-Jacob

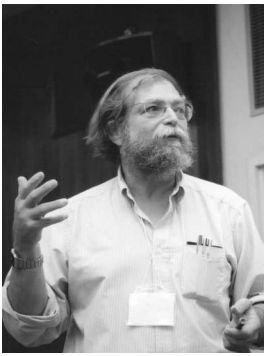
Mark Daley

Dipankar Sen

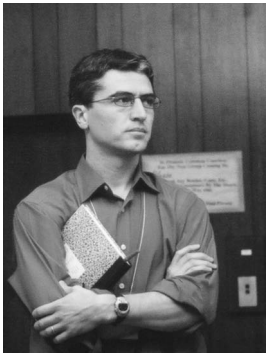
Erik Winfree

Perh Harbury

Junghuei Chen



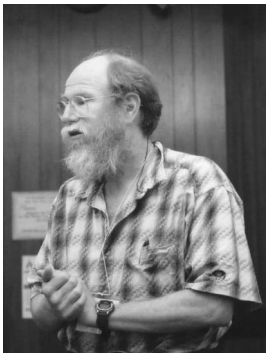
Ned Seeman



Niles Pierce

Hirotaka Nakagawa

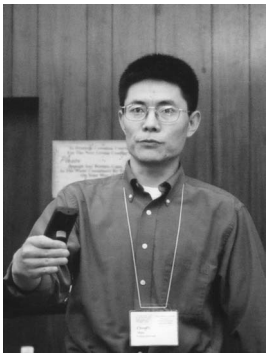
Yasubumi Sakakibara



John Reif

Oscar Ibarra

Robert Barish



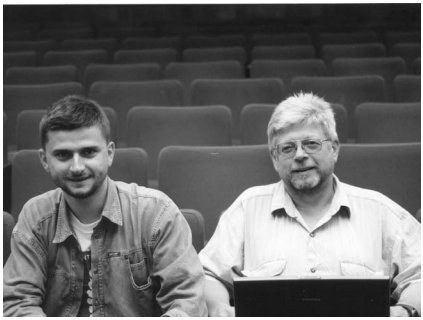
Chengde Mao

Byoung-Tak Zhang

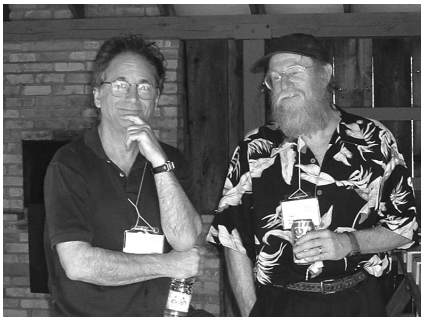
Masami Hagiya

G. Rozenberg, L. Adleman, and L. Kari

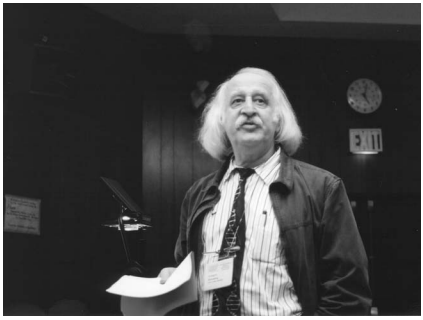
Elizabeth Goode at the Poster Session



Vladimir and Yurii Rogozin



L. Adleman and J. Reif



Grzegorz Rozenberg

E. Klavins and C. Zandron
at the Iroquois Village

more pictures at http://www.csd.uwo.ca/~nic/dna11

Abstracts of PhD Theses

Abstract of PhD Thesis

Author:	Violetta Lonati
Title:	Pattern statistics in rational models
Language:	English
Supervisor:	Alberto Bertoni and Massimiliano Goldwurm
Institute:	Università degli Studi di Milano, Italy
Date:	7 March 2005

Abstract

The thesis focuses on the frequency of occurrences of a repeated pattern in a random sequence of letters. If we assume to know the probabilistic model (and its parameters) that generates the text, the central question is: *how many occurrences of a given pattern shall we expect in such a random sequence?* Below, we shall refer to this question as the *frequency problem*.

Among the early motivations for the study of this problem, one should mention code synchronization and approximated pattern matching. However, possible applications are in molecular biology. Nowadays, biologists have large sets of DNA sequences from many different organisms and they need quantitative tools and statistical methods to help them in analyzing sequences. Identifying words that show relevant deviations between their observed frequency and their frequency predicted by a given model could be a useful way to extract information from DNA sequences.

The frequency problem can be studied under different assumptions concerning the source that generates the text, or the pattern to search for through the text. In this work we study pattern occurrences in a new framework, introducing a stochastic model defined via rational formal series in non-commuting variables a, b and non-negative coefficients (or, equivalently, by weighted automata). More precisely, for any integer n we define a probability space of all words in $\{a, b\}^n$ such that the probability associated with any word is proportional to its coefficient in the series. The *rational symbol frequency* problem is then taken in exam: intuitively this concerns the study of the sequence of random variables $\{Y_n\}_n$ representing the number of occurrences of the symbol a in words of length n chosen at random in $\{a, b\}^*$, according to the probability distribution given by the rational model.

The thesis shows how the rational model can be viewed as a proper extension of the Markovian model usually considered in the literature. Indeed, the question

of studying the number of occurrences of a regular pattern in a text generated by a Markovian source can always be translated into the rational symbol frequency problem for a suitable rational series over two non-commuting variables, while the converse does not hold in general.

The main contributions included in the thesis are estimating the moments of the random variable Y_n and determining local and central limit distributions of the sequence $\{Y_n\}_n$ as n tends to infinity.

First, the transition matrix associated with the series defining the model is assumed to be primitive. In this case the mean and the variance of Y_n turn out to be asymptotically linear and precise expressions for the constants appearing in their asymptotic formulas are given. Moreover, a central limit theorem holds and a condition is provided that guarantees the existence of a Gaussian local limit theorem; to state this condition, a notion of symbol periodicity for weighted automata is introduced, in analogy with the classical periodicity theory of Perron–Frobenius for non-negative matrices. As an application of the previous analysis, one obtains an asymptotic estimation of the growth of the coefficients for a subclass of rational formal series in two commuting variables.

These results are then extended dropping the primitive hypothesis usually assumed in the literature. First, bicomponent models (defined by weighted automaton with two strongly connected components) are studied, obtaining in many cases limit distributions quite different from the Gaussian one. Finally, to deal with arbitrary non-primitive models, a general approach is presented. It is based on the decomposition of the weighted automaton defining the model into strongly connected components, in order to detect the elements that mainly determine the limit distribution. In the most relevant cases the limit distribution is explicitely determined: it is characterized by a unimodal density function defined by polynomials over adjacent intervals.

Author's address Violetta Lonati
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Comelico, 39
20135 Milano - Italy
Email: lonati@dsi.unimi.it
URL: http://homes.dsi.unimi.it/~lonati

Abstract of PhD Thesis

Author:	Saeed Salehi
Title:	Varieties of Tree Languages
Language:	English
Supervisor:	Magnus Steinby
	Zoltán Ésik (Reviewer)
	Wolfgang Thomas (Reviewer)
	Thomas Wilke (Opponent)
Institute:	University of Turku, and
	Turku Centre for Computer Science
Date:	12 August 2005

Abstract

Trees are among the most fundamental and ubiquitous structures in mathematics. Tree languages and automata on trees have been studied extensively since the 1960s from both a purely mathematical and application point of view. When trees are defined as terms, universal algebra becomes directly applicable to tree automata and, on the other hand, the theory of tree automata suggests new notions and problems to universal algebra.

Different syntactic invariants have been proposed as bases for classifications of regular tree languages: syntactic algebras (Steinby 1979, 1992; Almeida 1990), syntactic monoids and syntactic semigroups (Thomas 1983; Nivat and Podelski 1989), tree algebras (Wilke 1996) and syntactic theories (Ésik 1999). However, so far variety theorems comparable with Eilenberg's classical theorems for regular string languages were known for syntactic algebras and syntactic theories only. In this thesis we consider several aspects of varieties of tree languages and settle some open questions concerning the various formalisms.

In Chapter 2 we extend the variety theorem for general recognizable subsets of free algebras (Steinby 1979) to the many-sorted case. In Chapter 3 we formulate Pin's (1996) theory of positive varieties for tree languages and prove a variety theorem that establishes a correspondence between positive varieties of tree languages and varieties of finite ordered algebras.

It has been known already for quite a long time that not all varieties of tree languages can be defined by syntactic monoids or syntactic semigroups, and the question about the exact defining power of these syntactic invariants has been raised by several authors. In Chapter 4 we answer this question by characterizing

the varieties of tree languages that correspond to some variety of finite monoids or semigroups. In Chapter 5 we characterize families of tree languages definable by ordered monoids and study some special instances of the above mentioned variety theorems.

Chapter 6 is devoted to Wilke's tree algebras. We introduce a convergent term rewriting system that yields an efficient method to decide the word problem of tree algebras. As a special case of the many-sorted theory developed in Chapter 2 we obtain a variety theorem for families of tree languages defined by tree algebras. Moreover, we prove that, for any sufficiently rich alphabet, all congruence-preserving functions of the tree term algebra are obtained as compositions of the basic tree-constructing operations.

(A Brief) Table of Contents

Author's address Saeed Salehi
Turku Centre for Computer Science
DataCity, Lemminkaisenk. 14 A
FI – 20520 TURKU
Finland
email: saeed@cs.utu.fi
URL: http://staff.cs.utu.fi/staff/saeed/pht.html

Abstract of PhD Thesis

Author:	Jan Strejček
Title:	Linear Temporal Logic: Expressiveness and Model Checking
Language:	English
Supervisor:	Mojmír Křetínský
Institute:	Masaryk University in Brno, Czech Republic
Date:	25 February 2005

Abstract

Model checking of finite-state systems with specifications given as formulae of Linear Temporal Logic (LTL) is one of the most common verification problems. Like other verification problems, LTL model checking suffers from state explosion. Techniques tackling state explosion usually employ some specific property of the LTL fragment they are designed for. For example, a popular method called partial order reduction is based on the fact that specifications given by LTL formulae without the 'next' operator do not distinguish between the stutter equivalent behaviours of a system.

We study the properties of LTL fragments that are related to model checking. In particular, we are interested in the properties that can potentially lead to new techniques suppressing the state explosion problem. At the same time we study expressiveness and decidability of LTL fragments, and complexity of model checking problem for the fragments.

Besides a broad unifying overview of hitherto known results, this thesis presents some original results about LTL fragments with temporal operators 'next' and 'until', where the nesting depths of one or both operators are bounded. More precisely, we extend the above-mentioned stuttering principle to these fragments and describe a new concept of characteristic patterns. In both cases we indicate that our results can improve existing model checking techniques. Furthermore, we develop the established fact that LTL is expressively equivalent to alternating 1-weak Büchi automata (A1W automata). Specifically, we identify the classes of A1W automata that are expressively equivalent to LTL fragments of the until-release hierarchy and LTL fragments using only future temporal operators with or without bounded nesting depths.

This thesis also contains a collection of open questions and topics for future work.

(Abridged) Table of Contents

Author's correspondence address

Jan Strejček
Faculty of Informatics, Masaryk University
Botanická 68a
60200 - Brno
Czech Republic
email: strejcek@fi.muni.cz
url: http://www.fi.muni.cz/~xstrejc

Abstract of PhD Thesis

Author:	Stijn Vansummeren
Title:	Well-Definedness, Semantic Type-Checking, and Type Inference for Database Query Languages
Language:	English
Supervisor:	Jan Van den Bussche
Institute:	University of Hasselt, Belgium
Date:	20 May 2005

Abstract

The operations of a general-purpose programming language such as C or Java are only defined on certain kinds of inputs. For example, if a is an array, then the array indexation $a[i]$ is only defined if i lies within the boundaries of the array. If, during the execution of a program, an operation is supplied with the wrong kind of input, then the output of the program is undefined. Indeed, the program may exit with a runtime error or, worse yet, it may compute the wrong output.

To detect such programming errors as early as possible, it is hence natural to ask whether we can solve the *well-definedness problem*: given an expression and an input type, decide whether the semantics of the expression is defined for all inputs adhering to the input type. Unfortunately, this problem is undecidable for any computationally complete programming language, by Rice's Theorem.

Most programming languages therefore provide a *static type system* to detect programming errors. These systems ensure "type safety" in the sense that every expression that passes the type system's tests is guaranteed to be well-defined. Due to the undecidability of the well-definedness problem, these systems are necessarily incomplete, i.e., there are expressions that are well-defined, but do not type-check. Such expressions are problematic from a programmer's point of view, as he must rewrite his code in order to get it to type-check. As such, a major quest in the theory of programming languages consists of finding type systems for which the set of well-defined but ill-typed expressions is as small as possible.

Although the Holy Grail in this quest (i.e., a type system that is both sound and complete) can never be found for general-purpose programming languages, this does not mean it cannot be found for smaller, specific-purpose programming languages. The most prominent examples of the latter are *database query languages* such as SQL, OQL, and XQuery. Expressions in all these languages can be undefined. As query languages do not have full computational power, Rice's

theorem does not apply and it is hence worthwhile to investigate if we can't decide the well-definedness problem for them[1]. If so, then we obtain in essence a type system that is both sound and complete. In this dissertation we therefore study the well-definedness problem for database query languages. We start our study with well-definedness for the Nested Relational Calculus (NRC for short), a well-known query language for the complex object data model. The NRC is an extension of the relational algebra (which serves as the data processing core of SQL) and can itself be viewed as a data processing core of OQL. Furthermore, the NRC inspired the design of various semi-structured languages such as UnQL, StruQL, and Quilt, on which XQuery is based. As such, our study of well-definedness for the NRC serves as a good starting point for the study of well-definedness in SQL, OQL, and XQuery.

Certain features of the latter two languages are not captured by the standard set-based NRC however. Indeed, OQL operates on bags and lists in addition to sets, while XQuery operates on lists. Both languages have object identity and the ability to create new objects. We therefore continue our study by identifying broad classes of first-order, object-creating query languages operating on list-based data for which the well-definedness problem is (un)decidable. Specifically, we identify properties of basic operations in such languages that can make the well-definedness problem undecidable and give corresponding restrictions that are sufficient to ensure decidability. The obtained results can be transfered to a bag-based data model, and are directly applicable to OQL and XQuery.

A problem related to well-definedness is the *semantic type-checking problem*: decide, given an input type, an expression and an output type, whether the expression only produces outputs in the output type on inputs in the input type. This problem is useful in a "producer-consumer" setting where a producer generates data, which is processed by a consumer. In order to ensure good operation by the consumer, the producer is expected to only produce data adhering to a certain type. Unfortunately, the semantic type-checking problem is also undecidable for any computationally complete programming language, by Rice's theorem. In practice however, the producer will often consist of a query against a database. It is therefore interesting to see if we can't solve the semantic type-checking problem for the query languages mentioned above. We study this problem for the NRC. For XQuery and other XML-related languages, the problem has already been studied extensively.

Concretely, our study shows that both well-definedness and semantic type-checking remain undecidable for query languages that are powerful enough to

[1] XQuery is in fact a full-fledged general-purpose programming language. Most XQuery programs are of the restricted form "for-let-where-return" however, which we regard as the true query language part of XQuery.

simulate the relational algebra. It follows that a sound and complete type system for SQL, OQL, or XQuery does not exist (although we identify several useful fragments for which such a system does exist). Query languages that want to verify the absence of certain programming errors statically hence have to do so by means of a traditional, incomplete type system. In the second part of this dissertation we therefore study classical type system problems from the theory of programming languages in the context of database query languages.

We first study the complexity of the *typability problem* for the relational algebra: decide, given a relational algebra expression e, whether there exists an assignment of types to the relation variables in e such that e type-checks in the classical type system of the relational algebra. Checking typability of relational algebra expressions is the analog in the relational algebra of static type-checking in implicitly typed programming languages, such as ML. It is therefore interesting to see what its complexity is. It is known for instance that typability is P-complete for the simply typed lambda calculus and EXPTIME-complete for ML. Van den Bussche and Waller have shown that typability for the relational algebra is in NP. The precise complexity remained open, however. In this dissertation, we show that the problem is NP-hard, even in various restricted settings.

Next, we turn our attention to the NNRC, a named version of the Nested Relational Calculus. The basic operators of the NNRC are polymorphic. We can inspect the A attribute of any record, as long as it has an attribute A. We can take the cartesian product of any two records whose attribute sets are disjoint. We can take the union of any two sets of the same type. Similar typing conditions can be formulated for the other operators of the NNRC. When combining operators into expressions, these typing conditions become more evolved. A natural question thus arises: given an NNRC expression e, under which assignments of free variables in e to types is e well-typed? And what is the resulting output type of e under these assignments? In particular, can we give an explicit description of the typically infinite collection of these *typings*? This is nothing but the NNRC version of the classical *type inference* problem, an extensively studied topic in the theory of programming languages. We propose an explicit description of the set of all possible typings of an NNRC expression e by means of a conjunctive logical formula ϕ_e, which is interpreted in the universe of all possible types. We show that ϕ_e is efficiently computable from e and that the satisfiability problem of such conjunctive formulas belongs to NP. Consequently, typability for the NNRC is also in NP. Since the NNRC is an extension of the relational algebra, for which typability is already NP-complete, this thus shows that typability for the NNRC is not more difficult than for the special case of the relational algebra.

Author's correspondence address: Stijn Vansummeren, University of Hasselt, Department WNI, Agoralaan Gebouw D, B-3590 Diepenbeek, Belgium. stijn.vansummeren@uhasselt.be

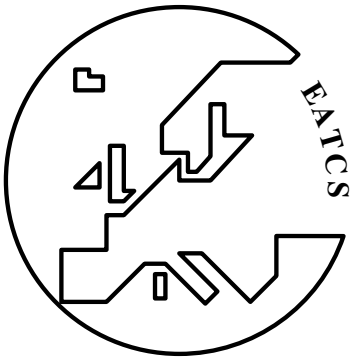
European

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, a Treasurer and a Secretary. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual **I**nternational **C**olloquium on **A**utomata, **L**anguages and **P**rogramming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the "Bulletin of the EATCS;"
- Publication of the "EATCS Monographs" and "EATCS Texts;"
- Publication of the journal "Theoretical Computer Science."

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: TAPSOFT (Conference on Theory and Practice of Software Development), STACS (Symposium on Theoretical Aspects of Computer Science), MFCS (Mathematical Foundations of Computer Science), LICS (Logic in Computer Science), ESA (European Symposium on Algorithms), Conference on Structure in Complexity Theory, SPAA (Symposium on Parallel Algorithms and Architectures), Workshop on Graph Theoretic Concepts in Computer Science, International Conference on Application and Theory of Petri Nets, International Conference on Database Theory, Workshop on Graph Grammars and their Applications in Computer Science.

Benefits offered by EATCS include:

- Subscription to the "Bulletin of the EATCS;"
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from "EATCS Monographs" and "EATCS Texts;"
- Discount (about 70%) per individual annual subscription to "Theoretical Computer Science;"
- Discount (about 70%) per individual annual subscription to "Fundamenta Informaticae."

(1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July.

Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

SITES OF ICALP MEETINGS:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, Great Britain 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, Great Britain 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D.Theses;
- Entertainments and pictures related to computer science.

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at `http://www.eatcs.org/bulletin`. Questions and proposals can be addressed to the Editor by email at `bulletin@eatcs.org`.

(3) EATCS MONOGRAPHS AND TEXTS

This is a series of monographs published by Springer-Verlag and launched during ICALP 1984; more than 50 volumes appears. The series includes monographs in all areas of theoretical computer science, such as the areas considered for ICALPs. Books published in this series present original research or material of interest to the research community and graduate students. Each volume is normally a uniform monograph rather than a compendium of articles. The series also contains high-level presentations of special topics. Nevertheless, as research and teaching usually go hand in hand, these volumes may still be useful as textbooks, too. Texts published in this series are intended mostly for the graduate level. Typically, an undergraduate background in computer science is assumed. However, the background required may vary from topic to topic, and some books may be self-contained. The texts cover both modern and classical areas with an innovative approach that may give them additional value as monographs. Most books in this series will have examples and exercises. Updated information about the series can be obtained from the publisher.

The editors of the series are W. Brauer (Munich), G. Rozenberg (Leiden), and A. Salomaa (Turku). Potential authors should contact one of the editors. The advisory board consists of G. Ausiello (Rome), M. Broy (Munich), C.S. Calude (Auckland), A. Condon (Vancouver), D. Harel (Rehovot), J. Hartmanis (Cornell), T. Henzinger (Lausanne), N. Jones (Copenhagen), T. Leighton (MIT), M. Nivat (Paris), C. Papadimitriou (Athens and San Diego), and D. Scott (Pittsburgh).

EATCS Monographs and Texts is a very important EATCS activity and its success depends largely on our members. If you are a potential author or know one please contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:

Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands

who acknowledges EATCS membership and forwards the order to Springer-Verlag.

(4) THEORETICAL COMPUTER SCIENCE

The journal *Theoretical Computer Science*, founded in 1975, is published by Elsevier Science Publishers, Amsterdam, currently in 20 volumes (40 issues) a year. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

All kinds of papers, introducing or studying mathematical, logical and formal concepts and methods are welcome, provided that their motivation is clearly drawn from the field of computing.

Papers published in *TCS* are grouped in three sections according to their nature. One section, "Algorithms, automata, complexity and games," is devoted to the study of algorithms and their complexity using analytical, combinatorial or probabilistic methods. It includes the fields of abstract complexity (i.e., all the results about the hierarchies that can be defined using Turing machines), of automata and language theory (including automata on infinite words and infinitary languages), of geometrical (graphic) applications and of system performance using statistical models. A subsection is the Mathematical Games Section, which is devoted to the mathematical and computational analysis of games. The second section, "Logic, semantics and theory of programming," is devoted to formal methods to check properties of programs or implement formally described languages; it contains all papers dealing with semantics of sequential and parallel programming languages. All formal methods treating these problems are published in this section, including rewriting techniques, abstract data types, automatic theorem proving, calculi such as SCP or CCS, Petri nets, new logic calculi and developments in categorical methods. The newly introduced third section is devoted to theoretical aspects of "Natural Computing."

The Editors-in-Chief of "Theoretical Computer Science" are:

G. Ausiello, Università di Roma 'La Sapienza', Dip. Inform. e Sistemistica,
via Salaria 113, 00198 Roma, Italy;

D. Sannella, University of Edinburgh, Lab. for Foundations of Computer Science,
Division of Informatics, King's Building, Mayfield Road, Edinburgh, EH9 3JZ, UK

Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands

M.W. Mislove, Tulane University, Dept. of Mathematics, New Orleans, LA 70118, USA.

ADDITIONAL INFORMATION

For further information please visit `http://www.eatcs.org`, or contact the Secretary of EATCS:

Prof. Dr. Branislav Rovan, Department of Computer Science, Comenius University, SK-84248 Bratislava, Slovakia, Email: `secretary@eatcs.org`

EATCS MEMBERSHIP

DUES

The dues are €30 for a period of one year. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for €25 per year. Additional €25 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website `www.eatcs.org`, where you will find an online registration form and the possibility of secure online payment. Alternatively, a subscription form can be downloaded from `www.eatcs.org` to be filled and sent together with the annual dues (or a multiple thereof, if membership for multiple years is required) to the **Treasurer** of EATCS:

Prof. Dr. Dirk Janssens, University of Antwerp, Dept. of Math. and Computer Science
Middelheimlaan 1, B-2020 Antwerpen, Belgium
Email: `treasurer@eatcs.org`, *Tel: +32 3 2653904, Fax: +32 3 2653777*

The dues can be paid (in order of preference) by VISA or EUROCARD/MASTERCARD credit card, by cheques, or convertible currency cash. Transfers of larger amounts may be made via the following bank account. Please, add €5 per transfer to cover bank charges, and send the necessary information (reason for the payment, name and address) to the treasurer.

Fortis Bank, Bist 156, B-2610 Wilrijk, Belgium
Account number: 220–0596350–30–01130
IBAN code: BE 15 2200 5963 5030, SWIFT code: GEBABE BB 18A