COUNCIL OF THE
EUROPEAN ASSOCIATION FOR
THEORETICAL COMPUTER SCIENCE

PRESIDENT: GIORGIO Ausiello ITALY
VICE PRESIDENTS: Burkhard Monien GERMANY
Don Sannella UNITED KINGDOM
Paul Spirakis GREECE
TREASURER: Dirk Janssens BELGIUM
BULLETIN EDITOR: Maria Serna SPAIN

Luca Aceto ICELAND Madhavan Mukund INDIA
Krzysztof Apt The Netherlands Mogens Nielsen DENMARK
Pierre-Louis Curien France Catuscia Palamidessi FRANCE
Josep Diaz Spain David Peleg ISRAEL
Zoltán Ésk Hungary Vladimir Sassone UNITED KINGDOM
Fedor Fomin Norway Jiří Sgall ISRAEL
Giuseppe F. Italiano Italy Andrzej Tarlecki POLAND
Juha Karihumäki Finland Wolfgang Thomas GERMANY
Richard E. Ladner USA Ingo Wegener GERMANY
Jan van Leeuwen The Netherlands Emo Welzl SWITZERLAND
Eugenio Moggi Italy Gerhard Woeginger THE NETHERLANDS

PAST PRESIDENTS:
EATCS COUNCIL MEMBERS

EMAIL ADDRESSES

Luca Aceto ........................................... luca@ru.is
Giorgio Ausiello ................................. ausiello@dis.uniroma1.it
Krzysztof Apt ..................................... apt@cwil.nl
Pierre-Louis Curien ............................. curien@pps.jussieu.fr
Josep Díaz .......................................... diaz@lsi.upc.es
Zoltán Ésik ......................................... ze@inf.u-szeged.hu
Fedor Fomin ........................................... fomin@ii.uib.no
Giuseppe F. Italiano .............................. italiano@disp.uniroma2.it
Dirk Janssens .............................. Dirk.Janssens@ua.ac.be
Juhani Karhumäki ................................. karhumak@cs.utu.fi
Richard E. Ladner ............................... ladner@cs.washington.edu
Jan van Leeuwen ................................... jan@cs.utu.fi
Eugenio Moggi ...................................... moggi@disi.unige.it
Burkhard Monien ................................... bm@upb.de
Madhavan Mukund ................................. madhavan@cmi.ac.in
Mogens Nielsen ................................. mn@brics.dk
Catuscia Palamidessi .......................... catuscia@lix.polytechnique.fr
David Peleg ........................................ peleg@wisdom.weizmann.ac.il
Don Sannella ........................................... dts@dcs.ed.ac.uk
Vladimiro Sassone .............................. vs@ecs.soton.ac.uk
Jiří Sgall ............................................. sgall@math.cas.cz
Maria Serna ........................................ mjserna@lsi.upc.edu
Paul Spirakis ....................................... spirakis@cti.gr
Andrzej Tarlecki ................................. tarlecki@mimuw.edu.pl
Wolfgang Thomas ................................. thomas@informatik.rwth-aachen.de
Ingo Wegener ................................. ingo.wegener@uni-dortmund.de
Emo Welzl ........................................... emo@inf.ethz.ch
Gerhard Wöeginger ...................... g.j.woeginger@math.utwente.nl
All contributions are to be sent electronically to bulletin@eatcs.org

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

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

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

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

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

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

The EATCS home page is http://www.eatcs.org
# Table of Contents

## EATCS Matters

- Letter from the President .................................................. 3
- Letter from the Editor .......................................................... 6

## Institutional Sponsors

## EATCS News

- The Japanese Chapter, by K. Makino ........................................ 11
- News from India, by M. Mukund ............................................ 14
- News from New Zealand, by C.S. Calude ................................. 16

## The EATCS Columns

- The Algorithmics Column, by G. Woeginger
  - Algorithms Aspects of the Consecutive-Ones Property, by M. Dom .... 27
- The Algorithmic Game Theory Column, by M. Mavronikolas
  - Recent Advances in Approximability of Bimatrix Nash, by
    S. Kontogiannis, P. Spirakis, and H. Tsaknakis .......................... 60
- The Complexity Column, by J. Torán
  - Integer Multiplication and the Complexity of Binary Decision
    Diagrams, by B. Bollig .................................................. 78
- The Concurrency Column, by L. Aceto
  - Deriving Labelled Transition Systems – A Structural
    Approach, by J. Rathke and P. Sobocinski ............................. 107
- The Distributed Computing Column, by P. Fatourou
  - Transactional Memory: A Primer for Theorists, by M. Herlihy ...... 123
- The Formal Specification Column, by H. Ehrig
  - COSAN DPO Approach: An Alternative for DPO Graph
    Transformations, by H. Ehrig, F. Hermann, and U. Prange ........... 139
- The Logics in Computer Science Column, by Y. Gurevich
  - The Logics of Infons, by Y. Gurevich and I. Neeman .................. 150

## Reports from Conferences

- The British Colloquium for Theoretical Computer Science
  (BCTCS 2009), by A. Czumaj, S. Kalvala, and S. Matthews ............ 181
REPORTS FROM PROJECTS

AUTOMATA: FROM MATHEMATICS TO APPLICATIONS, by J.E. Pin and O. Serre ...................................................... 214

EATCS LEAFLET ................................................................. 225
EATCS Matters
Letter from the President

Dear EATCS members,

First of all let me make an important announcement. The Goedel Prize that is co-sponsored by EATCS and SIGACT, has been assigned this year to the papers "Entropy waves, the zig-zag graph product and new constant degree expanders", by Omer Reingold, Salil Vadhan and Avi Wigderson, Annals of Mathematics, Vol 155, (2002), 157-187, and "Undirected connectivity in Log-Space, by Omer Reingold, Journal of ACM, Vol 55 (4) (2007). These papers have opened the way to new important applications of graph expanders and to the solution of various long standing open problems. This year the Prize will be presented at STOC.

The program of ICALP 2009 is now ready. It is an excellent program, both for the high quality of the contributed papers and for the large number of scientific events, like the presentation of the EATCS Award to Gerard Huet, the special event for Christos Papadimitriou, and the invited talks by Georg Gottlob, Thomas Henzinger, Kurt Mehlhorn, Noam Nisan. The conference, chaired by Susan Albers (Track A), Wolfgang Thomas (Track B) and Alberto Marchetti-Spaccamela and Yossi Matias (co-Chairs of Track C), will be again a great success. The Conference will also host four satellite workshops: ALGOSENSORS 2009 (5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks), DCM 2009 (5th International Workshop on Developments in Computational Models), FOCLASA 2009 (8th International Workshop on Foundations of Coordination
Languages and Software Architectures), QUANTLOG 2009 (Workshop on Quantitative Logics). The General Chairs Paul Spirakis, Elias Koutsoupias, Christos Kaklamanis together with their teams, are doing a very good job. For more information please look at ICALP 2009 web site: http://icalp09.cti.gr/. Hurry up to register! We hope to see most of you in Rhodes.

It is worth noting that the ICALP Proceedings this year will appear in a new 'subline' of LNCS that has been initiated by Springer upon stimulus from EATCS and ETAPS. The name of the series is Advanced Research in Computing and Software Science. The new series will host proceedings of a few high profile conferences in the domain of theoretical computer science and software science, primarily, but not exclusively, proceedings of conferences promoted by EATCS and ETAPS.

Also the organization of ICALP 2010 is progressing. In Rhodes you will have the chance to see the first Call for Papers with the indication of program committees and invited speakers. At ICALP also bids for ICALP 2011 and (exceptionally) for ICALP 2012 will be presented. Come to the General Assembly if you want to know more.

Before concluding let me also inform you about a very important project that EATCS will launch this year in Rhodes. Based on the preliminary discussion we had last year the EATCS Council has finally approved the creation of a new prize for young researchers. The award is named after Mojzesz Presburger who accomplished his work on decidability of the theory of addition (which today is called Presburger arithmetic) as a student in 1930.
Nominations for the Presburger Award can be submitted by any member of the scientific community except the nominee and his/her advisors for the master thesis and the doctoral dissertation. Nominated scientists should be at most 35 years at the time of the deadline of nomination. The Call for the 2010 Presburger Award will be issued in the Fall 2009. Sincere greetings to all of you.

Giorgio Ausiello, Rome
June 2009
Dear Reader,

Welcome to the June 2009 issue of the Bulletin of the EATCS. Our columns deliver the usual richness and variety of interesting contents. We start with a survey on algorithmics aspects of the consecutive-ones property for binary matrices ("Algorithmics Column"), and proceed through an analysis of the state-of-the-art in approximability of bimatrix Nash equilibria ("Algorithmic Game Theory Column"), followed by a survey of results on the complexity of restricted binary decision diagrams for integer multiplication ("Computational Complexity Column"). An introduction to the recent directions in deriving labelled transition systems ("Concurrency Column"), an overview on Transactional Memory ("Distributed Computing Column"), an excursion on the cospan DPO approach for graph transformations ("Formal Specification Column"), and an overview on the logic of Infons ("Logic in Computer Science Column").

Let me draw your attention to the the report on the main ideas discussed at BCTCS 2009, the 25th British Colloquium on Theoretical Computer Science and to the report on the ESF project “AutoMathA,” project.

I trust that you will find a lot of interesting material to read in this issue. I want to express my gratitude to all the contributors and encourage you to do so in the near future.

Maria Serna, Barcelona

June 2009
INSTITUTIONAL SPONSORS
BRICS, Basic Research in Computer Science
Aarhus, Denmark

Microsoft Research
Cambridge, United Kingdom

Elsevier
Amsterdam, The Netherlands

Springer-Verlag
Heidelberg, Germany

CTI, Computer Technology Institute
Greece
REPORT FROM THE JAPANESE CHAPTER

K. Makino (Univ. of Tokyo)

7th EATCS-JP/LA Presentation Award
The seventh EATCS/LA Workshop on Theoretical Computer Science was held at Research Institute of Mathematical Sciences, Kyoto Univ., Feb. 2 ~ 4, 2009. Prof. Ryuhei Uehara (Japan Advanced Institute of Science and Technology (JAIST)), who presented the following paper, was selected as the 7th EATCS/LA Presentation Award.

Complexity of Pleats Folding
by Tsuyoshi Ito, Masashi Kiyomi, Shinji Imahori, and Ryuhei Uehara

The award will be given to him at the Summer LA Symposium in July 2009. Congratulations! Please check our web page for the detail information and the list of presented papers.

Report on the “CompView” GCOE Project
by Osamu Watanabe (Tokyo Inst. of Tech.)

I have started the CompView GCOE project from 2006 summer. As a leader and as a member of the EATCS who has been royal to Theoret. Comput. Sci., it is my great pleasure to explain our project here in this Bulletin.

1. Explanation on the Global Center of Excellence Program (GCOE) in Japan
I should start with explaining the GCOE program itself. This is a program run by Japanese government to support departments of selected universities for establishing a center of excellence on their target research. Here is a brief summary of this program.

Funding Agency: Japan’s Ministry of Education and Technology (MEXT)
Purpose: Strengthen and enhance the education and research functions of Japanese graduate schools, to foster highly creative young researchers, especially, doctors in their fields.
Method:
1. For each of 4 fields (chosen each year in turn), universities are invited to propose their GCOE projects for the field;
2. From such project applications, about 12 univ.s are selected; and
3. Provide five year funding support to those selected univ.s to establish education & research center.
Budget Size: Approximately 1.5 to 3 million dollars per year, which would be 10 four-year term prof. and/or post doc. positions and for hiring approximately 40 doctoral students as Research Assistants.

2. Explanation on our CompView GCOE Project

For the GCOE program, I proposed to establish a research/educational center for studying "computation" as a fundamental methodology to all sorts of sciences. This idea has been accepted by many researchers of our university from various fields such as pure math., statistics, operations research, and computer science (from theory to programming languages and systems, e.g., supercomputer architecture). Thus, we made up a team consisting of faculty members of several related departments and proposed our CompView project as described in a box below.

<table>
<thead>
<tr>
<th>CompView: A New Framework for the Sciences</th>
</tr>
</thead>
</table>
| Recently, importance of computation-centric methodology has been pointed out by several researchers. Most phenomena we can observe and analyze in various scientific fields, e.g., in physics, in biology, or even in social science, can be regarded as computational processes. In fact, these phenomena have been often analyzed by computer simulations. The computation-centric methodology (or CompView approach) is to investigate such computational processes by using techniques developed in computer science and applied mathematics. This COE aims to push the boundaries of the CompView methodology, to the point where it could become a viable new framework for scientific modeling and analysis. We strive to demonstrate the effectiveness of the methodology by solving some real-world “hard” problems in various scientific domains, implementing the methodologies on our TSUBAME supercomputer (*).

We also put our efforts for educating doctoral students and postdoctoral researchers to incubate young experts with high-level skills in the CompView approach, i.e., those who are trained in the theory and practice of our methodology. These experts will become general scientists of a new era who could apply their talents to actual problems at hand by working with domain-specific scientists to solve real-world, challenging scientific problems by using advanced mathematics and HPC.

* Our univ. hosts TSUBAME supercomputer that have received worldwide attention by making various innovative achievements.

It is not so easy to run a big project consisting of many researchers from various fields. So we decided to spend first two years for establishing our educational
program for doctoral students. We all have in a sense been working (whether or not being conscious of it) for developing/improving CompView methods of some sort. Thus, the first thing we tried was to educate what we have obtained to doctoral students with various backgrounds. For example, one of my students studying average-case analysis of SAT and related problems learned a way to write a code on GPU (Graphical Processing Unit), implemented a parallel string matching algorithm on GPU, and won a prize at some GPU programming contest. I think that this is not so bad idea for them to obtain additional talent besides what they are studying for their doctoral projects. We also have been encouraging students to visit some other research groups in the project; searching for new applications of their new findings or searching for new techniques for attacking their problems. It would be very good for them to sell ;-) what they have been working on to researchers of different areas. Even more importantly, through such educational collaborations, research collaborations among our project members have been also stimulated. And now we focus on three target topics and we are planning to obtain some results that would lead to some new CompView methods established in our project.

3. We Welcome You to the CompView GCOE Project!

We welcome you to join our project. Please recommend some good doctoral student candidates to us; it would be also possible to send some of your good students to join our research program, or it would be very nice if you can accept some of our students to work with you. In fact, from the very beginning, we asked a TCS research group of ETH Zürich (Prof. E. Welzl) to collaborate us for this project, and we have been exchanging young researchers/doctoral students. It would be nice if we can extend similar collaborations to the other active research groups in Europe and in the world. So if you got interested in this project, please visit our web page compview.titech.ac.jp and feel free to contact us.

Also if possible, please try to initiate similar projects with your colleagues. I sincerely hope that this sort of activity will bring us with interesting and important research opportunities.

---

**The Japanese Chapter**

**Chair:** Kazuo Iwama  
**Vice Chair:** Osamu Watanabe  
**Secretary:** Kazuhisa Makino  
**email:** eatcs-jp@is.titech.ac.jp  
**URL:** http://www.misojiro.t.u-tokyo.ac.jp/EATCS-J/index.html
NEWS FROM INDIA

BY

MADHAVAN MUKUND

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

In this edition of News from India, we preview forthcoming meetings and discuss some interesting new developments.

FSTTCS 2009 The 29th edition of FSTTCS, the annual conference of the Indian Association for Research in Computing Science (IARCS) will be held at the Indian Institute of Technology, Kanpur in December, 2009. The conference forms part of the golden jubilee celebrations of IIT Kanpur. The Programme Committee for FSTTCS 2009 is chaired by Ravi Kannan (Microsoft Research, Bangalore) and K. Narayan Kumar (Chennai Mathematical Institute).

The invited speakers for FSTTCS 2009 are Anuj Dawar (Cambridge), Kim G. Larsen (Aalborg), Martin Odersky (Lausanne), R. Ravi (CMU) and Avi Wigderson (Princeton).

The official website of FSTTCS is located at http://www.fsttcs.org.

LIPIcs Following the example set by STACS 2008, the proceedings of FSTTCS 2008 were published in the form of an open electronic archive, hosted on the Dagstuhl Research Online Publication Server (DROPS), with privately printed copies distributed to the participants. This initiative by STACS and FSTTCS has subsequently been formalized as a new conference proceedings series called Leibniz International Proceedings in Informatics (LIPIcs), to be published electronically by Schloss Dagstuhl–Leibniz Center of Informatics. The proceedings of STACS 2008 and STACS 2009 as well as FSTTCS 2008 have been inducted as part of this new series, and the proceedings of FSTTCS 2009 will also be published through this series.

The objective of LIPIcs to publish the proceedings of high-quality conferences in all fields of computer science, not just theory. The publications follow the
The Bulletin of the EATCS

principle of open access—they are available online and free-of-charge. LIPIcs has an editorial board to oversee the selection of conferences to be included in the series. For more details about LIPIcs, visit http://www.dagstuhl.de/en/publications/lipics.

Formal Methods Update 2009 For the past few years, the Indian Association for Research in Computing Science (IARCS) has organized regular “update” meetings in the area of formal methods. This year’s event will be organized by Rajdeep Niyogi at the Indian Institute of Technology, Roorkee during the period July 13–15, 2009.

These meetings are intended as a forum for Indian researchers and students to update themselves on current trends and to explore new research areas. Typically, the meetings are quite informal. Speakers volunteer to present the latest developments in areas related to their interest. The aim of the presentations are to update the participants on the recent developments in an area, preferably not restricted to an individual’s own contributions. The emphasis is usually on theory but presentations on the current state of the practice or recent tools in the area are also encouraged.

The formal methods community in India is now served by the webpage http://fmindia.cmi.ac.in.

ISLA 2010 Since 2006, the logic community in India has been conducting an annual series of conferences, workshops and instructional schools each January. The community is now formally organized as the Association for Logic in India (ALI). In January 2009, ALI organized the Indian Conference on Logic and Applications (ICLA) in Chennai. The general plan is to alternate conferences and instructional schools. With this in mind, ALI’s next meeting will be the Indian School on Logic and Applications (ISLA), to be held in Hyderabad in January, 2010. Vineet Padmanabhan (University of Hyderabad) will chair the organization. Mihir Chakraborty (Calcutta University) and Kamal Lodaya (Institute of Mathematical Sciences) chair the programme committee.

The ALI webpage is at http://ali.cmi.ac.in.

Madhavan Mukund, Chennai Mathematical Institute
Secretary, IARCS (Indian Association for Research in Computing Science)
http://www.cmi.ac.in/~madhavan
News from New Zealand
by
C. S. Calude
Department of Computer Science, University of Auckland
Auckland, New Zealand
cristian@cs.auckland.ac.nz

1 Scientific and Community News

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

344. C.S. Calude and J.P. Lewis. Is there a Universal Image Generator, 01/2009
345. M. Schimpf and K. Svozil. A Glance at Singlet States and Four-Partite Correlations, 02/2009
348. K. Svozil. Proposed Direct Test of Quantum Contextuality, 02/2009
350. C. S. Calude and L. Staiger. A Note on Accelerated Turing Machines, 02/2009
The Bulletin of the EATCS

353. K. Svozil. Cryptography with Quanta and Chocolate Balls, 03/2009
363. A. Nies. Lowness for Computable and Partial Computable Randomness, 04/2009

2 A Dialogue about Formal Methods with Professor Dines Bjørner

Professor Dines Bjørner, www.imm.dtu.dk/~db/, is a well-known computing scientist working on formal methods for the specification, development and verification of software and hardware systems. Professor Bjørner worked on VDM: the Vienna Development Method at IBM and was involved in producing RAISE: the
Rigorous Approach to Industrial Software Engineering formal method and tools support.

He co-founded VDM-Europe, which subsequently became FME: Formal Methods Europe, an organisation that supports conferences, education and related activities and ForTIA: the Formal Techniques Industry Association. Professor Bjørner’s Chef D’oeuvre is a 3-volume book Software Engineering published in 2005–2006 by Springer. He is a knight of the Order of the Dannebrog and a winner of the John von Neumann Medal (1994) and the Ths. Masaryk Gold Medal (1996), has received a Dr.h.c. from the Masaryk University (2004), is a member of the Academia Europaea (1989) and the Russian Academy of Natural Sciences (AB) (2001), and is a Fellow of the IEEE (2004) and ACM (2005).

Cristian Calude: Could we start by telling us the subject of your PhD Thesis?

Dines Bjørner: A ha! Yes, I thought one associated with EATCS would ask that question. The title of the PhD thesis (1969) was: The Theory of Finite State Transductions. I have much regretted the “The”. At most an “A” might have been more appropriate. In effect I had no tutor; no one at my department, for the last 2/3 of my study, had any background in computer science, let alone in automata theory and formal languages.

But a Danish PhD is not the highest academic degree one can obtain in Denmark. A PhD is society’s blue stamp of approval. Institutions hiring PhDs should be assured that PhDs from the Technical University of Denmark can be put to do advanced engineering and applied research work.

There is also, for example, a doctor technices degree. To obtain such a degree one has to have shown more than ability. One has to have also demonstrated that one’s research has influenced engineering in industry. Normally technical universities should not appoint (full) professors unless they have a Dr. techn. degree. Well, I had not at age 38, when I became a full professor, having spent 11 of my professional years, since my MSc. degree, at IBM, accumulated much in the direction of original, “path-breaking” work worthy of a dr. techn. degree. And, it was then customary that a Royally appointed professor not consider submitting a Dr. techn. thesis. Can you imagine what would happen if it was rejected? Would that professor have to step down?

But, I retired in early 2007. At that time I had begun writing my Dr. techn. thesis. It is based on my studies of software development methods, in particular programming methodology. The thesis was submitted in August 2008. The subject of my Dr. techn. thesis Software Engineering – an Unended Quest\(^1\) is a careful review of my three volume Springer book on Software Engineering: Did I achieve what I set out to achieve in my research since my IBM Vienna Laboratory

\(^1\)You may wish to browse that document at http://www.imm.dtu.dk/~{}db/dr.pdf.
The Bulletin of the EATCS

days (1970–1973)? Where do I think I have succeeded, where could contributions be sharpened, which methodology issues had I neglected, etc.?

CC: When did you join the EATCS?

DB: I believe it was in the late 1970s. The EATCS is the right organisation for European computer scientists. I would wish it also covered computing science. (Instead we have a bewildering set of Programming, Software Engineering, etc. associations — each founded by someone who wanted to be “President”.) To me a distinction is necessary: Computer science is the study and knowledge of the things that can exist “inside” computers as well as the science of the special techniques and theories of how computer science is pursued. Computing science is the study and knowledge of how to construct those things that can exist “inside” computers as well as the study and knowledge of how to apply the techniques of computing science to other than software (and hardware).

CC: Please reminiscence about your relation with the Bulletin.

DB: Grzegorz (Rozenberg) asked me, in the late 1970s, or was it in the very early 1980s, to orchestrate a funding drive to cover the EATCS membership for colleagues in Eastern Europe and the (then) USSR. I had travelled somewhat in several of these countries by then. The EATCS lend me their then current membership list and I wrote to “all” members asking them to contribute. Their willingness was most gratifying. And the EATCS was able to provide up towards 200 memberships — I think — and for several years.

Grzegorz — what a man, without outgoing, active, almost “shameless”, people like him where would we all be? — also invited a private informatics R&D center of which I was one of the two initiators, DDC: Dansk Datamatik Center, to become an EATCS sponsor. So DDC was an EATCS sponsor for almost eight years. Later, when I “co-founded” and directed UNU-IIST: the United Nations International Institute for Software Technology, UNU-IIST became an EATCS sponsor — at least in the years I was the director on that. I shamelessly pronounce, very successful research and post-graduate/post-doctoral training centre in Macau.

CC: Do you consider yourself mainly a computer scientist, a computing scientist or a superposition of both?

DB: I consider myself a computing scientist.

CC: What is your view on “software engineering”?

DB: Software engineering is applied computing science. Most software engineering textbooks, in my opinion, fail to cover advances in computing science, such as formal techniques.

CC: What do you mean by “domain”, “domain engineering” and “domain mod-
DB: By domain is understood a universe of discourse, such as physics, literature, musicology, banking, etc. Something for which there is a reasonably delineated set of terms that cover the simple entities, functions, events and behaviours of the domain: (i) entities that represent phenomena of the domain that one can point to (i.e., seen), smelled, touched, heard, tasted, or be measured by physical (incl. chemical) instruments, or that one can conceptualise from these simple entities or concepts thereof — with some subset of observed domain entities defining a domain state concept; (ii) functions that apply to entities and yield entities, i.e., functions that observe properties of entities, parts of entities, or create new entities, etc.; (iii) events that "occur", in some domain state, and, usually, change that state (that is: events are characterised by certain predicates and transition relations over states); and (iv) behaviours — set of sequences of applied functions, i.e., actions, and events. Examples of domains are: (a) the container line industry, (b) the financial service industry, (c) railway systems; (d) "the market" (consumers, retailers, whole salers, producers and the supply chain); (e) health-care; etc.

By domain engineering is understood the analysis and description of a(ny) domain. Many stages and steps are needed to do proper domain engineering: identification of domain stake holders, domain acquisition, domain analysis, domain terminologisation, domain modelling, domain verification, domain validation and domain theory formation. Some of these stages (and/or their embedded steps) are of pragmatic nature, but several (most in terms of time spent on pursuing these stages) are based on computing science — and domain theory formation requires deep insight into computer science.

By domain modelling is understood the specific tasks of creating a domain model in the form of a precisely narrated and — hopefully also — a precisely formalised description of the domain. The descriptions are of the domain, as it is, with no reference to requirements let alone software.

CC: How relevant is theory for software engineering?

DB: "There is nothing so practical, that is, of use in engineering, as a good theory", someone² is quoted to have said. The techniques of software engineering, whether for domains, requirements or software, mostly have a theoretical foundation. Where physics deals with what can be measured and hence favours such mathematics that can express measurable properties, software engineering deals primarily with domains about which we wish to express logic properties. Where classical mathematics focuses on continuity, computer & computing science, the theory-foundations of software engineering, deals with discreteness and hence modern algebra plays a rôle alongside mathematical logic.

² Kurt Lewin (or James Clark Maxwell or Albert Einstein).
A cornerstone of today software engineering, and hence a central focal point of computing science research is ‘formal specification’ and ‘verification’. A number of formal specification languages and related proof systems have evolved over the years. The first, a part from classical predicate calculus, was VDM. Some others are, in more or less chronological order, Z, RSL (RAISE), B, Event B, Casl, CafeOBJ, ASM and Alloy. Using anyone of these the software engineer can express dynamically evolving systems that grow and shrink in number of entities etc., something that classical calculus is not good at. Early notations of automata and formal language theory are steeped, it seems to me, in classical discrete mathematics — where use of the more modern formal specification languages, in addition to rather sophisticated theorem proving systems, is more appropriate for expressing much of today computer science work. The result is twofold: (i) published papers that attempt to formalise some, otherwise fascinating software-relevant notations or techniques, are fraught with errors, and (ii) the creators of these notations and techniques remain ignorant of progress in computing science and software engineering. A derivative, but far more “dangerous” effect, is that many of our candidates remain sceptical of formal techniques since their teachers never took these seriously. This is a strange phenomenon: It is normal to expect that all mathematicians of a mathematics department understand basic tenets of all the branches of mathematics that are taught to students. But not so in the average computer science department. It is a disgrace.

CC: Can you comment one of your most preferred results?

DB: Well, first of all, since I work in programming methodology: the study of the sets of principles whereby the phases, stages and steps of software development can select analysis and synthesis techniques and tools. The results are not simple 18 page papers where some beautiful set of theorems are postulated and proved. The results are bits and pieces, “bricks and mortar”, of methods. Together with (the late) Hans Bekiˇc, Clif Jones and Peter Lucas, I was one of the originators of VDM — whose VDM–SL was the first ISO standardised formal specification language. And I instigated and co-led the work that led to RAISEs RSL. So I consider my contributions here to be among my “preferred results”. But, in the last 20 years my work has been focused, not on the further research into formal specification languages and their proof systems, but in conceiving of the concept of Domain Engineering and by populating that concept with a number of principles and techniques; and in conceiving of how one can “derive”, major formalisable parts of requirement prescriptions in a systematic theory-based manner — those results, I think, will stand and ought to lead to a rather dramatic reappraisal of so-called requirements engineering. I am of the unabashed opinion that core parts of today

---

research into requirements engineering is seriously flawed.

Before software can be developed one must understand its requirements. But before requirement prescriptions can be developed one must understand the domain. Electronics, automotive, aeronautics & aerospace, etc., engineers all have a sound education in physics and a specifically deep one in those areas of physics that are related directly to their engineering field: plasmaphysics, mechanics and aerodynamics & celestial mechanics. So they understand their domain. Not so for today’s software engineers. Oh, well, yes, for those who develop compilers, database management systems and operating systems. But not those, and they are the vast majority, who develop software for container lines, banks, railways, the market, hospitals, etc.

Most computer science and informatics departments, by far the majority, does not offer anyway near a professional degree in software engineering.

My founding and directing UNU-IIST, the UN University’s Intl. Inst. for Softw. Technology, based in Macau, China SAR, I also count as one of my “preferred results”. Go see for yourself: www.iist.unu.edu.

CC: You worked for companies and universities. Can you please compare life at IBM with that at the Technical University of Denmark or JAIST?

DB: IBM, in my years, 1962–1976, was a very good company — also for us development engineers (62–70) and researchers (70–76). But a software development methodology, such as VDM, first conceived at the IBM Vienna Lab. (73–75), should not be developed in a competition and profit-oriented company, whether IBM or other. IBM’s decision not to make room for the R&D of VDM inside IBM was a ‘blessing in disguise’. VDM would never have thrived if only inside IBM. Such software development techniques should primarily be R&D’ed in open, peer-reviewed research centres — and those you found, in the past, in universities. So IBM offered great management, good salaries and exciting projects. Universities offer poor management, low salaries and freedom to pursue what is important. Well, the last seems to be a bit of a problem these days. Today informatics departments have, however, now a serious problem. Due to political pressures, spurred on by industry spokesmen, our informatics departments are caving into teaching “ready-to-wear” subjects, viz. Database design using Oracle SQL-developer version 1.5.2 or Microsoft Project Standard 2007, etc. There is nothing essentially wrong with these products — but they are just instantiated technology versions of more fundamental techniques and hence theories. Much progress in our sciences is getting lost by not being taught due to the far too many “ready-to-wear” courses. Hence we see that these fields (of progress) are less researched — and eventually disappear. I am rather pessimistic about a future of real education, as in German ‘bildung’, in our sciences.
CC: Your website includes links to other personal websites. How did you choose them?

DB: These personal website, www2.imm.dtu.dk/~{}db/homepages/, refer to such people as Robert Stephen Boyer, Luca Cardelli, Gerard J. Holzmann, Kouichi Kishida, Leslie Lamport and John McCarthy (alphabetically listed). I find their personal website worthwhile. Perhaps I like John McCarthy’s the best. I chose them for this: I know them, personally, their integrity and the fascinating material displayed on their personal Web pages: they are not afraid to act politically incorrect.

CC: Please describe the Informatics Section of the Academia Europaea, its history and goals.

DB: You can read about Academia Europaea (AE) at www.acadeuro.org and about its Informatics Section (IS) at www.acadeuro.org/index.php?id=180. (By the way, you, Cris is the editor of that last set of Web pages.) AE/IS comprises a reasonably fair selection of some 80+ European computer scientists with a few overseas ones as well (cf. www.acadeuro.org/fileadmin/user_upload/Informatics/is-memberlist.html). AE/IS has organised, so far, two events, one in Budapest in 2006, cf. www.jaist.ac.jp/~bjorner/ae-is-budapest (by me), and one in Liverpool, 2008 (by you). Suggested by Wolfgang Resisig, we plan to organise an event, together with members of other AE sections (Linguistics, Literary Studies, Musicology, Social Sciences, Mathematics, Physics, Engineering, Earth & Cosmic Sciences and the four Life Sciences sections) on Models & Modelling. It should be possible for the AE/IS members to issue, on request (as has been done) or at their own volition, as will be done, 3–5 page State-of-the-Informatics Sciences manifestos or position papers that could advise national government, international organisations, the EU Commission, for example its newly founded EIT, etc., on crucial issues before these institutions. The field of computer, computing and informatics science is far from enjoying the status, neither among fellow scientists, at university management levels and in relevant national ministries, that I think it ought to enjoy. AE/IS should try to address this problem. But first AE/IS need to delineate in a reasonable fashion the field of informatics. I was the chairman of AE/IS from early 2004 till April 2009.


DB: This JAIST Research Monograph (ISBN 978-4-903092-17-1) binds, under one cover, ten (10) reports that I wrote during my one year sabbatical in 2006 at JAIST. You can all request that monograph from Prof. Kokichi Futatsugi, 1-1, Asahidai, Nomi, Ishikawa 923-1292 Japan. My answers to your questions on
domains etc. give a hint of what is contained in this monograph. The chapters and appendices of this approximately 500 page monograph span from Technology Management via Science of Domain Engineering to Experimental Evidence. You may view the monograph at www.imm.dtu.dk/~db/jaistmono.pdf. It has 76 nice photos from Japan.

CC: What does give you most pleasure in life?
DB: 43 years with my wife.
CC: Many thanks.
THE EATCS COLUMNS
THE ALGORITHMS COLUMN

BY

GERHARD J. WOE GINGER

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

ALGORITHMIC ASPECTS OF THE
CONSECUTIVE-ONES PROPERTY

Michael Dom∗

Abstract

We survey the consecutive-ones property of binary matrices. Herein, a
binary matrix has the consecutive-ones property (C1P) if there is a permuta-
tion of its columns that places the 1s consecutively in every row. We provide
an overview over connections to graph theory, characterizations, recognition
algorithms, and applications such as integer linear programming and solving
Set Cover.

1 Introduction

We start with considering a short example for the occurrence of the C1P in practi-
cal applications. The example has its background in computational biology, where

∗Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, 07743 Jena,
Germany, michael.dom@uni-jena.de.
the construction of physical maps for the human DNA was a central issue in the past years [5, 6, 45, 72, 97]. A physical map is a map that describes the relative order of markers on a chromosome. A chromosome is basically a long sequence of DNA, and a marker is a short DNA sequence that appears only once on the chromosome and, therefore, is of special interest. To create a physical map, the chromosome is cut into shorter pieces, which are duplicated and called clones. Thereafter, one tests for each of the clones which of the markers appears on it. These tests find out whether a marker appears on a clone, but it is not possible to determine the order of the markers on the clone. The result is a binary matrix where every row corresponds to a clone and every column corresponds to a marker. If a marker appears on a clone, then the corresponding entry of the matrix is 1, otherwise it is 0. The crucial observation for finding the correct order of the markers is that if two markers A and B appear on a clone x, but another marker C does not appear on x, then C cannot lie between A and B on the chromosome. Therefore, to figure out the order of the markers on the chromosome, all one has to do is to order the columns of the matrix in such a way that in every row the 1s appear consecutively. In concrete practical applications, however, the biochemical methods always produce errors such that it is often impossible to order the columns in the resulting matrices as described. One way to deal with these errors is to discard a smallest possible number of clones such that the remaining clones lead to a consistent order of the markers. On the level of binary matrices, this approach means that one has to delete a minimum number of rows such that the resulting matrix has the C1P.

The survey is structured as follows. In the remainder of this section, we provide the definitions used throughout the paper and some basic observations and results on the C1P. Section 2 deals with the relation between the C1P and graph classes. In Section 3, we survey recognition algorithms for the C1P. Section 4 describes some cases of problems that are NP-hard in general, but become polynomial-time solvable on instances with the C1P.

1.1 Preliminaries

An \( m \times n \) matrix contains \( m \cdot n \) entries, which are arranged in \( m \) rows and \( n \) columns. The entry in the \( i \)th row and \( j \)th column of a matrix \( M \) is denoted by \( m_{i,j} \); moreover, we usually use \( r_i \) and \( c_j \) to denote the \( i \)th row and the \( j \)th column, respectively, of a matrix. One can also regard a matrix as a set of columns together with an order on this set; the order of the columns is called the column ordering of the matrix. Two matrices \( M \) and \( M' \) are called isomorphic if \( M' \) is a permutation of the rows and columns of \( M \). A matrix \( M' \) is a submatrix of a matrix \( M \) if we can select a subset of the rows and columns of \( M \) in such a way that deleting all but the selected rows and columns results in a matrix that is isomorphic to \( M' \). If one can
find a submatrix $M'$ of $M$ in this way, we say that $M$ contains $M'$ as a submatrix and that $M'$ is induced by the selected rows and columns. A matrix $M$ is $M'$-free if $M'$ is not a submatrix of $M$.

A matrix whose entries are all from $\{0, 1\}$ is called a binary matrix or $0/1$-matrix; a matrix whose entries are all from $\{0, 1, -1\}$ is called a $0/\pm 1$-matrix. Complementing a column or row of a matrix means that all 1-entries in this column or row, respectively, are replaced by 0s and all 0-entries are replaced by 1s.

Every $0/1$-matrix $M$ can be interpreted as a bipartite graph, which is called the representing graph $G_M$ of $M$: For every row and every column of a matrix $M$, there is a vertex in its representing graph $G_M$, and for every 1-entry $m_{i,j}$ in $M$, there is an edge in $G_M$ connecting the vertices corresponding to the $i$th row and the $j$th column of $M$.

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [30, 37, 79]. One dimension is the input size $n$ and the other one a parameter $d$. A problem is called fixed-parameter tractable if it can be solved in $f(d) \cdot n^{O(1)}$ time, where $f$ is a function only depending on $d$. The basic concept for parameterized intractability is $W[1]$-hardness; if a problem is $W[i]$-hard for any $i \geq 1$, it is presumably not fixed-parameter tractable.

1.2 The Consecutive-Ones Property

The consecutive-ones property of binary matrices appears in many practical applications, such as scheduling [8, 53, 54, 68, 93], information retrieval [67], railway optimization [75, 76, 86], and computational biology [3, 5, 6, 18, 45, 72, 97]. Moreover, the C1P has close connections to graph theory (see Section 2) and plays an important role in the area of solving (integer) linear programs [54, 55, 80, 81, 93] (see also Section 4.1). The formal definition of the C1P and some related concepts reads as follows.

**Definition 1.1.** A block of 1s (block of 0s) in a row of a binary matrix $M$ is a maximal set of consecutive 1-entries (0-entries) in this row. A binary matrix has the strong consecutive-ones property (strong C1P) if in every row the 1s appear consecutively, that is, if every row contains at most one block of 1s. A binary matrix has the consecutive-ones property (C1P) if its columns can be permuted in such a way that the resulting matrix has the strong C1P. If an ordering for the columns of a binary matrix yields the strong C1P, it is called a C1-ordering.

See Figure 1 for examples of matrices with and without the C1P. The terms introduced in Definition 1.1 can be defined analogously for 1-entries appearing consecutively in the columns of a matrix instead of the rows: If the rows of a matrix $M$ can be permuted in such a way that in every column the 1s appear consecutively, then $M$ has the C1P for columns.
Figure 1: Example for the C1P: The matrix on the left has the C1P because by permuting its columns (labeled with $c_1$–$c_4$) one can obtain the matrix shown in the middle where the 1s in each row appear consecutively. The matrix on the right, in contrast, does not have the C1P [91].

\[
\begin{array}{cccc}
c_1 & c_2 & c_3 & c_4 \\
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
\end{array}
\begin{array}{cccc}
c_1 & c_2 & c_3 & c_2 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
\end{array}
\begin{array}{cccc}
c_1 & c_2 & c_3 & c_4 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
\end{array}
\]

Figure 2: Example for the Circ1P: The matrix $A$ has the Circ1P because by permuting its columns (labeled with $c_1$–$c_4$) one can obtain the matrix $B$ where in each row the 1s or the 0s appear consecutively. The matrices $C$ and $D$ in contrast, do not have the Circ1P [90, 91].

A property that is very similar to the C1P but less restrictive is called the circular-ones property: Here one imagines the matrix as wrapped around a vertical cylinder and demands that, possibly after some column permutations, in every row the 1s appear consecutively on the cylinder (which implies that the 0s also appear consecutively):

**Definition 1.2.** A binary matrix has the **strong circular-ones property** (**strong Circ1P**) if in every row the 1s appear consecutively or the 0s appear consecutively (or both). A binary matrix has the **circular-ones property** (**Circ1P**) if its columns can be permuted in such a way that the resulting matrix has the strong Circ1P. If an ordering for the columns of a binary matrix yields the strong Circ1P, then it is called a **Circ1-ordering**.

See Figure 2 for an example. When imagining a matrix $M$ as wrapped around a vertical cylinder, it makes no sense to declare one of its columns as the “leftmost” or “rightmost” column. In this setting, therefore, the term **circular column ordering** is used to describe the order of $M$’s columns: The circular column ordering defines for every column $c$ of $M$ a predecessor and a successor, but it does not declare any of $M$’s columns as the “leftmost” or “rightmost” column. If $M$ has the strong Circ1P, then its circular column ordering is called a **Circ1-circular ordering**.
There exist several characterizations for matrices having the C1P (see Sections 2 and 3). Together with the following observation due to Tucker [90], these characterizations can also be used to recognize matrices with the Circ1P.

**Corollary 1.1** (following from [90, Theorem 1]). Let $M$ be an $m \times n$ matrix and let $j$ be an arbitrary integer with $1 \leq j \leq n$. Form the matrix $M'$ from $M$ by complementing all rows with a 1 in the $j$th column of $M$. Then, $M$ has the Circ1P if and only if $M'$ has the C1P.

An example for Corollary 1.1 can be seen in Figure 2: Complementing in one of the matrices $C$ and $D$ all rows with a 1 in column $c_4$ yields the matrix $D$, which does not have the C1P (this can easily be seen by considering the columns $c_1$–$c_3$); hence, the matrices $C$ and $D$ do not have the Circ1P. Corollary 1.1 implies the following conclusion.

**Corollary 1.2.** Let $M$ be a 0/1-matrix and $M'$ be the matrix obtained by inserting a column that contains only 0s to $M$. Then the following statements are equivalent.

1. $M'$ has the Circ1P.
2. $M'$ has the C1P.
3. $M$ has the C1P.

The relation between different C1-orderings and Circ1-orderings for the columns of a matrix is summarized by Hsu and McConnell [63]:

**Theorem 1.1** ([63, Theorems 3.4 and 3.8]). 1. Let $M$ be a matrix having the C1P. Then every C1-ordering for $M$’s columns can be obtained by starting from an arbitrary C1-ordering and applying a sequence of reverse operations, each of them reversing a linear module in the respective column ordering.
2. Let $M$ be a matrix having the Circ1P. Then every Circ1-circular ordering for $M$’s columns can be obtained by starting from an arbitrary Circ1-circular ordering and applying a sequence of reverse operations, each of them reversing a circular module in the respective circular ordering.

Thereby, a **linear module** of a matrix $M$ is a set $C$ of columns such that in every row of $M$ the entries belonging to $C$ have the same value or the entries not belonging to $C$ are all 0 (or both). A **circular module** is a set $C$ of columns such that in every row of $M$ the entries belonging to $C$ have the same value or the entries not belonging to $C$ have the same value (or both). A **reverse operation** takes a set of consecutive columns from the column ordering or the circular column ordering, respectively, of a matrix and puts it into reverse order. Clearly, applying a reverse operation to a linear module (a circular module) of a matrix that has the strong C1P (the strong Circ1P) does not destroy this property. Theorem 1.1, however, strengthens this observation.

One consequence of statement 2 in Theorem 1.1 is the following finding, which states a relation between the Circ1-orderings and the **shifted C1-orderings**
of the columns of a matrix having the C1P. Thereby, the column ordering of a matrix \( M \) is a shifted C1-ordering if the strong C1P can be obtained by repeatedly taking the column that is currently placed at the leftmost position and moving it from there to the rightmost position.

**Lemma 1.1** ([28]). Let \( M \) be an \( m \times n \) matrix that has the C1P and contains at most \( (n + 1)/2 \) 1s per row. Then every Circ1-ordering for \( M \)'s columns is also a shifted C1-ordering.

## 2 Graph Classes and the C1P/Circ1P

Matrices can be represented by graphs and vice versa, and, therefore, the C1P is related to certain properties of graphs.

**Graph classes closely related to the C1P or the Circ1P.** Given a graph \( G \), there are several “natural” ways to map \( G \) to a matrix that represents all information about \( G \). The following definition describes the most common types of such matrices that represent graphs.

**Definition 2.1.** Let \( G = (V, E_G) \) be a graph with \( V = \{v_1, \ldots, v_n\} \) and \( E_G = \{e_1, \ldots, e_m\} \), and let \( H = (V_1, V_2, E_H) \) be a bipartite graph with \( V_1 = \{u_1, \ldots, u_n\} \) and \( V_2 = \{w_1, \ldots, w_n\} \).

1. The adjacency matrix of \( G \) is the symmetric \( n \times n \) binary matrix \( M \) with \( M_{i,j} = 1 \) if and only if \( \{v_i, v_j\} \in E_G \).
2. The augmented adjacency matrix of \( G \) is the matrix obtained from \( G \)'s adjacency matrix by setting the entries of the main diagonal to 1.
3. The edge-vertex incidence matrix of \( G \) is the \( m \times n \) binary matrix \( M \) with \( M_{i,j} = 1 \) if and only if \( v_j \) is an endpoint of \( e_i \). The transpose of the edge-vertex incidence matrix is called the vertex-edge incidence matrix of \( G \).
4. Let \( c_1, \ldots, c_k \) be the maximal cliques of \( G \). The maximal clique matrix (also called vertex-clique incidence matrix) of \( G \) is the \( n \times k \) binary matrix \( M \) with \( M_{i,j} = 1 \) if and only if \( v_i \) belongs to \( c_j \).
5. The half adjacency matrix of \( H \) is the \( n_1 \times n_2 \) binary matrix \( M \) with \( M_{i,j} = 1 \) if and only if \( \{u_i, w_j\} \in E_H \).

Figures 3 and 5 show the matrix types introduced in Definition 2.1. Clearly, a matrix \( M \) is the half adjacency matrix of a bipartite graph \( H \) iff \( H \) is the representing graph of \( M \).

Some elementary graph classes that are directly related to the properties C1P and Circ1P are defined as follows.
The Bulletin of the EATCS

Figure 3: Matrices defined in Definition 2.1. Matrix $A$ is the adjacency matrix of the graph $G_1$, and Matrix $B$ is the augmented adjacency matrix of $G_1$. Matrix $C$ is obtained from $B$ by permuting the rows and columns; the shapes enclosing its $1$-entries illustrate the quasi Circ1P (see Definition 2.3), which will be used in Table 1. (Actually, the matrix $C$ shows not only that $B$ has the quasi Circ1P, but also that $B$ has the Circ1P.) Matrix $D$ is the maximal clique matrix of $G_2$, and Matrix $E$ is the half adjacency matrix of the bipartite graph $H$. Matrix $C$ shows that $G_1$ is a concave-round graph (see Definition 2.2) as well as a circular-arc graph (see Table 1), Matrix $D$ shows that $G_2$ is an interval graph (see Table 1), and Matrix $E$ shows that $H$ is a convex bipartite graph (see Definition 2.2).

Definition 2.2. 1. A graph is convex-round if its adjacency matrix has the Circ1P, and it is concave-round if its augmented adjacency matrix has the Circ1P [7].
2. A graph $G$ is an interval graph if its vertices can be mapped to intervals on the real line such that two vertices are adjacent if and only if their corresponding intervals overlap [9, 52]. If all intervals have the same length, then $G$ is a unit interval graph; if no interval properly contains another interval, then $G$ is a proper interval graph.
3. A graph $G$ is a circular-arc graph if its vertices can be mapped to a set $A$ of arcs on a circle such that two vertices are adjacent if and only if their corresponding arcs overlap. A circular-arc graph $G$ is a Helly circular-arc graph if for every subset $A' \subseteq A$ it holds that $(\forall a_1, a_2 \in A': a_1 \cap a_2 \neq \emptyset) \Rightarrow \bigcap_{a \in A'} a \neq \emptyset$.
4. A bipartite graph is convex if its half adjacency matrix has the C1P, it is biconvex if its half adjacency matrix has the C1P both for rows and for columns, and it is circular convex if its half adjacency matrix has the Circ1P.

See Figures 3 and 4 for illustrations. Interval graphs and circular-arc graphs are known in graph theory for a long time; they are well-studied (alone for the
recognition problem of these graphs there exists a number of results, see [13, 23, 38, 47, 58, 62, 66, 69] for the recognition of interval graphs and [31, 59, 65, 73, 92] for the recognition of circular-arc graphs and have applications in many fields [15, 46]. One reason for the attention that these two graph classes attract is that many problems that are NP-complete on general graphs (for example, Independent Set) are polynomial-time solvable on interval graphs and circular-arc graphs and also on the other graph classes mentioned in Definition 2.2 (see [1, 15, 46]). This important fact carries over to matrices with the $C1P$ or the Circ1P, where many in general NP-hard matrix problems can be solved in polynomial time [22, 32, 70, 78, 93] (see also Section 4 and [75, 76, 86]). We summarize the relationships between the graph classes of Definition 2.2 on the one hand and the C1P or Circ1P occurring in the matrices of Definition 2.1 on the other hand in Table 1. Note that, since proper interval graphs coincide with unit interval graphs [85, 40], there is only one row for both classes. The property “quasi Circ1P” occurring in the table is defined as follows.

**Definition 2.3** ([90]). A symmetric matrix has the quasi Circ1P if (possibly after permuting the rows and columns without destroying the symmetry) for every 1-entry $m_{ij}$ it holds that $m_{ij} = m_{i,(i+1) \mod n} = m_{i,(i+2) \mod n} = \ldots = m_{i,(i-1) \mod n} = m_{i,j} = 1$ or that $m_{ij} = m_{(j+1) \mod n,j} = m_{(j+2) \mod n,j} = \ldots = m_{((j-1) \mod n)j} = m_{i,j} = 1$.

Thereby, $i \mod j$ is defined as $i \mod j = \begin{cases} i \mod j & \text{if } i \mod j > 0 \\ j & \text{if } i \mod j = 0 \end{cases}$ with $i \mod j$ denoting the remainder of the division $i$ by $j$. The quasi Circ1P is illustrated in Figure 3. The Circ1P always implies the quasi Circ1P [90].

Table 1 does not contain graphs whose vertex-edge incidence matrix or edge-vertex-edge incidence matrix has the C1P; however, these graphs also have a very special structure.

**Definition 2.4.** A caterpillar is a tree in which every non-leaf vertex has at most two non-leaf neighbors.
Table 1: Relationship between graph classes and matrix properties. The symbol “⇒” expresses that the membership in a graph class implies the matrix property for the matrix associated with the corresponding graph; the symbols “⇐” and “⇔” denote implications in the back direction and in both directions, respectively. The abbreviation “CIP r+c” stands for “CIP for rows and for columns.” Of course, due to its symmetry an (augmented) adjacency matrix has the CIP or the Circ1P for rows iff it has the CIP or the Circ1P, respectively, for columns.

<table>
<thead>
<tr>
<th>graph class</th>
<th>adjacency matrix</th>
<th>augmented adjacency matrix</th>
<th>half adjacency matrix</th>
<th>maximal clique matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>convex-round</td>
<td>⇒ CIP</td>
<td>⇒ CIP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>concave-round</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>∩ [90]</td>
<td>⇒ quasi Circ1P</td>
<td>⇒ C1P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>circular-arc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊆</td>
<td>⇒ quasi Circ1P</td>
<td>⇒ C1P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Helly</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊆</td>
<td>⇒ C1P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>interval</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊆</td>
<td>⇒ C1P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proper / unit interval</td>
<td></td>
<td>⇒ C1P [85]</td>
<td>⇒ C1P r+c [35]</td>
<td></td>
</tr>
<tr>
<td>circular convex bipart.</td>
<td></td>
<td>⇒ C1P ([per def.])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊆</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>convex bipart.</td>
<td></td>
<td>⇒ C1P ([per def.])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊆</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>biconvex bipart.</td>
<td></td>
<td>⇒ C1P r+c ([per def.])</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The two characterizations given in the following theorem follow directly from the results of Tucker [91] described in the following paragraph and from the considerations of Hajiaghayi and Ganjali [51] and Tan and Zhang [88].

**Theorem 2.1.** A graph is a union of vertex-disjoint paths if and only if its edge-vertex incidence matrix has the C1P. A graph is a union of vertex-disjoint caterpillars if and only if its vertex-edge incidence matrix has the C1P.

See Figure 5 for an illustration.

**Tucker’s characterization of matrices having the C1P.** Matrices with the C1P can be characterized by a set of forbidden submatrices: A matrix has the C1P if it does not contain a matrix from this set as a submatrix. Such a characterization is very helpful when regarding matrix modification problems where one has to modify a matrix to achieve the C1P (see Section 1).

The characterization by Tucker [91] is based on a characterization of convex bipartite graphs in terms of so-called asteroidal triples defined as follows.

**Definition 2.5.** Let $G = (V, E)$ be a graph. Three vertices from $V$ form an asteroidal triple if between any two of them there exists a path in $G$ that does not contain any vertex from the closed neighborhood of the third vertex.

For example, every cycle of length at least six contains several asteroidal triples. More examples for graphs containing asteroidal triples are shown in Figure 6. In his characterization, Tucker does not use the term “convex bipartite...
Figure 6: Forbidden induced subgraphs due to Tucker [91]: The vertex set $V_2$ of a bipartite graph $G = (V_1, V_2, E)$ contains an asteroidal triple iff $G$ contains one of the displayed graphs as an induced subgraph, where white vertices correspond to vertices in $V_2$. The numbers $k$ and $k + 1$ refer to the number of black vertices in the lower parts of the first three graphs. In the case of the graph $G_{tk} \in \mathcal{T}$, every triple of white vertices is an asteroidal triple. In all other cases, there is exactly one asteroidal triple consisting of white vertices; this triple is denoted by $x, y, z$. 

A characterization that is very similar to the one given in Theorem 2.2 is also known for interval graphs: A graph is an interval graph iff it is chordal and contains no asteroidal triple [71]. The following theorem, which finally characterizes matrices with the C1P, is a direct consequence of Theorems 2.2 and 2.3.

**Theorem 2.4** ([91, Theorem 9]). A matrix $M$ has the C1P if and only if it contains none of the matrices $M_{I_k}, M_{II_k}, M_{III_k}$ (with $k \geq 1$), $M_{IV}$, and $M_{V}$ as shown in Figure 7 as a submatrix.\(^2\)

Clearly, if a matrix $M$ does not have the C1P, then it is possible to find in polynomial time one of the submatrices $M_{I_k}, M_{II_k}, M_{III_k}, M_{IV},$ and $M_{V}$ in $M$: just check for every row and every column of $M$ whether the matrix that results from the deleting the row or column, respectively, still does not have the C1P. It is also possible to find in polynomial time one of these submatrices that has a minimum

---

\(^1\)Tucker considers the C1P for columns, whereas we describe the C1P for rows. Hence, the roles of $V_1$ and $V_2$ are interchanged here compared to Tucker’s publication [91].

\(^2\)The roles of rows and columns are interchanged here compared to Tucker’s publication [91].
number of rows, a minimum number of columns, a minimum number of rows and columns, or a minimum number of entries [24]; however, we are not aware of a linear-time algorithm for this task. To our knowledge, there is no characterization similar to Theorem 2.4 for matrices with the Circ1P, although there is a kind of refinement of Theorem 2.4 with regard to the Circ1P [26, Theorem 4].

3 Recognizing the C1P

The C1P can be recognized in linear time. In this section, we survey several of the recognition algorithms. As we will see (Theorem 3.2), every algorithm that recognizes interval graphs can also be used to recognize matrices with the C1P. However, here we mention only those results that explicitly deal with matrices and the C1P, because first transforming a matrix into a graph and then testing whether this graph is an interval graph does not automatically yield an efficient (that is, linear-time) algorithm for recognizing the C1P. Some of the publications surveyed in this section consider the C1P for rows and some the C1P for columns; to give a consistent presentation, we formulate all results in terms of the C1P for rows. The following definition introduces some terms needed in this section.

Definition 3.1. Two rows \( r_1, r_2 \) of a 0/1-matrix overlap if there exist three columns \( c_1, c_2, c_3 \) such that column \( c_1 \) contains a 1 in both rows \( r_1 \) and \( r_2 \), column \( c_2 \) contains a 1 in \( r_1 \) but not in \( r_2 \), and column \( c_3 \) contains a 1 in \( r_2 \) but not in \( r_1 \).

A column \( c_1 \) contains a column \( c_2 \) if for every row \( r \) it holds that if \( c_2 \) has a 1 in row \( r \) then \( c_1 \) also has a 1 in row \( r \). If a column is not contained in any other column, it is maximal.\(^3\)

\(^3\)Note that this definition of a maximal column does not allow the existence of two identical
Using overlapping rows. The first polynomial-time algorithm to recognize matrices having the C1P\(^4\) was presented by Fulkerson and Gross [38]. Their idea is to decompose the input matrix \(M\) into disjoint row sets in such a way that the whole matrix has the C1P iff each matrix induced by one of the row sets has the C1P. The partitioning of \(M\)'s rows into different row sets is performed by defining an overlap graph \(G(M)\): Every vertex of this graph corresponds to a row of \(M\), and two vertices are connected iff their corresponding rows overlap. Every connected component of this graph defines one row set of the partition of \(M\) needed by the algorithm. Now, for the columns of every submatrix of \(M\) that is induced by one of the row sets of the partition, a C1-ordering can easily be found, if existing, by considering one row after the other in a certain order and re-arranging the columns if necessary. In the last phase of the algorithm, the column orderings computed for the submatrices are combined, while possibly re-arranging some of the columns. The whole matrix \(M\) has the C1P iff all of the submatrices have the C1P [38, Theorem 4.1]. The whole procedure takes polynomial time. A more recent linear-time algorithm by Hsu [61] for recognizing the C1P is based on very similar ideas.

PQ-Trees. Booth and Lueker were the first to present a linear-time algorithm for recognizing matrices with the C1P\(^4\) [13]. Linear time means a running time that is linear in the number of columns plus the number of rows plus the number of 1-entries of the given matrix. Booth and Lueker introduced so-called PQ-Trees, which are not only useful for recognizing matrices with the C1P, but also, for example, for recognizing matrices with the Circ1P and for recognizing interval graphs and planar graphs. In the context of recognizing matrices with the C1P, a PQ-tree is an ordered rooted tree that represents a C1-ordering for the columns of a matrix \(M\). To this end, the inner nodes of the PQ-tree are labeled as P-nodes and Q-nodes, and the leaves one-to-one correspond to the columns of the underlying matrix \(M\). Ordering the columns of \(M\) in the same way as their corresponding leaves in the PQ-tree yields the strong C1P. In addition, any PQ-tree for \(M\) implicitly represents all possible C1-orderings for \(M\)'s columns because by applying a series of certain node reordering operations every possible PQ-tree for the set of \(M\)'s columns can be transformed into any other possible PQ-tree for this column set. Therefore, PQ-trees have the following properties.

1. If \(T\) is a PQ-tree for a matrix \(M\), then the sequence of \(T\)'s leaves from left to right describes a C1-ordering for \(M\)'s columns.

---

\(^4\)Fulkerson and Gross [38] as well as Booth and Lueker [13] consider the C1P for columns, whereas we describe the C1P for rows. Hence, the roles of rows and columns are interchanged here compared to the original publications.
Figure 8: A matrix $M$ and the PQ-tree representing the column ordering of $M$.

2. Each C1-ordering for $M$’s columns one-to-one corresponds to a PQ-tree.
3. The set of PQ-trees for the columns of a matrix is closed under the following two operations:
   - Arbitrarily reordering the children of a P-node.
   - Putting the children of a Q-node in reverse order.

In particular, none of these two operations destroys property 1.

See Figure 8 for an illustration. In order to either construct a PQ-tree for a given matrix $M$ or decide that $M$ does not have the C1P, the algorithm of Booth and Lueker starts with a tree (the so-called universal PQ-tree) consisting of one P-node forming the root and one leaf node for every column of $M$. The algorithm considers the rows of $M$ one after the other, and in every step it either reports that $M$ does not have the C1P, or it modifies the tree, by using a complicated case distinction, in such a way that the resulting tree is a PQ-tree for the matrix that is induced by the rows considered so far.

Variations of PQ-Trees. Several variations of PQ-trees have been proposed since their first appearance. Korte and Möhring [66] introduced MPQ-trees (“modified PQ-trees”), where the inner nodes contain some additional information, which results in a simpler construction of these trees. Meidanis et al. [77] defined PQR-trees, which are a generalization of PQ-trees in the following sense: For every matrix $M$ that has the C1P for columns, the set of PQR-trees for $M$’s rows is identical with the set of PQ-trees for $M$’s columns. However, in contrast to PQ-trees, PQR-trees are also defined for matrices that do not have the C1P for columns; in this case they contain, in addition to P-nodes and Q-nodes, inner nodes labeled as R-nodes, which can be useful for identifying why the matrix does not have the C1P. A similar approach was used by McConnell [74]. He introduced generalized PQ-trees in order to determine if a matrix has the C1P for rows and, if it does not, to generate a “certificate” therefor: Such a certificate is a small (compared to the size of the input matrix) proof that can be verified by a “fast and
Figure 9: The matrix $M_{III}$ and its incompatibility graph [74]. The graph contains several odd cycles of length 7.

uncomplicated’ polynomial-time algorithm (for more details about such certificates see [69]). The certificate produced by the algorithm of McConnell [74] for an $m \times n$ input matrix $M$ consists of an odd cycle of length at most $n + 3$ in the so-called incompatibility graph of $M$. This graph is defined as the graph $G = (V,E)$ with

$$V = \{(j_1, j_2) \mid 1 \leq j_1 \leq n \land 1 \leq j_2 \leq n \land j_1 \neq j_2\}$$

$$E = \{(j_1, j_2), (j_2, j_1) \mid 1 \leq j_1 < j_2 \leq n\} \cup \{(j_1, j_2), (j_2, j_3) \mid j_1 \neq j_3 \land \exists l \in \{1, \ldots, m\} : (m_{i,l} = m_{i,j_1} = 1 \land m_{i,j_2} = 0)\}.$$

Intuitively speaking, the incompatibility graph contains two vertices for every pair of columns of $M$: one vertex for every possible relative ordering of the two columns. If two vertices $(j_1, j_2)$ and $(j_3, j_4)$ in the incompatibility graph are connected by an edge, then the corresponding two orderings conflict in the sense that there is no C1-ordering for $M$’s columns that places the column $j_1$ to the left of $j_2$ and the column $j_3$ to the left of $j_4$. See Figure 9 for an example. The connection between the incompatibility graph of a matrix and the CIP is specified in the following theorem. The original formulation of this theorem is due to McConnell [74] and contains a minor error concerning the cycle length.

**Theorem 3.1** ([74, Theorem 6.1]). An $m \times n$ matrix $M$ has the CIP iff its incompatibility graph is bipartite. If $M$ does not have the CIP, then its incompatibility graph has an odd cycle of length at most $n + 3$.

**PC-Trees.** A remarkable simplification for building PQ-trees was exhibited by Hsu and McConnell [63], who introduced PC-trees. These trees can be seen as unrooted PQ-trees that represent Circ1-circular orderings for the columns of a matrix instead of C1-orderings. Instead of Q-nodes, PC-trees contain C-nodes;
Figure 10: PC-trees for a matrix. Left: a matrix $M$. Middle: the PC-tree representing the circular column ordering of $M$. Right: The PC-tree obtained from the PC-tree in the middle by “flipping” the subtree rooted at the bottom-right C-node.

The order of the leaves of a PC-tree describes a Circ1-circular ordering for the columns of the underlying matrix. PC-trees have the following properties.

1. If $T$ is a PC-tree for a matrix $M$, then any sequence obtained by considering $T$’s leaves in clockwise or counter-clockwise order describes a Circ1-circular ordering for $M$’s columns.
2. Each Circ1-circular ordering for $M$’s columns one-to-one corresponds to a PC-tree.
3. The set of PC-trees for the columns of a matrix is closed under the following two operations:

- Arbitrarily reordering the neighbors of a P-node.
- First rooting $T$ at a neighbor of a C-node $v$, then “flipping” the subtree whose root is $v$, and finally un-rooting the tree. Herein, “flipping” a subtree means putting the children of every node of the subtree in reverse order.

In particular, none of these two operations destroys property 1.

See Figure 10 for an illustration.

Like in the case of PQ-trees, there is a linear-time algorithm that either constructs a PC-tree for a given matrix $M$ or decides that $M$ does not have the Circ1P [63]. Similarly to the algorithm of Booth and Lueker [13], this algorithm starts with a tree consisting of one P-node which has one leaf neighbor for every column of $M$. The algorithm considers the rows of $M$ one after the other, and in every step it either reports that $M$ does not have the Circ1P, or it modifies the tree. These modifications, however, are much simpler than those proposed by Booth and Lueker for updating a PQ-tree.
Due to Corollary 1.2, the PC-tree algorithm can not only be used to decide whether a matrix has the Circ1P, but also to decide whether it has the C1P: Just add a column that contains only 0s to the given matrix. Rooting the PC-tree for the resulting matrix at the neighbor of the leaf node that corresponds to the newly inserted column and then deleting this leaf node yields a PQ-tree for the original matrix.

Further recognition algorithms. A simple linear-time algorithm without using any variant of PQ-trees was presented by Habib et al. [47]: They use a so-called “Lex-BFS ordering” of the vertices of a graph to decide in linear time whether the graph is an interval graph. Habib et al. also prove Theorem 3.2 below, which implies that any algorithm recognizing interval graphs can also be used for recognizing matrices with the C1P. Habib et al. show that the recognition of matrices with the C1P in this way is possible in linear time.

For describing how an algorithm recognizing interval graphs can be used to decide whether a given matrix \( M \) has the C1P, let \( G_{ra}(M) \) denote the graph that has one vertex for every row of \( M \) and where two vertices are adjacent iff their corresponding rows in \( M \) have a 1 in a common column; we call this graph the row adjacency graph of \( M \). Note that several matrices can have identical row adjacency graphs; moreover, if a matrix \( M \) is the maximal clique matrix of a graph \( G \), then \( G \) is the row adjacency graph of \( M \). The latter observation leads to the following finding.

**Theorem 3.2** ([47, Theorem 2]). For a 0/1-matrix \( M \) the following statements are equivalent:

1. The row adjacency graph \( G_{ra}(M) \) is an interval graph and \( M \) is its maximal clique matrix.
2. The columns of \( M \) are maximal and \( M \) has the C1P for rows.

By appending a size-\( n \times n \) unit matrix \( I \) to a matrix \( M \), a matrix \( \tilde{M} = (M \ I) \) can be constructed, which obviously has the following properties: \( \tilde{M} \) has the C1P iff \( M \) has the C1P, every column of \( \tilde{M} \) is maximal, and \( \tilde{M} \) is the maximal clique matrix of \( G_{ra}(\tilde{M}) \). Testing whether \( M \) has the C1P now obviously reduces to checking whether \( G_{ra}(\tilde{M}) \) is an interval graph. However, note that using Theorem 3.2 in combination with a linear-time algorithm for recognizing interval graphs does not automatically yield a linear-time algorithm for recognizing matrices with the C1P. Nevertheless, Habib et al. give an algorithm using Theorem 3.2 for recognizing matrices with the C1P in linear time.

Clearly, if the columns and rows of a matrix \( M \) shall be permuted such that the resulting matrix contains at most one block of 1s per row and column, this can be...
done in linear time by first permuting the columns and then permuting the rows. However, weighted variants of this problem can be NP-hard [83]. Finally, we like to point out that recognizing matrices that have "almost the C1P" is much more difficult than recognizing matrices with the C1P: While matrices with the C1P can be recognized in linear time, the problem of deciding whether the columns of a matrix $M$ (not having the C1P) can be permuted in such a way that the overall number of blocks of 1s is at most $k$ is NP-complete [41, 48] (see also [49]); deciding whether $M$'s columns can be permuted such that the number of blocks of 1s in each row is at most $c$ is NP-complete for every constant $c \geq 2$ [6, 36, 45, 96] (see also [12, 97]). However, there are algorithms for recognizing matrices that are "close" to having the C1P in some other sense [60, 72].

Deciding whether a matrix can obtain the C1P by deleting a given number $d$ of rows or columns is NP-complete [41, 50, 51, 88]—however, for special cases there are approximation and fixed-parameter algorithms [26] (see [24] for a detailed overview). In related problems, 0-entries have to be flipped (that is, replaced by 1-entries) in order to obtain the C1P [41, 94], or arbitrary entries have to be flipped [82]. For all variants of these matrix modification problems as well as for the recognition of matrices being "close" to the C1P, (further) approximation and fixed-parameter results would be desirable.

4 Hard Problems on Instances with the C1P/Circ1P

Many in general NP-hard problems become polynomial-time solvable when restricted to inputs that have the C1P. In this section, we consider the in general NP-complete problems INTEGER LINEAR PROGRAMMING and SET COVER.

4.1 Integer Linear Programming with Coefficient Matrices having the C1P/Circ1P

Here, we consider the connection between matrices with the C1P or Circ1P and the hardness of solving integer linear programs (ILPs). While solving ILPs in general is NP-hard (see Section 4.1.1), we are here interested in the solvability of ILPs in the special case where the matrix consisting of the coefficients of the inequations has the C1P—such ILPs occur, for example, in biological applications [3]—or the Circ1P.

The algorithms that we consider do not only work for ILPs with 0/1-coefficient matrices, but also for ILPs whose coefficient matrices consist of entries from $\{0, 1, -1\}$. Therefore, we extend the definition of the C1P to 0/±1-matrices as follows: A 0/±1-matrix has the C1P if every row contains only entries either from $\{0, 1\}$ or from $\{0, -1\}$ and the columns can be permuted such that in every
row the non-zero entries appear consecutively. The Circ1P for 0/±1-matrices is defined analogously. We refer to Schrijver [87] for more details on (integer) linear programming.

4.1.1 (Integer) Linear Programming Basics

A linear program (LP) is an instance of the following optimization problem.

**Linear Programming**

**Input:** An \( m \times n \) matrix \( A = (a_{ij}) \), an \( n \)-entry column vector \( \vec{b} \) and an \( m \)-entry row vector \( \vec{c}^T \) with all entries in \( \mathbb{Q} \).

**Task:** Find an \( m \)-entry column vector \( \vec{x} \) that satisfies \( A\vec{x} \leq \vec{b} \) and maximizes \( \vec{c}^T \vec{x} \).

In other words, the task is to assign values to a set of variables \( x_1, \ldots, x_n \) (the entries of the vector \( \vec{x} \)) such that a set of \( m \) inequations (given by the matrix \( A \) and the vector \( \vec{b} \)) are satisfied and that the value of a linear objective function (given by the vector \( \vec{c}^T \)) is maximized. An assignment of values to the variables that satisfies the given inequations is called a (feasible) solution for the LP. A vector containing only integers is called integral. The variant of Linear Programming where only integral solutions are allowed is called Integer Linear Programming; its instances are called integer linear programs (ILPs). There are also decision problems corresponding to the optimization problems Linear Programming and Integer Linear Programming. The instances of these decision problems contain no objective function (no vector \( c \)); the task is to decide if there is any feasible solution.

Linear Programming can be solved in polynomial time. In particular, there is an algorithm for solving LPs that needs \( O((n^3/\ln n)L) \) arithmetic operations [4] (see also [84] for an overview over efficient algorithms for solving LPs), where \( L \) is the total bit number of the input. In contrast, it is easy to see that Integer Linear Programming is NP-hard. Actually, the decision version of Integer Linear Programming is NP-complete [14, 95, 64, 57].

An LP (ILP) is called feasible if it admits a feasible solution, and unfeasible otherwise. Given an LP on \( n \) variables, one can interpret its solution space as an \( n \)-dimensional Euclidean space; every inequation of the LP defines a half-space that contains all value-to-variable assignments satisfying this inequation. The intersection of all the half-spaces defined by the inequations of an LP is called the polyhedron defined by the LP. If the polyhedron defined by an LP is integral, which means that each of its corners corresponds to an integral solution, then the ILP defined by the matrix \( A \) and the vectors \( \vec{b}, \vec{c}^T \) of the LP can be solved in polynomial time by solving the LP.
Given an LP \( \text{Maximize } \mathbf{c}^T \mathbf{x} \text{ subject to } A \mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \), then the problem \( \text{Minimize } \mathbf{z}^T \mathbf{b} \text{ subject to } \mathbf{z}^T A = \mathbf{c}^T, \mathbf{z}^T \geq \mathbf{0}^T \) is called the dual (problem) for the given LP—here the task is to find an optimal row vector \( \mathbf{z}^T \). The dual problem has a well-defined optimal solution iff the original LP has a well-defined optimal solution; if this is the case then it holds that \( \max(\mathbf{c}^T \mathbf{x} | A \mathbf{x} \leq \mathbf{b}) = \min(\mathbf{z}^T \mathbf{b} | \mathbf{z}^T A = \mathbf{c}^T \land \mathbf{z}^T \geq \mathbf{0}^T) \).

Note that if the coefficient matrix of an LP has the C1P (Circ1P) for columns, then the coefficient matrix of its dual has the C1P (Circ1P) for rows. Therefore, all algorithms described below can be used for solving ILPs whose coefficient matrices have the C1P (Circ1P) for rows or the C1P (Circ1P) for columns.

### 4.1.2 Totally Unimodular Matrices

The problem \textsc{Integer Linear Programming} is NP-hard. However, there are special cases that can be solved in polynomial time. Here, we consider the special case where the coefficient matrix \( A \) of the ILP is “totally unimodular”—in this case, the polyhedron defined by \( A \mathbf{x} \leq \mathbf{b} \) is integral for every integral vector \( \mathbf{b} \). We will see that matrices with the C1P have this property. For a more detailed description of classes of matrices see [15, 21, 46, 87].

**Definition 4.1.** A \( 0/\pm 1 \)-matrix is \textit{totally unimodular} if every square submatrix has determinant 0, 1, or \( -1 \).

There is a useful “bicolorability” characterization for totally unimodular matrices.

**Theorem 4.1 ([44]).** An \( m \times n \) matrix \( A \) with entries 0, 1, \( -1 \) is totally unimodular if and only if each collection of columns from \( A \) can be partitioned into two column sets such that in each row the sum of the entries of the first set and the sum of the entries of the second set differ by at most 1.

The following theorem shows the polynomial-time solvability of ILPs with totally unimodular coefficient matrices.

**Theorem 4.2 ([56]).** Let \( A \) be an \( m \times n \) integral matrix. Then the polyhedron defined by \( A \mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \) is integral for every integral vector \( \mathbf{b} \in \mathbb{Z}^m \) if and only if \( A \) is totally unimodular.

From Theorem 4.2 it follows that for every totally unimodular matrix \( A \), every integral vector \( \mathbf{b} \) and every vector \( \mathbf{c}^T \) the ILP

\[
\begin{align*}
\text{Maximize } & \quad \mathbf{c}^T \mathbf{x} \\
\text{subject to } & \quad A \mathbf{x} \leq \mathbf{b} \\
& \quad \mathbf{x} \geq \mathbf{0} \\
& \quad \mathbf{x} \text{ is integral}
\end{align*}
\]
The Bulletin of the EATCS

can be solved with $O(n^3/\ln n)L$ arithmetic operations [4], where $L$ is the total bit number needed for encoding the ILP.

In the case of totally unimodular coefficient matrices $A$, the polyhedron defined by $A\vec{x} \leq \vec{b}, \vec{x} \geq \vec{0}$ is integral for every integral vector $\vec{b}$. There exist also matrices with the property that the polyhedron is integral only for certain vectors $\vec{b}$, these matrices are called balanced [10, 11, 20, 39, 89].

4.1.3 ILPs with Coefficient Matrices having the C1P

The first method to solve ILPs whose coefficient matrices have the C1P is to use the fact that any matrix $A$ having the C1P clearly fulfills the conditions of Theorem 4.1 and, hence, is totally unimodular. To see this, consider an arbitrary collection of columns from $A$ and order them according to the C1P. Partitioning the columns by putting every second column, starting with the first, into one column set and every remaining column into the other column set leads to a partitioning as required in Theorem 4.1 (see also [78, page 544]). Therefore, if a matrix $A$ has the C1P, then for every integral vector $\vec{b}$ and every vector $\vec{c}^T$ the ILP shown in (1) can be solved in polynomial time due to Theorem 4.2.

Using Theorem 4.2 to solve ILPs with coefficient matrices having the C1P exploits only the fact that such coefficient matrices are totally unimodular. However, it is known that an ILP whose coefficient matrix has the C1P can be solved even faster by transforming it into an edge-weighted graph and solving a shortest-path problem or a minimum-cost flow problem on this graph, depending on whether the decision version or the optimization version of INTEGER LINEAR PROGRAMMING is considered (see [93] and [2, pages 304–306] for the transformation of the optimization version into the flow problem and [2, pages 310–315] for the connection between minimum-cost flow problems and shortest-path problems; see also [78, pages 546–550]). The running time obtained in this way is $O(mn)$ for the decision version and $O(m^2 \log(n) + mn \log(n^2))$ for the optimization version; this approach is, therefore, much faster than using Theorem 4.2, where $O(n^3/\ln n)L$ operations [4] are needed—$L$ is the size of the ILP and, hence, lower-bounded by $mn$.

We start with showing how to solve the decision version of INTEGER LINEAR PROGRAMMING for coefficient matrices with the C1P by reducing it to a shortest path problem. We assume that the coefficient matrix $A$ of the given ILP has $m$ rows and that the rows of $A$ and the entries of $\vec{b}$ are sorted in such a way that the first $m' \leq m$ of these rows contain only entries from $[0, 1]$ and the remaining $m - m'$ rows contain only entries from $[0, -1]$. Moreover, we assume that $A$ has the strong C1P, which is not a restriction since a C1-ordering for $A$’s columns can be found in linear time (see Section 3). With $lx(i)$ and $rx(i)$ we denote the column index of the first and the last, respectively, non-zero entry in the $i$th row of $A$. Hence, an
instance of the problem to be solved consists of an inequation system as follows.

\[
\begin{align*}
x_{lx(i)} &+ x_{lx(i+1)} + \ldots + x_{rx(i)} \leq b_i & \forall i \in \{1, \ldots, m'\} \\
-x_{lx(i)} - x_{lx(i+1)} - \ldots - x_{rx(i)} &\leq b_i & \forall i \in \{m' + 1, \ldots, m\} \\
x_j &\in \mathbb{Z} & \forall j \in \{1, \ldots, n\}
\end{align*}
\] (2)

To transform the inequation system into a graph, we first drop the constraint of integrality and replace the \(n\) variables \(x_1, \ldots, x_n\) by \(n+1\) variables \(y_0, \ldots, y_n\) such that \(x_j = y_j - y_{j-1}\) for all \(j \in \{1, \ldots, n\}\). This yields the following inequation system.

\[
\begin{align*}
-y_{lx(i)} + y_{ly(i)} &\leq b_i & \forall i \in \{1, \ldots, m'\} \\
y_{lx(i)} - y_{ly(i)} &\leq b_i & \forall i \in \{m' + 1, \ldots, m\}
\end{align*}
\] (3)

In the resulting coefficient matrix, each row contains exactly one 1 and one \(-1\); hence, each row can be interpreted as a directed edge in a graph \(G\) whose vertices correspond to the variables \(y_0, \ldots, y_n\). More precisely, let \(G = (V, E)\) be the directed edge-weighted graph with

- \(V = \{v_j \mid \text{the inequation system (3) contains a variable } y_j\}\),
- \(E = \{(v_j, v_{j+1}) \mid \text{the inequation system (3) contains an inequation whose left side is } -y_j + y_{j+1}\}\),

where every edge \(e \in E\) has a weight that is equal to the right side of the inequation corresponding to \(e\) in the inequation system (3), see Figure 11.

Now consider the following statement known as Farkas’ Lemma (see [87]).

**Lemma 4.1.** Let \(A\) be an \(m \times n\) matrix with entries from \(\mathbb{R}\), and let \(\vec{b} \in \mathbb{R}^m\) be a vector. Then the inequation system \(A\vec{y} \leq \vec{b}\) has a solution \(\vec{y} \in \mathbb{R}^n\) if and only if the inequation system \(\vec{z}^T A = (0^n)^T, \vec{z}^T \vec{b} < 0, \vec{z} \geq 0^m\) has no solution \(\vec{z} \in \mathbb{R}^m\).

Applying Farkas’ Lemma to the inequation system (3), the lemma says that the inequation system is feasible iff \(G\) contains no negative cycle, that is, no directed cycle in which the sum of the edge weights is negative. To see this, observe that by interpreting the edge weights \(b_i\) as “per-flow-costs”, the inequation system \(\vec{z}^T A = (0^n)^T, \vec{z}^T \vec{b} < 0, \vec{z} \geq 0^m\) in Farkas’ Lemma can be interpreted as the following “negative-cost” flow problem on the graph \(G\): Find a flow function \(f : E \rightarrow \mathbb{R}\) such that

- the flow \(f(e)\) along every directed edge \(e \in E\) is nonnegative (expressed by the constraint \(\vec{z} \geq 0^m\)),
- for every vertex the sum of the ingoing and the outgoing flow is 0 (expressed by the constraint \(\vec{z}^T A = (0^n)^T\)), and
The Bulletin of the EATCS

\[ -y_0 + y_1 \leq 3 \quad (\text{interpreted as edge } e_1) \]
\[ -y_1 + y_2 \leq 4 \quad (\text{interpreted as edge } e_2) \]
\[ -y_2 + y_3 \leq 7 \quad (\text{interpreted as edge } e_3) \]
\[ y_0 - y_3 \leq 8 \quad (\text{interpreted as edge } e_4) \]
\[ y_0 - y_2 \leq -9 \quad (\text{interpreted as edge } e_5) \]

\[ z_1 - z_4 + z_5 = 0 \quad (\text{interpreted as flow conservation at } v_0) \]
\[ z_1 - z_2 \quad = 0 \quad (\text{interpreted as flow conservation at } v_1) \]
\[ z_2 - z_3 - z_5 = 0 \quad (\text{interpreted as flow conservation at } v_2) \]
\[ 3z_1 + 4z_2 + 7z_3 + 8z_4 - 9z_5 < 0 \]
\[ z_1, \ldots, z_5 \geq 0 \]

Figure 11: Solving an ILP whose coefficient matrix has the C1P. Top left: An example for the inequation system (3) obtained from an ILP with the C1P. Every row can be interpreted as an edge in a directed, edge-weighted graph \( G \). Top right: The graph corresponding to the ILP. Bottom: This inequation system is not feasible if the inequation system displayed at the top of the figure is feasible (Farkas’ Lemma). The inequation system at the bottom can be interpreted as a flow problem.

- the sum \( \sum_{i=1}^{m} b_i f(e_i) \) of all costs arising from sending flow along the edges is negative (expressed by the constraint \( \vec{z}^T \vec{b} < 0 \)).

See Figure 11. If a flow \( f \) has these three properties, then setting \( z_i = f(e_i) \) clearly yields a feasible solution for the LP. Obviously, a flow with negative cost can only exist if the graph \( G \) contains a negative cycle. By using the Bellmann-Ford-Moore-Algorithm (see [22]), it can be decided in \( O(|V| \cdot |E|) \) time whether \( G \) contains a negative cycle. Hence, the decision version of Integer Linear Programming with C1P can be decided in \( O(n \cdot m) \) time.

If \( G \) contains no negative cycle and a solution for the inequation system (3) shall be constructed (that is, the values of the \( y_j \) shall be computed), then just select an arbitrary \( k \in \{0, \ldots, n\} \) and set \( y_k \) to 0. For every \( j \in \{0, \ldots, n\} \setminus \{k\} \) for which there exists no directed path from \( v_k \) to \( v_j \) in \( G \), add an edge \((v_k, v_j)\) of weight \( |E| \cdot \max\{-b_i \mid i \in \{1, \ldots, m\} \setminus \{k\} \} \). Note that this operation does not create any negative cycles; note also that in the resulting graph \( G' \) every vertex is reachable from \( v_k \) on a directed path. Now, for every \( j \in \{0, \ldots, n\} \setminus \{k\} \), set \( y_j \) to the length of the shortest path in \( G' \) from \( v_k \) to \( v_j \). Since \( G' \) contains no negative cycle, these shortest paths are all well-defined. It is easy to see that this solution satisfies all inequalities of the inequation system (3). The shortest paths can be
computed by the Bellmann-Ford-Moore-Algorithm in $O(|V| \cdot |E|) = O(n \cdot m)$ time. A solution for the original ILP (2) can be computed by setting $x_j = y_j - y_{j-1}$ for all $j \in \{1, \ldots, n\}$.

For solving the optimization version of the problem, one has to use a minimum-cost flow algorithm instead of a shortest path algorithm. Again, start with omitting the integrality constraint and replacing the variables $x_1, \ldots, x_n$ by variables $y_0, \ldots, y_n$. The dual of the resulting LP can now be interpreted as a minimum-cost flow problem on a directed, vertex-weighted and edge-weighted graph, which is constructed in analogy to the graph used for solving the decision problem, and in which every vertex has a (positive or negative) “flow demand”. Computing a minimum-cost flow which respects the flow demands yields an optimal solution for the ILP. Such a minimum-cost flow can be found in $O(m^2 \log(n) + mn \log(n)^2)$ time [2].

4.1.4 ILPs with Coefficient Matrices having the Circ1P

Not all matrices that have the Circ1P are totally unimodular. For example, all matrices $M_I k$ (see Figure 7) with even $k$ are totally unimodular, while all matrices $M_I k$ with odd $k$ are not (this can easily be seen by using the characterization of Theorem 4.1). Nevertheless, every ILP whose coefficient matrix has the Circ1P can be solved in polynomial time by solving a series of ILPs that all have the C1P [8] (see also [2, page 346–347] and [55]).

To solve a given ILP

Maximize $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$
subject to $a_{1,1} x_1 + a_{1,2} x_2 + \ldots + a_{1,n} x_n \leq b_i \quad \forall i \in \{1, \ldots, m\}$
$x_j \in Z \quad \forall j \in \{1, \ldots, n\}$

(4)

whose $0/\pm 1$-coefficient matrix $A = (a_{i,j})$ has the Circ1P, define $L_k, k \in \mathbb{Z}$, as the ILP that results from appending the constraint

$x_1 + x_2 + \ldots + x_n = k$

(5)
to the the ILP (4). It is obvious that the ILP (4) is feasible iff there is a value $k$ such that $L_k$ is feasible. Moreover, if the ILP (4) is feasible, then there is a $k$ such that the optimal solution for $L_k$ is an optimal solution for the ILP (4): just set $k$ to the sum of the $x_i$ in an optimal solution for the ILP (4). Now, any ILP $L_k$ can be transformed into an ILP having the C1P: Add the equation (5) to every inequation of $L_k$ whose coefficients are from $[0, -1]$ and in which the non-zero coefficients do not appear consecutively, and subtract the equation (5) from every inequation of $L_k$ whose coefficients are from $[0, 1]$ and in which the non-zero coefficients do not appear consecutively. The resulting ILP is equivalent to the ILP $L_k$ (that
is, every feasible solution of the latter ILP is a feasible solution for \( L_k \) and vice versa) and has the C1P—therefore, it can be solved with the approach described in Section 4.1.3. The optimum value of \( k \) can be determined by a binary search [2, 8], such that the number of ILPs that have to be solved is linear in the size of the given ILP (4).

### 4.2 Set Cover Problems and the C1P

The C1P has attracted interest not least because it often makes hard problems easy. Our first example substantiating this statement was I\( \text{M} \)\( \text{B} \)\( \text{L} \)\( \text{E} \)\( \text{A} \)\( \text{T} \)\( \text{C} \)\( \text{S} \) in Section 4.1. As a second example, we consider the problem \( \text{S} \)\( \text{E} \)\( \text{H} \)\( \text{E} \)\( \text{L} \)\( \text{D} \)\( \text{E} \)\( \text{R} \)\( \text{S} \)\( \text{P} \). Formulated as a matrix problem, \( \text{S} \)\( \text{E} \)\( \text{H} \)\( \text{E} \)\( \text{L} \)\( \text{D} \)\( \text{E} \)\( \text{R} \)\( \text{S} \)\( \text{P} \) is defined as follows.

**Set Cover**

**Input:** A binary matrix \( M \) and a positive integer \( k \).

**Question:** Is there a set \( C' \) of at most \( k \) columns of \( M \) such that \( C' \) contains at least one 1 from every row?\(^5\)

Due to its generality, \( \text{Set Cover} \) has practical applications in almost all disciplines (see [22, 19, 16]); unfortunately, \( \text{Set Cover} \) is not only NP-hard, but it allows only for a logarithmic-factor polynomial-time approximation [34]. Moreover, \( \text{Set Cover} \) is \( \text{W}[2] \)-complete (that is, parameterized intractable) with respect to the parameter \( k = \text{"solution size"} \) [30]. In the \textit{weighted version} of \( \text{Set Cover} \), each column of the given matrix \( M \) has a positive integral weight and one asks for a column set \( C' \) of \textit{weight} at most \( k \).

#### 4.2.1 Set Cover with the C1P/Circ1P

Whereas \( \text{Set Cover} \) in general is NP-complete, the problem becomes polynomial-time solvable when the input matrix \( M \) has the C1P or the Circ1P. To solve such a restricted instance of \( \text{Set Cover} \), formulate the problem in a straightforward way as an ILP that has one variable for each column of \( M \) and whose coefficient matrix is \( M \). As described in Section 4.1, this ILP is polynomial-time solvable. Moreover, there is a well-known greedy algorithm for \( \text{Set Cover} \) on input matrices with the C1P: First order the columns of \( M \) such that in each row the 1s appear consecutively (this takes linear time, see Section 3), and then proceed from left to right as follows: Repeatedly search for a row \( r \) with minimum \( \text{rx}(r) \), where \( \text{rx}(r) \) denotes the index of the rightmost column having a 1 in row \( r \). Then take column \( c_{\text{rx}(r)} \) into the solution and delete all rows from \( M \) that have a 1 in column \( c_{\text{rx}(r)} \).

---

\(^5\)The reader may be familiar with \( \text{Set Cover} \) as a subset selection problem; however, the equivalence of our definition and the more common definition of \( \text{Set Cover} \) as a subset problem can easily be seen by identifying columns with subsets and rows with elements to be covered.
If the input matrix $M$ has the C1P or the Circ1P, even the weighted version of Set Cover can be solved in polynomial time: one can either use the ILP approach or, in case of input matrices having the C1P, use a simple dynamic programming algorithm. The following theorem summarizes these results.

**Theorem 4.3.** Weighted Set Cover can be solved in polynomial time if the input matrix has the C1P or the Circ1P (for rows or for columns).

A further approach for tackling Set Cover is to use polynomial-time data reduction rules: Such a reduction rule takes as input an instance $X$ of a problem and outputs in polynomial-time an instance $X'$ with $|X'| \leq |X|$ of the same problem such that $X'$ is a yes-instance iff $X$ is a yes-instance.

The following data reduction rules for Set Cover are well-known; their correctness is obvious.

—If $M$ contains two rows $r_i, r_j$ such that for each column $c_j$ it holds that $m_{i,j} = 1$ implies $m_{i,j} = 1$, then remove row $r_i$ from $M$.
—If $M$ contains two columns $c_i, c_j$ such that for each row $r_i$ it holds that $m_{i,j} = 1$ implies $m_{i,j} = 1$, then remove column $c_j$ from $M$.
—If $M$ contains a column $c_j$ without any 1-entry, then remove column $c_j$ from $M$.
—If $M$ contains a row $r_i$ that contains exactly one 1-entry $m_{i,j}$, then remove $r_i$ and all rows $r_r$ with $m_{r,j} = 1$ from $M$, remove column $c_j$ from $M$, and decrease $k$ by one.
—If $k \geq 0$ and $M$ has no rows, then answer “$M$ is a yes-instance.”
—If $M$ contains a row without any 1-entry, or if $k < 0$, or if $k = 0$ and $M$ contains at least one row, then answer “$M$ is a no-instance.”

For Set Cover without restrictions, these rules can be used in a preprocessing step in order to decrease the size of a problem instance before solving it. However, since there are Set Cover instances to which none of the rules applies, it is not possible to give any guarantee on the size of the problem instance resulting from the preprocessing step. If, however, the input matrix $M$ has the C1P (for rows or for columns), then the instance can be solved by iteratively applying the rules, that is, eventually one of the rules will output “$M$ is a yes-instance” or “$M$ is a no-instance.”

The C1P does not only help in the case of Set Cover, but even for more general problems: Red-Blue Set Cover [17] and Minimum-Degree Hypergraph [33] are generalizations of Set Cover; when restricted to instances that have the C1P, they become polynomial-time solvable [27].

**4.2.2 Set Cover with almost C1P**

In some applications, the arising Set Cover instances do not have the C1P, but are “close to the C1P”. Motivated by problems arising from railway optimization,
Mecke and Wagner [76], Ruf and Schöbel [86], and Mecke et al. [75] consider
Weighted Set Cover on input matrices that have “almost C1P”, which basically
means that either the input matrices have been generated by starting with a matrix
that has the C1P and replacing randomly a certain percentage of the 1’s by 0’s [76],
that the average number of blocks of 1’s per row is much smaller than the number
of columns of the matrix [86], or that the maximum number of blocks of 1’s
per row is small [75]. The latter restriction was also considered by Dom and
Sikdar [29] and Dom et al. [25]; here, the problem is interpreted as a a geometric
covering problem called Rectangle Stabbing. Apart from heuristics performing
well in practice [76, 86], the following results have been obtained.

**Theorem 4.4** ([25, 42, 75, 76]). 1. Set Cover is NP-complete even if the input
matrix $M$ can be split into two submatrices $M_1, M_2$ such that $M = (M_1 \mid M_2)$
and both $M_1$ and $M_2$ have the strong C1P [42, 75]. Moreover, this restricted
variant of Set Cover is W[$1$]-complete with respect to the parameter $k$ [25].

2. Weighted Set Cover restricted to input matrices with at most $d$ blocks of 1s
per row can be approximated in polynomial time with a factor $d$ [75].

3. Weighted Set Cover can be solved in $2^\ell \cdot \text{poly}(m, n)$ time with $\ell$ denoting the
maximum distance between the topmost and the bottommost 1 in any column
of $M$ [75, 76].

## 5 Concluding Remarks

In summary, the C1P can be recognized in linear time and many in general NP-
hard problems become polynomial-time solvable when restricted to inputs that
have the C1P. In particular, Integer Linear Programming and several variants
of Set Cover are polynomial-time solvable if the input has the C1P. In contrast,
the recognition of matrices that are “close” to the C1P (for example, matrices
whose columns can be permuted such that there are two blocks of 1s per row or
matrices that can obtain the C1P by few column or row deletions) is typically NP-
hard; moreover, NP-hard problems, such as Integer Linear Programming and Set
Cover, tend to stay NP-hard on inputs being “close” to the C1P.

## References


The Bulletin of the EATCS


The Bulletin of the EATCS


THE ALGORITHMIC GAME THEORY COLUMN

BY

MARIOS MAVRONICOLAS

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

RECENT ADVANCES IN APPROXIMABILITY OF BIMATRIX NASH *

Spyros Kontogiannis†,b Paul Spirakisb
Haralambos Tsaknakisb

† Department of Computer Science, University of Ioannina
Dourouti University Campus, 45110 Ioannina, Greece.
kontog@cs.uoi.gr

b Research Academic Computer Technology Institute
N. Kazantzaki Str., Patra University Campus, 26500 Rio-Patra, Greece.
{tsaknak,spirakis}@cti.gr

Abstract

In view of the apparent intractability of constructing Nash Equilibria (NE in short) of normal form games [8], even for only two players [4], understanding the limitations of the approximability of the problem is an important challenge. The purpose of this document is to present some recent, state-of-art advances in this particular line of research aiming towards polynomial time constructions of approximate Nash equilibria in bimatrix games.

*Work partially supported by the ICT PROGRAMME, FET Proactive Initiative: PERSVATIVE ADAPTATION (PERADA), with contract number 215270 (FRONTS).
1 Introduction

One of the most appealing concepts in game theory is the notion of a Nash equilibrium: A collection of strategies for the players from which no player has an incentive to unilaterally deviate from its own strategy. The extremely nice thing about Nash equilibria is that they always exist in any finite $k$-person normal form game [18]. This is one of the most important reasons why Nash equilibria are considered to be the prevailing solution concept for finite games in normal form. The problem with Nash equilibria is that there can be exponentially many of them, of quite different characteristics, even for bimatrix games. Additionally, we do not know yet how to construct them, even in subexponential time. Therefore, $k$-NASH, the problem of computing an arbitrary Nash equilibrium of a finite $k$-person game in normal form, is a fundamental problem in algorithmic game theory and perhaps one of the most outstanding problems at the boundary of P [20]. Its complexity has been a long standing open problem, since the introduction of the pioneering (pivoting) algorithm of Lemke and Howson [16]. Unfortunately, it has been shown (Savani, von Stengel [22]) that this algorithm requires an exponential number of steps; it is also known that the smoothed complexity of the algorithm is likely to be superpolynomial [5]. On the other hand, it is quite interesting that almost all (quite natural) refinements of $k$-NASH (e.g., are known to be NP-complete problems (e.g., existence of NE points assigning positive probability to a particular action of some player, or assuring some lower bound on a particular player’s payoff, e.t.c. [12, 7]).

A flurry of results in the last years have lead to the conclusion that $k$-NASH is indeed complete problem for the complexity class PPAD (introduced by Papadimitriou [19]). This was originally proved for at least four players [8], then for three players [11], and eventually even for two players [4]. In particular, the result of Chen and Deng [4], complemented by that of Abbott, Kane and Valiant [1], shows that 2-NASH is PPAD-complete even for win lose games.

Due to the apparent intractability of $k$-NASH for any $k \geq 2$, the research community has turned towards approximations of NE points. The two most popular notions of approximate Nash equilibria in the literature are (i) those which require that each player gets the maximum possible payoff, within some additive constant $\varepsilon$ (denoted here by $\varepsilon$-ApproxNE), and those which require that each player is allowed to adopt wpp only actions that are approximate best responses to the opponent’s strategy, within again an additive constant $\varepsilon$ (denoted here by $\varepsilon$-SuppNE). ApproxNE seem to be the dominant notion of approximation for NE points in the literature, while SuppNE is a rather new notion, yet quite interesting and qualitatively harder notion (e.g., see [5, 6, 9]). For sake of comparison, it

\footnote{With positive probability.}
is typical in the literature (mainly due to the additive nature in the error term) to consider bimatrix games whose payoff matrices have all their values from $[0, 1]$ (we call them $[0, 1]$-bimatrix, or normalized bimatrix games). Apart from additive notions of approximation for NE points, one could also use a relative error term, which is closer to the flavor of the CS community for approximate solutions. Nevertheless, since most of the literature focuses on these two additive notions of approximate NE points, we shall focus our interest on them.

Apart from the intractability of the exact problem $2\text{-NASH}$, the problem has also been shown to remain $\text{PPAD}$-hard even for the construction of $\varepsilon$-$\text{ApproxNE}$ (or even $\varepsilon$-$\text{SuppNE}$, due to a reduction from $\varepsilon$-$\text{ApproxNE}$ to $\frac{\varepsilon}{2}$-$\text{SuppNE}$) when $\varepsilon$ is allowed to be proportional to an inverse polynomial of the size of the game [5]. In particular, they proved that unless $\text{PPAD} \subseteq \text{P}$, there is no algorithm that constructs an $\varepsilon$-$\text{ApproxNE}$ in time $\text{poly}(n, 1/\varepsilon)$, for any $\varepsilon = n^{-O(1)}$. Moreover, they proved that unless $\text{PPAD} \subseteq \text{RP}$, there is no algorithm that constructs a NE in time $\text{poly}(n, 1/\sigma)$, where $\sigma$ is the size of the deviation of the elements of the bimatrix. This latter result essentially states that even the smoothed complexity of the algorithm of Lemke and Howson is not polynomial. Hence, the existence of a fully polynomial time approximation scheme (FPTAS) for both kinds of NE approximations seems unlikely as well.

On the positive side, [17] were the first to observe the existence of a quasi-polynomial time approximation scheme (QPTAS) for $\text{ApproxNE}$ in $2\text{-NASH}$. Recently, [15] extended this result to a QPTAS for the construction of $\text{SuppNE}$ for $2\text{-NASH}$, by a simple application of Althöfer's Approximation Lemma [2]. As of the writing of this survey, these two results are the best currently known constructions of NE approximations, for arbitrarily small (but constant) values of the error term $\varepsilon > 0$.

As for polynomial time constructions, two independent results [9, 13] initiated the discussion of providing in polynomial time $\varepsilon$-$\text{ApproxNE}$ for $[0, 1]$-bimatrix games and some constant $1 > \varepsilon > 0$. In particular, [9] gave a simple $\frac{1}{2}$-$\text{ApproxNE}$ for $[0, 1]$-bimatrix games, involving only two strategies per player. In [13], a simple algorithm for computing a $\frac{1}{4}$-$\text{ApproxNE}$ equilibrium for any bimatrix game in strongly polynomial time and we next showed how to extend this algorithm so as to obtain a (potentially stronger) parameterized approximation. Namely, we presented an algorithm that computes a $\frac{5}{4}$-$\text{ApproxNE}$, where $\lambda$ is the minimum, among all Nash equilibria, expected payoff of either player. The suggested algorithm runs in time polynomial in the number of strategies available to the players. A series of results then appeared, improving the constant for polynomial time constructions of $\text{ApproxNE}$. First [10] proposed an efficient construction of a $0.38$-$\text{ApproxNE}$, and consequently [3] proposed a $0.36392$-$\text{ApproxNE}$ based on the solvability of zero sum bimatrix games (an idea that was borrowed by [14] for the efficient construction of $\text{SuppNE}$). Finally, [23] proposed a new,
steepest descent method for determining stationary points of a regret function (for the two players of the bimatrix games) and, based on that stationary point, they provided a polynomial time algorithm for computing $0.3393$-$\text{ApproxNE}$ of any $[0, 1]$–bimatrix game. To our knowledge, this is currently the best result for polynomial time constructions of ApproxNE in bimatrix games.

Finally, for the efficient approximation of SuppNE, [9] introduced the problem and proposed a quite simple algorithm, which, under a quite interesting graph theoretic conjecture, constructs in polynomial time some non–trivial SuppNE. Unfortunately, the status of this conjecture is still unknown (it is false for some small instances of graphs). [9] made also a quite interesting connection of the problem of constructing $\frac{1}{4}$–SuppNE in an arbitrary $[0, 1]$–bimatrix game, to that of constructing $\varepsilon$–SuppNE for a properly chosen win lose game of the same size. [15] provides a $0.5$–SuppNE for win lose bimatrix games and a $0.667$–SuppNE for $[0, 1]$–bimatrix games. As of this writing, these are again the best approximation bounds for polynomial time constructions of SuppNE in bimatrix games.

1.1 Preliminaries

Mathematical Notation.

For any integer $k \in \mathbb{N}$, let $[k] \equiv \{1, 2, \ldots, k\}$. We denote by $M \in F^{\text{\textbf{max}}}$ any $m \times n$ matrix whose elements have values in some set $F$. We also denote by $(A, B) \in (F \times F)^{\text{\textbf{max}}}$ any $m \times n$ matrix whose elements are ordered pairs of values from $F$. Equivalently, this structure can be seen as an ordered pair of $m \times n$ matrices $A, B \in F^{\text{\textbf{max}}}$. Such a pair of matrices is called a bimatrix. A $k \times 1$ matrix is also considered to be an $k$-vector. Vectors are denoted by bold small letters (e.g., $\mathbf{x}, \mathbf{y}$). A vector having a 1 in the $i$-th position and 0 everywhere else is denoted by $e_i$. We denote by $1_k (0_k)$ the $k$-vector having 1s (0s) in all its coordinates. The $k \times k$ matrix $E = 1_k \cdot 1_k^\top \in \{1\}^{k \times k}$ has value 1 in all its elements. For a pair of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we denote the component–wise comparison by $\mathbf{x} \geq \mathbf{y}$: $\forall i \in [n], x_i \geq y_i$. Matrices are denoted by capital letters (e.g., $A, B, C, \ldots$), and bimatrices are denoted by ordered pairs of capital letters (e.g., $(A, B), (R, C), \ldots$).

For any $m \times n$ (bi)matrix $M$, $M_j$ is its $j$-th column (as an $m \times 1$ vector), $M^i$ is the $i$-th row (as a (transposed) $1 \times n$ vector) and $M_{ij}$ is the $(i, j)$-th element. For any integer $k \geq 1$, we denote by $\Delta_k \equiv [z \in \mathbb{R}^k : z \geq 0, (1_k)^\top z = 1]$ the $(k - 1)$-simplex. For any point $\mathbf{z} \in \Delta_k$, its support $\text{supp}(\mathbf{z})$ is the subset of coordinates with positive value: $\text{supp}(\mathbf{z}) \equiv \{i \in [k] : z_i > 0\}$. Also, let $\text{suppmax}(\mathbf{z}) \equiv \{i \in [k] : z_i \geq z_j \forall j \in [k]\}$ be the subset of coordinates with maximum value, and denote by $\text{max}(\mathbf{z})$ the value of the maximum entry of $\mathbf{z}$. For a subset of coordinates $S \subseteq [k]$, let $\text{max}_S(\mathbf{z})$ be the value of the maximum entry of vector $\mathbf{v}$ within the subset $S$. We denote by $\overline{S}$ the complement of a subset of coordinates $S$, i.e. $\overline{S} = \{i \in [k], i \notin S\}$.
For an arbitrary logical expression $E$, we denote by $P\{E\}$ the probability of this expression being true, while $I\{E\}$ is the indicator variable of whether this expression is true or false. For any random variable $x$, $E\{x\}$ is its expected value (with respect to some probability measure).

**Game Theoretic Definitions and Notation.**

An $m \times n$ **bimatrix game** $\langle A, B \rangle$ is a 2–person game in normal form, that is determined by the bimatrix $(A, B) \in (\mathbb{R} \times \mathbb{R})^{m \times n}$ as follows: The first player (called the **row player**) has an $m$–element action set $[m]$, and the second player (called the **column player**) has an $n$–element action set $[n]$. Each row (column) of the bimatrix corresponds to a different action of the row (column) player. The row and the column player’s payoffs are determined by the $m \times n$ real matrices $A$ and $B$ respectively. In the special case that the payoff matrices have only rational entries, we refer to a **rational bimatrix game**. If both payoff matrices belong to $[0,1]^{m \times n}$, then we have a $[0,1]$–**bimatrix** (or else **normalized** game). The special case of bimatrix games in which all elements of the bimatrix belong to $\{0,1\} \times \{0,1\}$, is called a $\{0,1\}$–**bimatrix** (or else, **win lose** game). A win lose game having (for some integer $\lambda \geq 1$) at most $\lambda$ 1, 0)–elements per row and at most $\lambda$ number (0, 1)–element per column of the bimatrix, is called a $\lambda$–**sparse** win lose game. A bimatrix game $\langle A, B \rangle$ is called **zero sum**, if it happens that $B = -A$. In that case the game is solvable in polynomial time, since the two players’ optimization problems form a primal–dual linear programming pair. In all cases of bimatrix games we assume wlog\(^2\) that $2 \leq m \leq n$.

Any probability distribution on the action set $[m]$ of the row player, i.e., any point $x \in \Delta_m$, is a **mixed strategy** for her. I.e, the row player determines her action independently from the column player, according to the probability distribution determined by $x$. Similarly, any point $y \in \Delta_n$ is a mixed strategy for the column player. Each extreme point $e_i \in \Delta_n$ ($e_j \in \Delta_n$) that enforces the use of the $i$-th row ($j$-th column) by the row (column) player, is called a **pure strategy** for her. Any element $(x, y) \in \Delta_m \times \Delta_n$ is a (mixed in general) **strategy profile** for the players. We now define the set of approximate best responses for the two players, that will help us simplify the forthcoming definitions:

**Definition 1** (Approximate Best Response). Fix arbitrary constant $\varepsilon > 0$. Given that the column player adopts a strategy $y \in \Delta_n$ and the payoff matrix of the row player is $A$, the row player’s sets of $\varepsilon$–**approximate (pure)** best responses are:

$$
BR(\varepsilon, A, y) = \{x \in \Delta_m : x^T Ay \geq x^T Ay - \varepsilon, \forall z \in \Delta_n\}
$$

$$
PBR(\varepsilon, A, y) = \{i \in [m] : A'y \geq A'y - \varepsilon, \forall r \in [m]\}
$$

\(^2\)Without loss of generality.
Similarly we define the column player’s sets of \( \varepsilon \)-approximate (pure) best responses:

\[
BR(\varepsilon, B^T, x) = \left\{ y \in \Delta_n : y^T B^T x \geq z^T B^T x - \varepsilon, \forall z \in \Delta_n \right\}
\]

\[
PBR(\varepsilon, B^T, x) = \left\{ j \in [n] : B^T_j x \geq B^T_{\epsilon} x - \varepsilon, \forall r \in [n] \right\}.
\]

For the notion of Nash equilibria, originally introduced by Nash [18], we give the definition wrt\(^3\) bimatrix games:

**Definition 2** (Nash Equilibrium). For any bimatrix game \((A, B)\), a profile \((x, y) \in \Delta_m \times \Delta_n\) is a Nash Equilibrium point (NE in short), iff \(x \in BR(0, A, y)\) and \(y \in BR(0, B^T, x)\). Equivalently, \((x, y) \in \Delta_m \times \Delta_n\) is a NE of \((A, B)\) iff \(supp(x) \subseteq PBR(0, A, y)\) and \(supp(y) \subseteq PBR(0, B^T, x)\). The set of profiles that are NE of \((A, B)\) is denoted by \(NE(A, B)\).

Due to the apparent difficulty in computing NE for arbitrary bimatrix games, the recent trend is to look for approximate equilibria. Two definitions of approximate equilibria that concern this paper are the following:

**Definition 3** (Approximate Nash Equilibria). For any \(\varepsilon > 0\) and any bimatrix game \((A, B)\), \((x, y) \in \Delta_m \times \Delta_n\) is an \(\varepsilon\)-approximate Nash Equilibrium point (\(\varepsilon\)-ApproxNE) iff each player chooses an \(\varepsilon\)-approximate best response against the opponent: \( [x \in BR(\varepsilon, A, y) ] \wedge [y \in BR(\varepsilon, B^T, x) ] \) \((x, y) \) is an \(\varepsilon\)-well-supported Nash Equilibrium point (\(\varepsilon\)-SuppNE) iff each player assigns positive probability only to \(\varepsilon\)-approximate pure best responses against the strategy of the opponent: \( \forall i \in [n], x_i > 0 \Rightarrow i \in PBR(\varepsilon, A, y) \) and \( \forall j \in [n], y_j > 0 \Rightarrow j \in PBR(\varepsilon, B^T, x) \)

It is not hard to see that any NE is both a 0–ApproxNE and a 0–SuppNE. Observe also that any \(\varepsilon\)-SuppNE is an \(\varepsilon\)-ApproxNE, but not necessarily vice versa. Indeed, the only thing we currently know towards this direction is that from an arbitrary \(\varepsilon_{\text{lnu}}\)-ApproxNE one can construct an \(\varepsilon\)-SuppNE in polynomial time [5]. Note that both notions of approximate equilibria are defined wrt an additive error term \(\varepsilon\). Although (exact) NE are known not to be affected by any positive scaling, it is important to mention that approximate notions of NE are indeed affected. Therefore, from now on we adopt the commonly used assumption in the literature (e.g., [17, 9, 13, 4, 5]) that, when referring to \(\varepsilon\)-ApproxNE or \(\varepsilon\)-SuppNE, the bimatrix game is considered to be a \([0, 1]\)-bimatrix game. This is mainly done for sake of comparison of the results on approximate equilibria. Of particular importance are the uniform points of the \((k - 1)\)-simplex \(\Delta_k\):

---

\(^3\)With respect to.
Definition 4 (Uniform Profiles). A point \( x \in \Delta_r \) is called a \( k \)-uniform strategy iff \( x \in \Delta_r \cap \{ 0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1 \} \) \( \equiv \Delta_r(k) \). I.e. \( x \) assigns to each action a probability mass that is some multiple of \( \frac{1}{k} \). In the special case that the only possibility for an action is to get either zero probability or \( \frac{1}{k} \), we refer to a strict \( k \)-uniform strategy. We denote the space of strict \( k \)-uniform strategies by \( \hat{\Delta}_r(k) \equiv \Delta_r \cap \{ 0, \frac{1}{k} \} \). A profile \( (x, y) \in \Delta_m \times \Delta_n \) for which \( x \) is a (strict) \( k \)-uniform strategy and \( y \) is a (strict) \( \ell \)-uniform strategy, is called a (strict) \((k, \ell)\)-uniform profile.

We shall finally denote by \( k \)-NASH the problem of constructing an arbitrary NE for a finite \( k \)-player game in normal form.

2 An Optimization Approach for ApproxNE

In this section we present an efficient algorithm that computes a 0.3393−ApproxNE for any \([0, 1]\)-bimatrix game, the best approximation till now. The methodology is based on the formulation of an appropriate function of pairs of mixed strategies reflecting the maximum deviation of the players’ payoffs from the best payoff each player could achieve given the strategy chosen by the other. We then seek to minimize such a function using descent procedures. For further details we refer the interested reader to [23].

Optimization formulation.

Let \( \Gamma = (A, B) \) be an \( m \times n \) \([0, 1]\)-bimatrix game. Key to our approach is the definition of the following continuous function mapping \( \Delta_m \times \Delta_n \) into \([0, 1]\]: \( \varPhi(x, y) = \max \{ \max(Ay) - x^T Ay, \max(B^T x) - x^T By \} \). It is evident that \( \varPhi(x, y) \geq 0 \) for all \((x, y) \in \Delta_m \times \Delta_n \) and that exact Nash equilibria of \((A, B)\) correspond to pairs of strategies such that \( \varPhi(x, y) = 0 \). Furthermore, \( \varepsilon \)-ApproxNE correspond to strategy pairs that satisfy \( \varPhi(x, y) \leq \varepsilon \). This function represents the maximum deviation of the players’ payoffs from the best payoff each player could achieve given the strategy chosen by the other. An optimization formulation based on mixed integer programming methods was suggested in [21]. However, no approximation results were obtained there.

The function \( \varPhi(x, y) \) is not jointly convex with respect to both \( x \) and \( y \). However, it is convex in \( x \) alone, if \( y \) is kept fixed and vice versa. Let us define the two ingredients of the function \( \varPhi(x, y) \) as follows: \( \varPhi_\alpha(x, y) = \max(Ay) - x^T Ay \) and \( \varPhi_\beta(x, y) = \max(B^T x) - x^T By \). From any point in \((x, y) \in \Delta_m \times \Delta_n \) we consider variations of \( \varPhi(x, y) \) along feasible directions in both players’ strategy spaces of the following form:

\[
(1 - \varepsilon) \begin{bmatrix} x \\ y \end{bmatrix} + \varepsilon \begin{bmatrix} x' \\ y' \end{bmatrix}
\]
The variation of the function along such a feasible direction is defined by the following relationship: \( Df(x, y, x', y', \varepsilon) = f(x + \varepsilon(x' - x), y + \varepsilon(y' - y)) - f(x, y) \).

We have been able to derive an explicit formula for \( Df(x, y, x', y', \varepsilon) \) which is a piecewise quadratic function of \( \varepsilon \) and the number of switches of the linear terms of the function is at most \( m + n \). Therefore, for fixed \((x', y')\) this function can be minimized with respect to \( \varepsilon \) in polynomial time. Furthermore, there always exists a positive number, say \( \varepsilon^* \), such that for any \( 0 \leq \varepsilon \leq \varepsilon^* \) the coefficient of the linear term of this function of \( \varepsilon \) coincides with the gradient, as defined below. The number \( \varepsilon^* \) generally depends on both \((x, y)\) and \((x', y')\). We define the gradient of \( f \) at the point \((x, y)\) along an arbitrary feasible direction specified by another point \((x', y')\) as follows: \( Df(x, y, x', y') = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} Df(x, y, x', y', \varepsilon) \).

The gradient \( Df(x, y, x', y') \) of \( f \) at any point \((x, y)\) \((x, y) \in \Delta_m \times \Delta_n \) along a feasible direction (determined by another point \((x', y') \in \Delta_m \times \Delta_n \)) provides the rate of decrease (or increase) of the function along that direction. For fixed \((x, y)\), \( Df(x, y, x', y') \) is a convex polyhedral function in \((x', y')\). In fact we have derived the explicit form of \( Df(x, y, x', y') \) as the maximum of two linear forms in the \((x', y')\) space (see the derivations below). At any point \((x, y)\) we wish to minimize the gradient function with respect to \((x', y')\) to find the steepest possible descent direction, or to determine that no such descent is possible.

Let us define the following subsets of coordinates: \( S_A(y) = \text{suppmax}(Ay) \) and \( S_B(x) = \text{suppmax}(B^Tx) \). By definition, \( S_A(y) \subset [m] \) and \( S_B(x) \subset [n] \). There are three cases:

(a) If \( f_a(x, y) = f_B(x, y) \) then
   \[
   Df(x, y, x', y') = \max(T_1(x, y, x', y'), T_2(x, y, x', y')) - f(x, y)
   \]
   where \( m_1(y') = \max(Ay') \) over the subset \( S_A(y) \), \( m_2(x') = \max(B^Tx') \) over the subset \( S_B(x) \), \( T_1(x, y, x', y') = m_1(y') - x^T Ay' - (x')^T Ay + x^T Ay \) and \( T_2(x, y, x', y') = m_2(x') - x^T By' - (x')^T By + x^T By \).

(b) If \( f_a(x, y) > f_B(x, y) \) then \( Df(x, y, x', y') = T_1(x, y, x', y') - f(x, y) \).

(c) If \( f_a(x, y) < f_B(x, y) \) then \( Df(x, y, x', y') = T_2(x, y, x', y') - f(x, y) \).

The problem of finding \( Df(x, y) \) as the minimum over all \((x', y') \in \Delta_m \times \Delta_n \) of the function \( Df(x, y, x', y') \), is a linear programming problem.

This problem can be equivalently expressed as the following minmax problem by introducing appropriate dual variables (we derive it for \((x, y)\) such that \( f_a(x, y) = f_B(x, y) \) since this is the most interesting case and the cases where the two terms are different can be reduced to this by solving an LP, as we shall see.
The maximum is taken with respect to dual variables $w, z, \rho$ such that $w \in \Delta_m, supp(w) \subset S_A(y)$ and $z \in \Delta_n, supp(z) \subset S_B(x)$ and $\rho \in [0, 1]$.

(b) The minimum is taken with respect to $(x', y') \in \Delta_m \times \Delta_n$.

(c) The matrix $G(x, y)$ is the following $(m + n) \times (m + n)$ matrix:

$$G(x, y) = \begin{bmatrix}
A - 1_mx^TA & -1_my^TB + 1_n1_n^T \times B \\
-1_nx^TB + 1_m1_m^T \times y & B^T - 1_ny^TB^T
\end{bmatrix}$$

The probability vectors $w$ and $z$ play the role of price vectors (or penalty vectors) for penalizing deviations from the support sets $S_A(y)$ and $S_B(x)$, and the parameter $\rho$ plays the role of a trade-off parameter between the two parts of the function $f(x, y)$. In fact, $w, z$ and $\rho$ are not independent variables but they are taken all together to represent a single $(m + n)$-dimensional probability vector on the left hand side (the maximizing term) of the linear minmax problem.

Solving the above minmax problem we obtain $w, z, \rho, x'$ and $y'$ that are all functions of the point $(x, y)$ and take values in their respective domains of definition. Let us denote by $V(x, y)$ the value of the solution of the minmax problem at the point $(x, y)$. The solution of this problem yields a feasible descent direction (as a matter of fact the steepest feasible descent direction) for the function $f(x, y)$ if $Df(x, y) = V(x, y) - f(x, y) < 0$. Following such a descent direction we can perform an appropriate line search with respect to the parameter $\varepsilon$ and find a new point that gives a lower value of the function $f(x, y)$. Applying repeatedly such a descent procedure we will eventually reach a point where no further reduction is possible. Such a point is a stationary point that satisfies $Df(x, y) \geq 0$. In the next subsection we examine the approximation properties of stationary points. In fact, we prove that given any stationary point we can determine pairs of strategies such that at least one of them is a $0.3393$-approximate Nash equilibrium.

**Approximation properties of stationary points.**

Let us assume that we have a stationary point $(x^*, y^*)$ of the function $f(x, y)$. Then, based on the above analysis and notation, the following relationship should be true: $Df(x^*, y^*) = V(x^*, y^*) - f(x^*, y^*) \geq 0$. Let $(w^*, z^*) \in \Delta_m \times \Delta_n, \rho^* \in [0, 1]$ be a solution of the linear minmax problem (with matrix $G(x^*, y^*)$) with
The pair of strategies $(x^*, y^*)$. Such a solution should satisfy the relations $\text{supp}(w^*) \subset S_A(y^*)$ and $\text{supp}(z^*) \subset S_B(x^*)$. Let us define the following quantities: $\lambda = \min_{z^* : \text{supp}(z^*) \subset S_B(x^*)} (w^* - x^* )^T Ay^*$ and $\mu = \min_{x^* : \text{supp}(x^*) \subset S_A(y^*)} (x^* )^T B (z^* - y^*)$. From the fact that $(A, B)$ is a $[0, 1]$-bimatrix game, it follows that both $\lambda$ and $\mu$ are less than or equal to $1$. At any point $(x^*, y^*)$ these quantities basically define the rates of decrease (or increase) of the function $f$ along directions of the form $(1-\epsilon)(x^*, y^*) + \epsilon(x^*, y^*)$ and $(1-\epsilon)(x^*, y^*) + \epsilon(x^*, y^*)$, i.e. the rates of decrease that are obtained when we keep one player’s strategy fixed and move probability mass of the other player into his own maximum support, towards decreasing his own deviation from the maximum payoff he can achieve.

From the stationarity property of the point $(x^*, y^*)$ it follows that both $\lambda$ and $\mu$ are nonnegative. Indeed, in the opposite case there would be a descent direction, which contradicts the stationarity condition. Let us define a pair of strategies $(\tilde{x}, \tilde{y}) \in \Delta_n \times \Delta_n$ as follows:

$$
(\tilde{x}, \tilde{y}) = \begin{cases} 
(x^*, y^*) & \text{if } f(x^*, y^*) \leq f(\tilde{x}, \tilde{y}) \\
(\bar{x}, \bar{y}) & \text{otherwise}
\end{cases}
$$

where

$$
(\bar{x}, \bar{y}) = \begin{cases} 
\left( \frac{1}{1+\epsilon-\mu} w^* + \frac{\lambda-\mu}{1+\epsilon-\mu} z^* \right) & \text{if } \lambda \geq \mu \\
\left( w^* + \frac{\lambda-\mu}{1+\epsilon-\mu} y^* \right) & \text{if } \lambda < \mu.
\end{cases}
$$

We now express the main result of this section in the following theorem:

**Theorem 1.** The pair of strategies $(\tilde{x}, \tilde{y})$ defined above, is a $0.3393$-approximate Nash equilibrium.

**Proof.** From the definition of $(\tilde{x}, \tilde{y})$ we have:

$$
f(\tilde{x}, \tilde{y}) \leq \min \{ f(x^*, y^*) , f(\bar{x}, \bar{y}) \}. \tag{1}
$$

Using the stationarity condition for $(x^*, y^*)$ we obtain that $f(x^*, y^*) \leq V(x^*, y^*)$. But $V(x^*, y^*)$ is less than or equal to $\rho^* E_1 + (1 - \rho^*) E_2$ where $E_1 = (w^* B y^* - x^* x^* B y^* + x^* A y^*)$ and $E_2 = (z^* B x^* - x^* B y^* + x^* A y^*)$ and this holds $\forall (x', y') \in \Delta_n \times \Delta_n$.

Setting $x' = x^*$ and $y' : \text{supp}(y') \subset S_B(x^*)$ in the above inequality we get:

$$
f(x^*, y^*) \leq \rho^* \lambda. \tag{2}
$$

Next, setting $y' = y^*$ and $x' : \text{supp}(x') \subset S_A(y^*)$ in the same inequality, we get:

$$
f(x^*, y^*) \leq (1 - \rho^*) \mu. \tag{3}
$$

69
Now using the definition of the strategy pair \((\tilde{x}, \tilde{y})\) above and exploiting the inequalities \((w^* - x^*)^T A z^* \geq \lambda\), since \(\text{supp}(z^*) \subseteq S_B(x^*)\), and \(w^T B (z^* - y^*) \geq \mu\), since \(\text{supp}(w^*) \subseteq S_A(y^*)\), we obtain (assume \(\lambda \geq \mu\)):

\[
f_A(\tilde{x}, \tilde{y}) = \max(A \tilde{y}) - \tilde{x}^T A \tilde{y} = \max(A z^*) - \left(\frac{1}{1 + \lambda - \mu} w^* + \frac{\lambda - \mu}{1 + \lambda - \mu} x^*\right)^T A z^*
\]

\[
= \max(A z^*) - \frac{1}{1 + \lambda - \mu} w^T A z^* - \frac{\lambda - \mu}{1 + \lambda - \mu} x^T A z^*
\]

\[
\leq \max(A z^*) - x^T A z^* - \frac{\lambda}{1 + \lambda - \mu} \leq \frac{1 - \mu}{1 + \lambda - \mu}.
\]

Similarly, it can be shown that \(f_B(\tilde{x}, \tilde{y}) \leq \frac{1 - \mu}{1 + \lambda - \mu}\). From the above relationships we obtain:

\[
f(\hat{x}, \hat{y}) \leq \frac{1 - \mu}{1 + \lambda - \mu} \quad \text{for } \lambda \geq \mu \quad (4)
\]

A similar inequality can be obtained if \(\lambda < \mu\) and we interchange \(\lambda\) and \(\mu\). In all cases, combining inequalities (2), (3), (4) and using the definition of \((\hat{x}, \hat{y})\) above, we get the following:

\[
f(\hat{x}, \hat{y}) \leq \min\left\{\rho^* \lambda, (1 - \rho^*) \mu, \frac{1 - \min(\lambda, \mu)}{1 + \max(\lambda, \mu) - \min(\lambda, \mu)}\right\} \quad (5)
\]

We can prove that the quantity in (5) cannot exceed the number 0.3393 for any \(\rho^*, \lambda, \mu \in [0, 1]\), and this concludes the proof of the theorem.

A stationary point of any general Linear Complementarity problem can be approximated arbitrarily close in polynomial time via the method of Ye [24]. In [23] we give an alternative approach, directly applicable to our problem; our method is an FPTAS with respect to approximating a stationary point and hence an approximate equilibrium of the stated quality.

### 3 QPTAS for SuppNE

In this section we prove the existence of SuppNE of arbitrary (constant) precision, with logarithmic (in the numbers of players’ actions) support sizes. This result is analogous to the existential result [17] of ApproxNE arbitrary (constant) precision, with logarithmic (in the numbers of players’ actions) support sizes. Nevertheless, our approach is different from that in [17], since we prove this result as an application of Althöfer’s Approximation Lemma [2]. These two existential results for ApproxNE and SuppNE imply (to our knowledge) the only quasi-polynomial approximation schemes respectively (by a simple exhaustive-search argument).
The Bulletin of the EATCS

We initially recall the Approximation Lemma of Althöfer [2], which will be used in our existence proof of SuppNE of arbitrarily small precision and logarithmic support sizes:

**Theorem 2** (Approximation Lemma [2]). Assume any \( m \times n \) real matrix \( C \in [0,1]^{m \times n} \), any probability vector \( p \in \Delta_n \) and any constant \( \varepsilon > 0 \). Then, there exists another probability vector \( \hat{p} \in \Delta_n \) with \( \supp(\hat{p}) \leq k \equiv \lceil \log(2n)/(2\varepsilon^2) \rceil \), such that \( |p^T C_i - \hat{p}^T C_i| \leq \varepsilon, \forall j \in [n] \). Moreover, \( \hat{p} \) is a \( k \)-uniform strategy, i.e., \( \hat{p} \in \Delta_k(k) \).

The following simple observation will be quite useful in our discussion:

**Proposition 1.** For any real matrix \( C \in [0,1]^{m \times n} \) and any \( p \in \Delta_n \) for the empirical distribution \( \hat{p} \in \Delta_n \) produced by the Approximation Lemma it holds that positive probabilities only to rows whose indices belong to \( \supp(\hat{p}) \subseteq \supp(p) \).

We now demonstrate how the Approximation Lemma, along with the previous observation, guarantees the existence of a uniform profile which is also a \((2\varepsilon)\)-SuppNE with support sizes at most \( \lceil \log(2n)/(2\varepsilon^2) \rceil \), for any \( \varepsilon > 0 \):

**Theorem 3.** Fix any positive constant \( \varepsilon > 0 \) and any \([0,1] \)-bimatrix game \( \langle A,B \rangle \). There is at least one \((k,\ell)\)-uniform profile which is also a \((2\varepsilon)\)-SuppNE for this game, where \( k \leq \lceil \log(2n)/(2\varepsilon^2) \rceil \) and \( \ell \leq \lceil \log(2m)/(2\varepsilon^2) \rceil \).

**Proof.** Assume any profile \( \langle p,q \rangle \in NE(A,B) \), which we of course know to exist for any finite game in normal form [18]. We use the Approximation Lemma to assure the existence of some \( k \)-uniform strategy \( \hat{p} \in \Delta_n \) with \( \supp(\hat{p}) \leq k \equiv \lceil \log(2n)/(2\varepsilon^2) \rceil \), such that \( |p^T B_j - \hat{p}^T B_j| \leq \varepsilon, \forall j \in [n] \). Similarly, we assume the existence of some \( \ell \)-uniform strategy \( \hat{q} \in \Delta_n \) with \( \supp(\hat{q}) \leq \ell \equiv \lceil \log(2m)/(2\varepsilon^2) \rceil \), such that \( |A'q - \hat{A}'q| \leq \varepsilon, \forall i \in [m] \).

Observe now that, trivially, \( \hat{p}^T B - 1^T \cdot \varepsilon \leq p^T B \leq \hat{p}^T B + 1^T \cdot \varepsilon \) . Similarly, \( A' \hat{q} - 1 \cdot \varepsilon \leq A \cdot q \leq A' \hat{q} + 1 \cdot \varepsilon \) . Therefore (also exploiting the Nash Property of \( \langle p,q \rangle \) and the fact that \( \supp(\hat{p}) \subseteq \supp(p) \)) we have: \( \forall i \in [m] \),

\[
\hat{p}_i > 0 \quad \implies \quad p_i > 0 \quad \implies \quad A'q \geq A'q, \forall r \in [m] \\
\implies \quad A'q + \varepsilon \geq A'q_i - \varepsilon, \forall r \in [m] \implies \quad A'q \geq A'q - 2\varepsilon, \forall r \in [m]
\]

The argument for the column player is identical. Therefore, we conclude that \( \hat{p}, \hat{q} \) is a \((k,\ell)\)-uniform profile that is also a \((2\varepsilon)\)-SuppNE for \( \langle A,B \rangle \). \( \square \)
4 A Linear Programming Approach for Constructing SuppNE

We shall now exploit the tractability of zero sum games due to their connection to linear programming, in order to provide a 0.5–SuppNE for arbitrary win lose games and a 0.667–SuppNE for any normalized bimatrix game.

4.1 Construction of a 0.5–SuppNE for Win Lose Games

In this subsection we provide a 0.5–SuppNE for win lose games, which directly translates to a 0.75–SuppNE for arbitrary normalized games, if one exploits the nice observation of [9]. But first we remark that additive transformations (i.e., shifting) have no effect on well supported equilibria:

Lemma 1. Fix arbitrary $[0, 1]$–bimatrix game $(A, B)$ and any real matrices $R, C \in \mathbb{R}^{m \times n}$, such that $\forall i \in [m], R^i = r^i \in \mathbb{R}^n$ and $\forall j \in [n], C^j = c^j \in \mathbb{R}^m$. Then, $\forall 1 > \varepsilon > 0$, $(x, y) \in \Delta_m \times \Delta_n$, if $(x, y)$ is an $\varepsilon$–SuppNE for $(A, B)$ then it is also an $\varepsilon$–SuppNE for $(A + R, B + C)$.

Proof. The proof is rather simple and therefore it is left as an exercise. □

Our next theorem tries to construct the “right” zero sum game that would stand between the two extreme zero sum games $(R, −R)$ and $(−C, C)$, wrt an arbitrary win lose bimatrix game $(R, C)$.

Theorem 4. For arbitrary win lose bimatrix game $(A, B)$, there is a polynomial time constructible profile that is a 0.5–SuppNE of the game.

Proof. Consider arbitrary win lose game $(A, B) \in \{(0, 0), (0, 1), (1, 0)\}^{m \times n}$. We have excluded the $(1, 1)$–elements because, as we already know, these are trivial PNE of the game. We transform the bimatrix $(A, B)$ into a bimatrix $(R, C)$ by subtracting $1/2$ from all the possible payoffs in the bimatrix: $R = A - \frac{1}{2}E$ and $C = B - \frac{1}{2}E$, where $E = 1 \cdot 1^T$. We already know that this transformation does not affect the quality of a SuppNE (cf. Lemma 1).

We observe that the row player would never accept a payoff less than the one achieved by the (exact) Nash equilibrium $(\hat{x}, \hat{y})$ of the (zero sum) game $(R, −R)$. This is because strategy $\hat{x}$ is a maximin strategy for the row player, and thus the row player can achieve a payoff of at least $\hat{V}_I \equiv \hat{x}^T R \hat{y}$ by adopting $\hat{x}$, for any possible column that the column player chooses wpp. Similarly, the column player would never accept a profile $(x, y)$ with payoff for her less than $\hat{V}_{II} \equiv \hat{x}^T C \hat{y}$, where $(\hat{x}, \hat{y})$ is the (exact) NE of the zero sum game $(−C, C)$. So, we already know that any 0–SuppNE for $(R, C)$ should assure payoffs at least $\hat{V}_I$ and at
least $V_{ll}$ for the row and the column player respectively. Clearly, $(\bar{x}, \bar{y})$ is a $\max \left\{ \frac{1}{2} - \bar{V}_r, \frac{1}{2} - \bar{V}_l \right\}$-ApproxNE of the game, but we cannot assure that it is a nontrivial SuppNE of $(R, C)$. Nevertheless, inspired by this observation, we attempt to set up the right zero sum game that is somehow connected to $(R, C)$, whose (exact) NE would provide a good SuppNE for $(R, C)$. Therefore, we consider an arbitrary zero sum game $(D, -D)$, for which it holds that $D = R + X \Rightarrow X = D - R$ and $-D = C + Y \Rightarrow Y = -(D + C)$ for some $m \times n$ bimatrix $(X, Y)$. Let again $(\bar{x}, \bar{y}) \in NE(D, -D)$. Then we have (by Nash property):

$$(\bar{x}, \bar{y}) \in NE(D, -D) = NE(R + X, C + Y) \Rightarrow \forall i, r \in [m], \forall j, s \in [n],\begin{cases} \bar{x}_i > 0 \Rightarrow (R + X)\bar{y} \geq (R + X)\bar{y} \Rightarrow (R + X)\bar{y} \geq (R + X)\bar{y} \Rightarrow (C + Y)\bar{x} \geq (C + Y)\bar{x} \Rightarrow (C + Y)\bar{x} \geq (C + Y)\bar{x} \Rightarrow (\bar{x}, \bar{y}) \in NE(D, -D) \end{cases}$$

Since $D = R + X = -(D) = -(C + Y) \Rightarrow -Z \equiv R + C = -(X + Y)$, we can simply set $X = Y = \frac{1}{2}Z$, and then we conclude that:

$$(\bar{x}, \bar{y}) \in NE(D, -D) \Rightarrow \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R\bar{y} \geq R\bar{y} - \frac{1}{2} \cdot [Z' - Z']\bar{y} \Rightarrow C\bar{x} \geq C\bar{x} - \frac{1}{2} \cdot [Z - Z]\bar{x} \Rightarrow (\bar{x}, \bar{y}) \in \frac{1}{2}-\text{SuppNE}(R, C).$$

Observe now that, since $R, C \in \left\{\left(-\frac{1}{2}, \frac{1}{2}\right), \left(-\frac{1}{2}, \frac{1}{2}\right), \left(\frac{1}{2}, -\frac{1}{2}\right)\right\}^{\text{mon}}$, any row of $Z = -(R + C)$ is a vector in $[0, 1]^m$, and any column of $Z$ is a vector in $[0, 1]^n$. But it holds that $\forall \bar{x}, \bar{z} \in [0, 1]^k, \forall w \in \Delta_m, (\bar{z} - \bar{y})^T \bar{w} \leq \bar{1}^T \bar{w} = 1$. So we conclude that $\forall i, r \in [m], \forall y \in \Delta_m, [Z' - Z']\bar{y} \leq \bar{1}^T \bar{y} = 1$, and $\forall j, s \in [n], \forall x \in \Delta_n, [Z - Z]\bar{x} \leq \bar{1}^T \bar{x} = 1$. Therefore we conclude that:

$$(\bar{x}, \bar{y}) \in NE \left(\frac{1}{2}Z, C + \frac{1}{2}Z\right) \Rightarrow \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R\bar{y} \geq R\bar{y} - \frac{1}{2} \Rightarrow C\bar{x} \geq C\bar{x} - \frac{1}{2} \Rightarrow (\bar{x}, \bar{y}) \in \frac{1}{2}-\text{SuppNE}(R, C).$$

\[\square\]

### 4.2 SuppNE for Normalized Bimatrix Games

Given our result on win lose games, applying a lemma of Daskalakis et al. [9, Lemma 4.6] for constructing $\frac{1}{2}\epsilon$-SuppNE of a $[0, 1]$-bimatrix game $(A, B)$ by any $\epsilon$-SuppNE of a properly chosen win lose game of the same size, we could directly generalize our result to SuppNE for any $[0, 1]$-bimatrix game:

**Corollary 1.** For any $[0, 1]$-bimatrix game $(R, C)$, there is a $0.75$-$\text{SuppNE}$ that can be computed in polynomial time.
The question is whether we can do better than that. Indeed we can, if we modify the rationale of the proof of Theorem 4. This way we shall get a parameterized approximation for [0, 1]–bimatrix games. The next theorem demonstrates this parameterized method.

**Theorem 5.** For any [0, 1]–bimatrix game \((R, C)\), and the matrix \(Z = -(R + C)\), there is a polynomial–time constructible \(\varepsilon(\delta)\)–SuppNE for any \(0 < \delta < 1\), where \(\varepsilon(\delta) \leq \max \{\delta, 1 - \delta\} \cdot (Z_{\text{max}} - Z_{\text{min}})\).

**Proof.** We try to find a zero sum game that lies somehow between \((R, -R)\) and \((-C, C)\) and indeed provides a guaranteed SuppNE for \((R, C)\). Therefore, we fix a constant \(\delta \in (0, 1)\), to be determined later. Consequently, we consider the matrix \(Z = -(R + C)\). The zero sum bimatrix game \((R + \delta Z, -(R + \delta Z))\) is solvable in polynomial time (by use of linear programming). We denote with \((\bar{x}, \bar{y})\) the (exact) NE of this game. By the definition of NE, the row and the column player assign positive probability mass only to maximizing elements of the vectors \((R + \delta Z)\bar{y}\) and \(-(R - \delta Z)\bar{x}\) respectively. Obviously, for any \(\delta \in [0, 1]\) it holds that \((\bar{x}, \bar{y}) \in \text{NE}(R + \delta Z, -(R + \delta Z)) \Leftrightarrow \forall i, r \in [m], \forall j, s \in [n],\)

\[
\begin{align*}
\bar{x}_i > 0 & \Rightarrow (R + \delta Z)\bar{y} \geq (R + \delta Z)\bar{y} \\
\bar{y}_j > 0 & \Rightarrow (-R - \delta Z)\bar{x} \geq (-R - \delta Z)\bar{x}
\end{align*}
\]

where,

\[
\varepsilon(\delta) = \max_{i \in [m], j \in [n], x \in \Delta_n, y \in \Delta_n} \left\{ \delta \cdot \left[ Z' - Z\right] y, (1 - \delta) \cdot \left[ Z'_j - Z_j\right] x \right\}
\]

\[
\leq \max(\delta, 1 - \delta) \cdot (Z_{\text{max}} - Z_{\text{min}})
\]  

The last inequality holds since the vectors \(x \in \Delta_n\) and \(y \in \Delta_n\) considered in the definition of \(\varepsilon(\delta)\) are probability distributions over the rows and the columns of \(Z\) respectively. Obviously, for any \(\delta \in [0, 1]\) it holds that \((\bar{x}, \bar{y})\) is an \(\varepsilon(\delta)\)–SuppNE for \((R, C)\).

We already know that for win lose bimatrix games \(\forall i, r \in [m], \forall y \in \Delta_n, [Z' - Z']y \leq 1^\top y = 1\). This directly yields the result of Theorem 4, if we simply set \(\delta = 0.5\). But let’s see what can be said about arbitrary [0, 1]–bimatrix games:

**Theorem 6.** For any [0, 1]–bimatrix game, a \(\frac{2}{3}\)–SuppNE is constructible in polynomial time.
The Bulletin of the EATCS

Proof. Our initial steps are in complete analogy as in the proof of Theorem 5. Therefore, we know how to construct in polynomial time an $\varepsilon(0.5)$–SuppNE, where, $\varepsilon(0.5) \leq \frac{Z_{\min}}{Z_{\max}}$.

Let’s assume now that, for some $0 < \zeta < 1$, we are in the seek of some $\zeta$–SuppNE of $(R, C)$. It is clear that the existence of any element $(R, C)_{i,j} \in [1 - \zeta, 1] \times [1 - \zeta, 1]$ would indicate a (pure) profile $(e_i, e_j)$ that is already a $\zeta$–SuppNE. Since these are detectable in time $O(nm)$, we suppose wlog that for each element of the bimatrix $(R, C)$, it holds that $(R_{i,j} < 1 - \zeta) \lor (C_{i,j} < 1 - \zeta)$. Now, for $Z$ we observe that $\forall (i, j) \in [m] \times [n],$

\[
\begin{align*}
\text{if} & \quad 0 \leq R_{i,j}, C_{i,j} < 1 - \zeta \\
\text{then} & \quad -2 + 2\zeta < Z_{i,j} = -(R_{i,j} + C_{i,j}) \leq 0 \\
\text{else} & \quad (0 \leq R_{i,j} < 1 - \zeta \leq C_{i,j} \leq 1) \lor (0 \leq C_{i,j} < 1 - \zeta \leq R_{i,j} \leq 1) \\
& \quad \implies -2 + \zeta < Z_{i,j} = -(R_{i,j} + C_{i,j}) \leq -1 + \zeta
\end{align*}
\]

So, since $0 \leq \zeta < 1$, we conclude that $Z \in (-2 + \zeta, 0]^{\max}$ and therefore, $\forall i, r \in [m], \forall y \in A_r, (Z' - Z')y \leq Z_{\max} - Z_{\min} < 2 - \zeta$, which implies that $\varepsilon(0.5) = 1 - \frac{1}{2}$. Since our approximation is max $\{\zeta, 1 - \frac{1}{2}\}$, our best choice is to set $\zeta^* = \frac{1}{2}$ and we are done. □

5 Open Problems

In this work we have presented some recent advances concerning the tractability of NE points in bimatrix games. We focused on the tractability of additive notions of approximation, namely ApproxNE and SuppNE, and demonstrated the state-of-the-art results to date, both for quasipolynomial and polynomial time constructions.

The important questions whether there exist polynomial time approximation schemes (PTAS) for the construction of $\varepsilon$–SuppNE or $\varepsilon$–ApproxNE, for any positive constant $1 > \varepsilon > 0$, still remain open. For the time being, we only know that the construction of fully polynomial time approximation schemes (FPTAS) for the weaker notion of ApproxNE is as hard as the exact problem. But this does not exclude the existence of some PTAS for either ApproxNE or SuppNE.

It would also be interesting even to find polynomial time algorithms for constructing $\varepsilon$–SuppNE, for some constant $0 < \varepsilon < 0.5$ for win lose games and $0 < \varepsilon < 0.667$ for the general case. Similarly, for the case of other notion of approximate equilibria (ApproxNE), we do not currently know how to construct $\varepsilon$–ApproxNE for some precision $0 < \varepsilon < \frac{1}{2}$, or whether there is a matching lower bound on the approximability of ApproxNE.
References


THE COMPUTATIONAL COMPLEXITY COLUMN

BY

JACOBO TORÁN

Dept. Theoretische Informatik, Universität Ulm
Oberer Eselsberg, 89069 Ulm, Germany
jacobo.toran@uni-ulm.de

Integer multiplication is one of the most basic and important arithmetic functions and its complexity in several computation models has been subject of research for many decades. There are nevertheless still many open questions related to the exact complexity of this function. Beate Bollig gives in this survey a beautiful overview of several recent results dealing with the complexity of multiplication in the Ordered Binary Decision Diagram model.

INTEGRAL MULTIPLICATION AND THE COMPLEXITY OF BINARY DECISION DIAGRAMS

Beate Bollig∗

Abstract

Integer multiplication as one of the basic arithmetic functions has been in the focus of several complexity theoretical investigations and ordered binary decision diagrams (OBDDs) are one of the most common dynamic data structures for Boolean functions. The BDD complexity of two output bits of integer multiplication, the so-called middle bit and the most significant bit, has been investigated intensively. In this column we briefly survey results on the complexity of restricted binary decision diagrams for integer multiplication and concentrate on two recent results on the complexity of OBDDs for

∗TU Dortmund, LS 2 Informatik, beate.bollig@uni-dortmund.de
1 Introduction

Integer multiplication is certainly one of the most important functions in computer science and a lot of effort has been spent in designing good algorithms and small circuits and in determining its complexity. For some computation models integer multiplication is a quite simple function. It is contained in $\text{NC}^1$ (polynomial-size threshold circuits of depth 3) but neither in $\text{AC}^0$ (polynomial-size $\{\lor, \land, \neg\}$-circuits of unbounded fan-in and constant depth) nor in $\text{TC}^{0,2}$ [21].

For more than 35 years the algorithm of Schönhage-Strassen [30] has been the fastest method for integer multiplication running in time $O(n \log n \log \log n)$. Only recently, Fürer has presented an algorithm running in time $n \log n \cdot 2^{O(\log^* n)}$, where the running time holds for multitape Turing machines [16]. A n algorithm with the same running time based on modular arithmetic has been obtained by De, Kurur, Saha, and Sapthariski [13]. Until now it is open whether integer multiplication is possible in time $O(n \log n)$.

Definition 1.1. Let $B_{n,m}$ denote the set of all Boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and $B_n$ the special case that $m = 1$. The Boolean function $\text{MUL}_{i,n} \in B_{2n}$ maps two $n$-bit integers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$ to the $i$th bit of their product, i.e., $\text{MUL}_{i,n}(x, y) = z_i$, where $x \cdot y = z_{2n-1} \ldots z_0$ and $x_0, y_0, z_0$ denote the least significant bits. For $c \in \{0, 1\}^n$ the Boolean function $\text{MUL}_c \in B_n$ is defined by $\text{MUL}_{c,n}(x) = \text{MUL}_{c,i,n}(x, c)$.

The Boolean function $\text{MUL-Graph}_n \in B_{4n}$ maps two $n$-bit integers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$, and a $2n$-bit integer $z = z_{2n-1} \ldots z_0$ to 1 iff the product of $x$ and $y$ equals $z$.

Besides Boolean circuits and formulas, circuits whose underlying graph is a tree after a suitable duplication of the inputs, branching programs (BPs) are one of the standard representations for Boolean functions. (For a history of results on branching programs see, e.g., the monograph of Wegener [32]).

Definition 1.2. A branching program (BP) on the variable set $X_n = \{x_1, \ldots, x_n\}$ is a directed acyclic graph with one source and two sinks labeled by the constants 0 and 1. Each non-sink node (or decision node) is labeled by a Boolean variable and has two outgoing edges, one labeled by 0 and the other by 1. An input $b \in \{0, 1\}^n$ activates all edges consistent with $b$, i.e., the edges labeled by $b_i$, which leave nodes labeled by $x_i$. A computation path for an input $b$ in a BP $G$ is a path of edges activated by the input $b$ which leads from the source to a sink. A computation
path for an input \( b \) which leads to the 1-sink is called an \textit{accepting path} for \( b \). The BP \( G \) represents a function \( f \in B_n \) for which \( f(b) = 1 \) if there exists an accepting path for the input \( b \). The size of a branching program \( G \) is the number of its nodes. The \textit{branching program size} of a Boolean function \( f \) is the size of the smallest BP representing \( f \). The length of a branching program is the maximum length of a path.

Not only in complexity theory but also in applications people have used (restricted) branching programs, where they are most often called binary decision diagrams (BDDs). Representations of Boolean functions that allow efficient algorithms for many operations, in particular synthesis (combine two functions by a binary operation) and equality test (do two representations represent the same function?) are necessary. Bryant [11] introduced ordered binary decision diagrams (OBDDs) which have become one of the most popular data structures for Boolean functions. Among the many areas of application are verification, model checking, computer-aided design, and symbolic graph algorithms.

Lower and upper bounds for integer multiplication are motivated by the general interest in the complexity of important arithmetic functions. The complexity of two output bits of integer multiplication has been investigated intensively in the last years. The first one is the middle bit of integer multiplication, the bit with significance \( 2^{n-1} \), which is the hardest bit to compute for space bounded models of computation in the sense that if it can be computed with size \( s(n) \), then any other bit can be computed with size at most \( s(2n) \). More precisely, any branching program for MUL\(_{2n-1,2n}\) can be converted into a branching program representing MUL\(_{i,n}\), \( 0 \leq i \leq 2n - 1 \), by relabeling the nodes and by replacing some inputs with the constant 0. As a consequence the first large lower bounds on the size of restricted branching programs have been shown for MUL\(_{2n-1,n}\). The second one is the bit \( z_{2n-1} \) which is the most important bit of integer multiplication in the following sense. Since it has the highest value, for the approximation of the value of the product of two \( n \)-bit numbers \( x \) and \( y \) it is the most interesting one. On the other hand for space bounded models of computation \( z_{2n-1} \) is easy to compute in the sense that if it cannot be computed with size \( s(n) \), then any other bit \( z_i, 2n - 1 > i \geq 0 \), cannot be computed with size \( s(i/4) \).

In the following we give some motivation for the investigation of the OBDD size of integer multiplication.

\textbf{The middle bit of integer multiplication}

A lot of effort has been spent in trying to verify multiplier circuits using OBDDs. In 1998 an OBDD for the 16-bit multiplication circuit \( c6288 \), one of the most important ISCAS (International Symposium on Circuits and Systems) benchmark
circuits, has been constructed [39]. To the best of our knowledge, until now it has been impossible to construct OBDDs for input length \( n = 32 \) and even the representation of all output bits of 16-bit multiplication by SBDDs, a more general OBDD model for the representation of multiple output Boolean functions, is a challenging task. Since the size of OBDDs and SBDDs can be quite sensitive to the chosen variable order, one of the reasons might be that different output bits of integer multiplication have different variable orders leading to reasonable size (for the definition of OBDDs and variable orders see Definition 2.2). Bryant [11] has already bounded the size of SBDDs for integer multiplication by proving that for each variable order there exists an output bit for which the OBDD size is at least \( 2^{n/8} \). For many applications it would be sufficient to represent each output bit by an OBDD of moderate size according to a suitably chosen variable order. Already Bryant has destroyed this hope in 1991 [12]. He has shown that OBDDs for the representation of the middle bit of integer multiplication have at least size \( 2^{n/8} \) for any variable order. Nevertheless, Bryant’s lower bound does not exclude that even 256-bit multiplication can be represented in reasonable size. Therefore, the OBDD size for integer multiplication has been further investigated.

### The most significant bit of integer multiplication

In the last years a new research branch has emerged which is concerned with the theoretical design and analysis of so-called symbolic algorithms for classical graph problems on OBDD-represented graph instances (see, e.g., [17, 18], [28], and [38]). Symbolic algorithms have to solve problems on a given graph instance by efficient functional operations offered by the OBDD data structure. Therefore, at the beginning the OBDD-based algorithms have been justified by analyzing the number of executed OBDD operations (see, e.g., [17, 18]). Since the runtime of an operation on an OBDD \( G \) often depends on the size of \( G \) the analysis of the over-all runtime of symbolic methods including the analysis of all OBDD sizes occurring during such an algorithm is more significant (see, e.g., [38]). In order to investigate the limits of symbolic graph algorithms for the all pairs shortest paths problem Sawitzki [28] has investigated the graph of integer multiplication and has presented an exponential lower bound on its OBDD size. Afterwards he has defined inputs for the all pairs shortest paths problem such that during the computation representations for MUL-Graph_\( n \) are necessary. Another investigated graph problem is the following. Computing the set of nodes that are reachable from some source \( s \in V \) in a digraph \( G = (V,E) \) is an important problem in computer-aided design, hardware verification, and model checking. Proving exponential lower bounds on the space complexity of a common class of OBDD-based algorithms for this reachability problem, Sawitzki [29] has presented the first exponential lower bound on the size of \( \pi \)-OBDDs representing the most significant
bit for the variable order $\pi$ where the variables are tested according to increasing significance, i.e. $\pi = (x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1})$. For the lower bounds on the space complexity of the OBDD-based algorithms he has used the assumption that the output OBDDs use the same variable order as the input OBDDs. But in contrast, practical algorithms usually run variable reordering heuristics on intermediate OBDD results in order to minimize their size. Therefore, lower and upper bounds on the OBDD size of the most significant bit of multiplication with respect to an arbitrary variable order are interesting.

**Organization**

In this column we give a brief overview on results concerning the complexity of restricted branching programs or binary decision diagrams for the functions $\text{MUL}_{n-1, n}$ and $\text{MUL}_{2n-1, n}$. We do not aim to be comprehensive but focus on their OBDD complexity, in particular new results on the most significant bit. Since the article is meant to be self-contained, in Section 2 we start with the presentation of some restricted branching program or binary decision diagram models. Moreover, we repeat the relevant relation between one-way communication complexity and the size of OBDDs.

Section 3 contains results on the size of restricted branching programs or binary decision diagrams for the middle bit of integer multiplication. We present lower and upper bounds on the OBDD size and sketch results on the size of more general models.

The main results of this survey are presented in Section 4. Using only methods from one-way communication complexity Sawitzki’s restricted lower bound on the size of OBDDs for $\text{MUL}_{2n-1, n}$ [29] is improved. Afterwards a general lower bound and upper bounds are presented. Remarks on the size of more general models for the most significant bit complete our investigation.

Finally, in Section 5 we summarize our results by a comparison between the middle and the most significant bit.

## 2 Preliminaries

In this section we introduce some notation. Furthermore, we give an overview on some restricted branching program or binary decision diagram models and provide relevant technical background from communication complexity.

### 2.1 Notation

In the rest of the paper we use the following notation.
Let $[x]_l^n$, $n-1 \geq l \geq r \geq 0$, denote the bits $x_l \ldots x_r$ of a binary number $x = (x_{n-1}, \ldots, x_0)$. For the ease of description we use the notation $[x]_l^n = z$ if $(x_l, \ldots, x_r)$ is the binary representation of the integer $z \in [0, \ldots, 2^{n-r+1}-1]$. Sometimes, we identify $[x]_l^n$ with $z$ if the meaning is clear from the context. We use the notation $[x]_l^n = z$ if the mean is clear from the context. We use the notation $(z)_l^n$ for an integer $z$ to identify the bits at position $l, \ldots, r$ in the binary representation of $z$.

Let $\ell \in \{0, \ldots, 2^m-1\}$, then $\ell$ denotes the number $(2^m-1) - \ell$. For a binary number $x = (x_{n-1}, \ldots, x_0)$ we use the notation $\overline{x}$ for the binary number $(\overline{x}_{n-1}, \ldots, \overline{x}_0)$.

Let $a_S$ be an assignment to variables in a set $S$ and $a_S(x_k) \in \{0, 1\}$ be the assignment to $x_k \in S$, then we define $\|a_S\| := \sum_{x_k \in S} a_S(x_k) \cdot 2^k$.

In the following for the sake of simplicity we do not apply floor or ceiling functions to numbers even when they need to be integers whenever this is clear from the context and has no bearing on the essence of the presented proofs.

### 2.2 Restricted branching programs or binary decision diagrams

It is well known that the logarithm of the branching program size is essentially the same as the space complexity of the nonuniform variant of Turing machines (see, e.g., [32]). Hence, it is a fundamental open problem to prove superpolynomial lower bounds on the size of branching programs for explicitly defined Boolean functions, i.e., functions contained in NP. In order to develop and strengthen lower bound techniques one considers restricted computation models. There are several possibilities to restrict branching programs, among them restrictions on the multiplicity of variable tests or the order in which variables may be tested.

**Definition 2.1.**

i) A branching program is called (syntactically) read-$k$-times (BP$k$) if each variable is tested on each path at most $k$ times.

ii) A branching program is called $s$-oblivious for a sequence of variables $s = (s_1, \ldots, s_l)$, $s_i \in X_n$, or short oblivious, if the set of decision nodes can be partitioned into disjoint sets $V_i$, $1 \leq i \leq l$, such that all nodes from $V_i$ are labeled by $s_i$ and the edges which leave $V_j$-nodes reach a sink or a $V_j$-node where $j > i$. The length of an $s$-oblivious branching program is the length of the sequence $s$.

Nondeterministic branching programs and randomized branching programs are defined in the obvious way by introducing additional, unlabeled nodes at which nondeterministic or randomized decisions, resp., are taken. An approximating branching program for a Boolean function $f$ with (two-sided) error $\varepsilon$ is a deterministic branching program computing an $\varepsilon$-approximation of $f$, i.e., a function that differs from $f$ on at most an $\varepsilon$-fraction of the inputs.
For nondeterministic read-once branching programs a further generalization of obliviousness can be obtained by restricting the order of variables in such a way that it equals for each input the order of variables for this input performed in a complete given deterministic read-once branching program, i.e., a BP1 where on each path from the source to the sinks all variables are tested. This complete read-once branching program is called graph order and the resulting nondeterministic read-once branching program is called graph-driven. If the graph order is a tree of polynomial size, then it is called tree-driven.

Combining restrictions on the multiplicity of variable tests with the property of obliviousness we obtain oblivious read-once branching programs, better known as OBDDs.

**Definition 2.2.** An OBDD is a branching program with a variable order given by a permutation \( \pi \) on the variable set. On each path from the source to the sinks, the variables at the nodes have to appear in the order prescribed by \( \pi \) (where some variables may be left out). A \( \pi \)-OBDD is an OBDD ordered according to \( \pi \). The \( \pi \)-OBDD size of \( f \) denoted by \( \pi \)-OBDD(\( f \)) is the size of the smallest \( \pi \)-OBDD representing \( f \). The OBDD size of \( f \), sometimes also called OBDD complexity of \( f \), (denoted by OBDD(\( f \))) is the minimum of all \( \pi \)-OBDD(\( f \)).

The size of the minimal \( \pi \)-OBDD representing a Boolean function \( f \) on \( n \) variables, i.e., \( f \in B_n \), is described by the following structure theorem [31].

**Theorem 2.3.** The number of \( x_{\pi(i)} \)-nodes of the minimal \( \pi \)-OBDD for \( f \) is the number \( s_i \) of different subfunctions \( f_{x_{\pi(1)}}=a_1,...,x_{\pi(i)}=a_i,...,x_{\pi(n)}=1 \), essentially depending on \( x_{\pi(i)} \) (a function \( g \) depends essentially on a variable \( z \) if \( g|_{z=0} \neq g|_{z=1} \)).

It is well known that the size of an OBDD representing a function \( f \) depends on the chosen variable order. Since in applications the variable order is not given in advance we have the freedom (and the problem) to choose a good or even an optimal order for the representation of \( f \). In general OBDDs do not have nice algorithmic properties. There are examples known such that \( g_n \) and \( h_n \) are two Boolean functions which have OBDDs of linear size (for different variable orders) but \( f_n \equiv g_n \lor h_n \) has even exponential BP1 size (for an example see, e.g., Proposition 2 in [6]). If a variable order \( \pi \) is fixed, all important operations can be performed efficiently.

SBDDs (shared binary decision diagrams) are an extension of OBDDs that can express multiple functions. An SBDD represents a Boolean function \( f \in B_{n,m} : \{0,1\}^n \to \{0,1\}^m \) by representing simultaneously the output functions \( f_1, f_2, \ldots, f_m \) of \( f \), where the representations for the different coordinate functions \( f_1, f_2, \ldots, f_m \) may share nodes.
2.3 One-way communication complexity and the size of OBDDs

In order to obtain lower bounds on the size of OBDDs one-way communication complexity has become a standard technique (see Hromkovič [22] and Kushilevitz and Nisan [23] for the theory of communication complexity and the results mentioned below).

The main subject is the analysis of the following (restricted) communication game. Consider a Boolean function \( f \in B_n \) which is defined on the variables in \( X_n = \{x_1, \ldots, x_n\} \), and let \( \Pi = (X_A, X_B) \) be a partition of \( X_n \). Assume that Alice has only access to the input variables in \( X_A \) and Bob has only access to the input variables in \( X_B \). In a one-way communication protocol, upon a given input \( x \), Alice is allowed to send a single message (depending on the input variables in \( X_A \)) to Bob who must then be able to compute the answer \( f(x) \). The one-way communication complexity \( C(f) \) of the function \( f \) is the worst case number of bits of communication which need to be transmitted by such a protocol that computes \( f \). It is easy to see that an OBDD \( G \) with respect to a variable order where the variables in \( X_A \) are tested before the variables in \( X_B \) can be transformed into a communication protocol and \( C(f) \leq \lceil \log |G| \rceil \). Therefore, linear lower bounds on the communication complexity of a function \( f \) lead to exponential lower bounds on the size of \( \pi \)-OBDDs where the \( X_A \)-variables are before the \( X_B \)-variables in \( \pi \).

One central notion of communication complexity are strong fooling sets which play an important role in the lower bound proofs later on.

Definition 2.4. Let \( f : \{0, 1\}^{|X_A|} \times \{0, 1\}^{|X_B|} \rightarrow \{0, 1\} \). A set \( S \subseteq \{0, 1\}^{|X_A|} \times \{0, 1\}^{|X_B|} \) is called strong fooling set for \( f \) if \( f(a, b) = c \) for all \( (a, b) \in S \) and some \( c \in \{0, 1\} \) and if for different pairs \( (a_1, b_1), (a_2, b_2) \in S \) at least one of \( f(a_1, b_2) \) and \( f(a_2, b_1) \) is unequal to \( c \).

Theorem 2.5. If \( f : \{0, 1\}^{|X_A|} \times \{0, 1\}^{|X_B|} \rightarrow \{0, 1\} \) has a strong fooling set of size \( t \), the communication complexity of \( f \) is bounded below by \( \lceil \log t \rceil \).

Because of our considerations above, the size \( t \) of a strong fooling set for \( f \) is a lower bound on the size of OBDDs representing \( f \) with respect to a variable order where the variables \( X_A \) are tested before the variables \( X_B \). Because of the symmetric definition of strong fooling sets, \( t \) is also a lower bound on the size of OBDDs representing \( f \) with respect to a variable order where the variables \( X_B \) are tested before the variables \( X_A \). The crucial step to prove large lower bounds on the OBDD complexity of a function is to obtain for all partitions of the variables large lower bounds on the size of fooling sets for subfunctions of the given function (best case communication complexity).
In the rest of this section our aim is to define a function \( f_n \) with large communication complexity which is a main ingredient in our lower bound proof on the OBDD size of the most significant bit of integer multiplication.

First, we take a look at known results about the communication complexity of some popular functions. Let \( \text{EQ}_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \) be defined by \( \text{EQ}_n(a, b) = 1 \) iff the vectors \( a = (a_1, \ldots, a_n) \) and \( b = (b_1, \ldots, b_n) \) are equal. It is well-known and easy to prove that \( C(\text{EQ}_n) = n \). Obviously the same results can be achieved if Alice gets exactly one of the variables \( a_i, \) and \( b_i, \) \( 1 \leq i \leq n. \)

Similar results can be obtained for the functions \( \text{GT}_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \) and \( \overline{\text{GT}}_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \), where \( \text{GT}_n(a, b) = 1 \) iff \( [a]_i > [b]_i \) and \( \overline{\text{GT}}_n(a, b) = 1 \) iff \( [a]_i \leq [b]_i \).

Now, we are ready to define the function \( f_n \in B_{3n} \) on the variables \( a = (a_1, \ldots, a_n), \) \( b = (b_1, \ldots, b_n), \) and \( c = (c_1, \ldots, c_n) \):

\[
f_n(a, b, c) := (\text{EQ}_n(a, c) \land \overline{\text{GT}}_n(a, b)) \lor \text{GT}_n(a, c).
\]

Using case inspection on the distribution of the \( c \)-variables it is not difficult to prove that for a partition, where the \( a \)- and \( b \)-variables are separated, there exists a strong fooling set of size \( 2^n \) for \( f_n \). In other words the communication complexity of \( f_n \) is not smaller than the communication complexity of \( \overline{\text{GT}}_n \) and the distribution of the \( c \)-variables does not simplify the task. The same result can be obtained if Alice gets exactly one of the variables \( a_i, \) and \( b_i, \) for all \( i \in \{1, \ldots, n\}. \) In this case it is not important whether the investigated \( c \)-variables belong to Alice or Bob but whether the considered \( a \)- and \( c \)-variables or \( b \)- and \( c \)-variables are tested together.

3 The middle bit of integer multiplication

In this section we present some results on the OBDD size of the middle bit of integer multiplication. Furthermore, we investigate more general BDD models.

3.1 On the OBDD size of the middle bit of integer multiplication

Bryant’s lower bound of \( 2^{n/8} \) on the OBDD size of \( \text{MUL}_{n-1,n} \) is unsatisfactory since it does not rule out the possibility that 64-bit multipliers can be represented by OBDDs containing only 256 nodes. Since the aim is to use OBDDs for realistic applications one is interested in small constructions or a better lower bound. Introducing a new technique based on universal hashing Woelfel [35] has improved the lower bound considerably to \( 2^{(n/2)/61 - 4}. \) This result implies that any OBDD for 64-bit multiplication needs more than 70 million nodes and the verification
of 128-bit multipliers is infeasible because more than $3 \cdot 10^{17}$ OBDD-nodes are necessary.

The main proof idea in Bryant’s and Woelfel’s lower bound proofs is to show that for every variable order $\pi$ there exists an integer $c \in \{1, \ldots, 2^n - 1\}$ such that the $\pi$-OBDD size of MUL$_{n-1,n}$ is exponential. Bryant has chosen $c$ in such a way that only two input bits of $c$ are set to 1. Therefore, the product of $x$ and $y$ can be seen as the sum of two integers obtained by shifting $x$ in an appropriate way. More precisely, if $y$ is replaced by the binary representation of $c$ and $c = 2^i + 2^{i+d}$ then $c \cdot x = x \cdot 2^i + x \cdot 2^{i+d}$, Woelfel has enlarged the possible choices for the integer $c$. As a result he has been able to prove that for every variable order $\pi$ there exists an integer $c$ such that MUL$_{n-1,n}$ has a large number of subfunctions obtained by replacements of the first $n/2$ $x$-variables in $\pi$ by constants. Summarizing Bryant’s and Woelfel’s lower bound proofs rely only on the existence of a constant factor $c$ for each variable order $\pi$ for which MUL$_{n-1,n}$ leads to a large $\pi$-OBDD representation. If one would like to improve the lower bound there are two possibilities. The first one is to consider multiple values for $c$, the second one to improve the lower bound for the $\pi$-OBDD size of MUL$_{n-1,n}$ for a suitably chosen constant $c$. Woelfel has shown that the latter approach cannot yield significant better lower bounds because the variable order $\pi = (x_0, x_1, \ldots, x_{n-1})$ leads to OBDDs of size at most $3 \cdot 2^{n/2}$ for each integer $c$. By combining this result with the observation that the $k$ most significant bits of one input vector are not important any more if the $k$ least significant bits of the other input vector are known, Woelfel has obtained the first non-trivial upper bound of $(7/3) \cdot 2^{(4/3)n}$ on the size of OBDDs for MUL$_{n-1,n}$ with respect to the variable order $\pi = (y_0, \ldots, y_{n-1}, x_0, \ldots, x_{n-1})$. Amano and Maruoka [3] have improved this upper bound to $2.8 \cdot 2^{(6/5)n}$ for so-called quasi-reduced or complete OBDDs, i.e., OBDDs where on each path from the source to the sinks all variables have to be tested, and the pairwise ascending variable order $\pi = (x_0, y_0, \ldots, x_{n-1}, y_{n-1})$. (It is not difficult to see that the size of a quasi-reduced OBDD can be at most $n + 1$ times larger than the size of a reduced OBDD for a given function $f$ with respect to the same variable order.) Despite the considerable amount of research dealing with the complexity of the middle bit of multiplication, the gap between lower and upper bounds on its OBDD size is still large. Furthermore, even Woelfel’s improved lower bound does not really justify why OBDDs for multipliers of input length $n = 64$ cannot be constructed nowadays using current standard PC hardware. Sauerhoff [26] has shown that the upper bound of Amano and Maruoka [3] is in fact asymptotically optimal for the order chosen by them which is believed to be one of the best ones. For $n = 64$ his bound is larger than $1.62 \cdot 10^{21}$. This surely explains why an OBDD with respect to this variable order cannot be generated. Nevertheless, there is the possibility that there are considerably better variable orders.
3.2 On the size of more general BDD models for the middle bit of integer multiplication

In learning theory and genetic programming OBDDs are used to represent approximations of Boolean functions. Gronemeier [20] has shown that for every variable order $\pi$ the approximation of some output bits of integer multiplication with respect to the uniform distribution and constant error requires $\pi$-OBDDs of exponential size. Nevertheless, approximating the middle bit of integer multiplication with polynomially small error is easy even for read-once branching programs [27].

Although there has been considerable progress in the development of lower bound proofs by the investigation of weakly restricted BDD models, the lower bound methods often only work for a quite limited class of functions. Besides the interest in finding lower bounds as large as possible or proving superpolynomial lower bounds for more and more general BDD models, it is important to apply the existing methods to (practically) important functions. Lower bound proofs for such functions may help to develop new or refined proof techniques, or can lead to new insights into the properties of the considered functions. This is the motivation for the further investigation of the complexity of integer multiplication for more general BDD models.

Methods from communication complexity have been used to prove large lower bounds in several binary decision diagram models. Bryant [12] has used the fooling set method to obtain lower bounds on the communication complexity of the middle bit of multiplication which implies an exponential lower bound of size $2^{n/8}$ for OBDDs representing $\text{MUL}_{n-1,n}$. Incorporating Ramsey theoretic arguments of Alon and Maass [2] and using the rank method of communication complexity Gergov [19] has extended Bryant’s lower bound to arbitrary nondeterministic linear-length oblivious BPs. His lower bound is still non-polynomial for length $o(n \log n / \log \log n)$. Since Woelfel’s larger lower bound on the OBDD size of $\text{MUL}_{n-1,n}$ has not been proved using strong fooling sets his result cannot generalized in the same way as Bryant’s to nondeterministic linear-length oblivious branching programs. In [1] Gergov’s reduction has been applied to deduce that also randomized OBDDs require exponential size and it has been shown that in contrast the graph of integer multiplication $\text{MUL-Graph}_n$ has randomized OBDDs of polynomial size. For the later result the fact has been used that it is easy to verify with small error probability whether the product of two integers equals some given output applying arithmetic modulo a random chosen prime. For non-oblivious models Ponzio [25] has presented the first (weakly) exponential lower bound. He has shown that the complexity of the middle bit of integer multiplication is $2^{\Omega(n^{1/2})}$ for read-once branching programs. In [6] the first exponential lower bound on the size of a nondeterministic non-oblivious read-once branching program model, namely for nondeterministic tree-driven BP1s, has been presented.
An extension of the proof shows that all subfunctions of $MUL_{n-1,n}$ obtained by the replacement of up to $(n/\log n)^{1/2 - \epsilon}$ variables, $\epsilon > 0$ any constant, have exponential size for nondeterministic OBDDs. Since the result also holds for the parity-acceptance mode, where the function value equals 1 for an input iff the number of its accepting paths is odd, this has been the first non-trivial lower bound for an important function on non-oblivious restricted BP1s with an unlimited number of parity nodes.

The fact that integer multiplication defines a universal hash class [14, 15, 36], called multiplicative hash class, has also been used by Bollig and Woelfel [9] to improve the exponential lower bound on the size of BP1s up to $2^{\left\lceil \sqrt{2}n \right\rceil}$ which is even larger than Bryant’s lower bound on the OBDD size. Moreover, the analysis seems to be much easier than the counting technique used by Poncio. At the beginning one reason for the difficulties in proving exponential lower bounds on the size of binary decision diagram models representing $MUL_{n-1,n}$ could have been arisen from the fact that integer multiplication can express many different shifting and adding combinations such that the effect of partial assignments and therefore the subfunctions are not easy to analyze. Using methods which rely on universal hashing it has been shown that even if almost a quarter of the variables of each factor has been replaced by constants, each result of the product bits between the positions $n-1$ and $(3/4)n$ (the results for $MUL_{n-1,n}$ to $MUL_{(3/4)n-1,n}$) is still possible. Using an algebraic approach in [8] a lower bound of $2^{(n^{46}/12)n^{-1}}$ for a restricted nondeterministic BP1 model with parity acceptance mode, called parity graph-driven BP1s, has been shown. This result has been motivated by the fact that until now no superpolynomial lower bound on the size of unrestricted nondeterministic BP1s with parity acceptance mode for an explicitly defined Boolean function has been known. Since exponential lower bounds on the size of unrestricted nondeterministic read-once branching programs which represent $MUL_{n-1,n}$ had been unknown, one step towards proving such bounds was to investigate BP models “inbetween” deterministic and nondeterministic BP1s and a model where some but not all variables may be tested multiple times [10, 37]. Finally, Sauerrhoff and Woelfel [27] have achieved a major breakthrough presenting exponential lower bounds on the size of nondeterministic and randomized BPks for $MUL_{n-1,n}$.

Wegener and Woelfel [34] have considered unrestricted branching programs and Boolean formulas over the basis $B_2$ of all binary operations. Since more than 40 years the best lower bounds for explicitly defined functions are for general branching programs of order $n^2 / \log^2 n$ and for Boolean formulas of order $n^2 / \log n$. These results have been proved with Nechiporuk’s technique [24]. It is well known that this method cannot yield better lower bounds. In [34] the following results have been presented. Any branching program for $MUL_{n-1,n}$ has at least $\Omega(n^{1/2} / \log n)$ nodes and any Boolean formula for $MUL_{n-1,n}$ has size at least $\Omega(n^{3/2})$. Furthermore, it has been proved that using Nechiporuk’s technique it is
impossible to prove better lower bounds than $\Omega(n^{5/3}/\log n)$ and $\Omega(n^{5/3})$ for the branching program and Boolean formula size of $\text{MUL}_{2^n-1,n}$. These are non-trivial limits of Nechiporuk’s technique. Until now it is still an open question whether the lower bound method has been applied in the best possible way in [34].

4 The most significant bit of integer multiplication

Although many exponential lower bounds on the OBDD size of Boolean functions are known and the lower bound methods are simple, it is often a more difficult task to prove large lower bounds for some predefined and interesting functions. The most significant bit of integer multiplication is a good example. Despite the well-known lower bounds on the OBDD size of the so-called middle bit of multiplication ([12], [35]), only recently it has been shown that the OBDD complexity of the most significant bit is also exponential [5] answering an open question posed by Wegener [32]. Here, we start our investigation by an improved lower bound on the size of $\pi$-OBDDs for $\text{MUL}_{2^n-1,n}$, where $\pi$ is a fixed variable order. Using communication complexity the proof is very simple and elementary. Afterwards we present the best general lower bound on the OBDD size for $\text{MUL}_{2^n-1,n}$ known so far and conclude our considerations by the best known upper bound.

4.1 Lower bounds on the OBDD size of the most significant bit of integer multiplication

In this section we start our investigation of lower bounds on the size of OBDDs for $\text{MUL}_{2^n-1,n}$ by presenting a lower bound for some fixed variable order. Afterwards we present a general lower bound which is much smaller but also exponential. The ideas of the general lower bound have been presented in [4].

Using techniques from analytical number theory Sawitzki [29] has presented a lower bound of $2^{n/6}$ on the size of $\pi$-OBDDs representing the most significant bit of integer multiplication for the variable order $\pi$ where the variables are tested according to increasing significance, i.e., $\pi = (x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1})$. Here, we prove a larger lower bound in an easier way and without analytical number theory.

Theorem 4.1. Let $\pi = (x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1})$. The $\pi$-OBDD size for the representation of $\text{MUL}_{2^n-1,n}$ is $\Omega(2^{n/4})$.

Proof. We start with the following two useful observations. For a number $2^n - 1 + \ell2^{n/2}$ the corresponding smallest number such that the product of the two numbers
Proof. We start with a (simplified) presentation of the main proof ideas for a lower bound of \( \Omega(2^{n/4}) \). Furthermore,

\[
2^{n/2} > 4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2} + \ell} \right\rfloor > 4(\ell - 1)^2
\]

for \( 0 < \ell \leq 2^{n/4-1} \). Using these two facts it is not difficult to construct a strong fooling set of size \( 2^{n/4-1} \):

Let \( X_U := \{x_{n-1}, x_{n-2}, \ldots, x_{n/2}\} \), \( Y_U := \{y_{n-1}, y_{n-2}, \ldots, y_{n/2}\} \), \( X_L := \{x_{n/2-1}, x_{n/2-2}, \ldots, x_0\} \), and \( Y_L := \{y_{n/2-1}, y_{n/2-2}, \ldots, y_0\} \). We define \( Z_A := X_U \cup Y_U \) and \( Z_B := X_L \cup Y_L \). The Set \( S \) contains all pairs \( (a, b) \) for \( \ell \in \{1, 2, \ldots, 2^{n/4-1}\} \) with the following properties:

1. \( a \) is an assignment that consists of a partial assignment \( a_i \) to the variables in \( X_U \) and a partial assignment \( a_j \) to the \( Y_U \)-variables where \( \|a_i\| = 2^{n-1} + \ell 2^{n/2} \) and \( \|a_j\| = 2^{n-\ell} 2^{n/2} + 1 \) and

2. \( b \) is an assignment that consists of a partial assignment \( b_i \) to the variables in \( X_L \) and a partial assignment \( b_j \) to the \( Y_L \)-variables where \( \|b_i\| = 0 \) and \( \|b_j\| = 4\ell^2 - \left\lfloor \frac{4\ell^3}{2^{n/2} + \ell} \right\rfloor \).

For all pairs in \( S \) the function value of \( \text{MUL}_{2n-1,n} \) is 1. Let \( (a_1, b_1) \) and \( (a_2, b_2) \) be two different pairs in \( S \). If the value of the partial assignment of the \( X_U \)-variables according to \( a_1 \) is \( 2^{n-1} + \ell_1 2^{n/2} \) and the value of the partial assignment of the \( X_U \)-variables according to \( a_2 \) is \( 2^{n-1} + \ell_2 2^{n/2} \), where w.l.o.g. \( \ell_1 < \ell_2 \), the function value of \( \text{MUL}_{2n-1,n}(a_2, b_1) \) is 0. Therefore, \( S \) is a fooling set of size \( 2^{n/4-1} \).

Because of the symmetric definition of strong fooling sets we also obtain a lower bound of \( 2^{n/4-1} \) on the size of \( \pi' \)-OBDDs for the most significant bit, where \( \pi' = \{x_{n-1}, y_{n-1}, x_{n-2}, y_{n-2}, \ldots, x_0, y_0\} \).

Now, we prove the general lower bound.

Theorem 4.2. The OBDD size for the representation of \( \text{MUL}_{2n-1,n} \) is \( \Omega(2^{n/6}) \).

Proof. We start with a (simplified) presentation of the main proof ideas for a lower bound of \( \Omega(2^{n/6}) \) and present afterwards the idea how to improve this lower bound up to \( \Omega(2^{n/4}) \).

Our aim is to show for an arbitrary variable order \( \pi \) that a \( \pi \)-OBDD for \( \text{MUL}_{2n-1,n} \) contains a \( \pi \)-OBDD for the Boolean function \( f_{\pi'} \) defined in Section 2.3:

\[
f_{\pi'}(a, b, c) = (\text{EQ}_{\pi'}(a, \bar{c}) \land \overline{\text{GT}_{\pi'}(a, b)}) \lor \text{GT}_{\pi'}(a, \bar{c}),
\]

where for each position \( i \) the variables \( a_i \) and \( b_j \) are suitably separated in \( \pi \) and \( n' = \Theta(n) \). Therefore, the size of the \( \pi \)-OBDD for \( \text{MUL}_{2n-1,n} \) has to be large.
vector $a$ is a subvector of one of the inputs $x$ and $y$ for MUL$_{2n-1,n}$, the vectors $b$ and $c$ of the other input.

We use the idea of the following reduction from multiplication to squaring presented by Wegener [33], where squaring computes the square of an $m$-bit input. For two $m$-bit numbers $u$ and $w$ the number $\ell := u \cdot 2^{2(m+1)} + w$ is defined. Then

$$\ell^2 = u^2 \cdot 2^{4(m+1)} + uw2^{2(m+1)+1} + w^2.$$  

Since $w^2$ and $uw$ are numbers of length $2m$, the binary representation of the product $uw$ can be found in the binary representation of $\ell^2$. (Figure 1 shows the bit composition of the number $\ell^2$)

A key observation is the following one. The number $\left\lfloor \frac{4\ell}{2n+1} \right\rfloor$ is smaller than $\ell$ if $\ell \leq 2^{n/2}$. As a consequence if $b_\ell$ is the binary representation of $\ell$, $b_\ell$ is the binary representation of $\ell^2$. $L$ the length of $b_\ell$, and if there exists $j$, where $j \geq L - 2$, and $[b_\ell]_j = 1$, there is no difference in the upper half of the binary representations of the numbers $4\ell^2$ and $4\ell^2 - \left\lfloor \frac{4\ell}{2n+1} \right\rfloor$. More precisely, if $b'$ is the binary representation of $4\ell^2$ and $b''$ is the binary representation of $4\ell^2 - \left\lfloor \frac{4\ell}{2n+1} \right\rfloor$, then $[b']_{j+1} = [b'']_{j+1}$.

Next, we investigate requirements that have to be fulfilled for inputs $x$ and $y$, where MUL$_{2n-1,n}(x,y) = 1$. If $x$ represents a number $2^{n-i} + \ell 2^{n/2}$, $1 \leq \ell \leq 2^{n/2}/2$, the upper half of $y$ has to represent a number of at least $2^{n/2} - 2\ell$, i.e., $[y]_{2n/2} < 2^{n/2} - 2\ell$. If the upper half of $y$ represents a number greater than $2^{n/2} - 2\ell$, the function value MUL$_{2n-1,n}(x,y)$ is 1. Let $j$ be the minimum integer in the set $\{i \mid n/2 \leq i < (3/4)n - 3/2 \text{ and } x_i = 1 \}$. If $[y]_{j+1} > [x]_{j+1}$, the function value MUL$_{2n-1,n}$ is 0. If $[y]_{j+1} < [x]_{j+1}$, the function value MUL$_{2n-1,n}$ is 1. If $y_{j+1} = 1$, $[y]_{j+2} = [x]_{j+2}$, and $[y]_{0/2} = 0$, $[y]_{0/2}$ has to represent a number of at least $4\ell^2 - \left\lfloor \frac{4\ell}{2n+1} \right\rfloor$.

In order to use Wegener’s observation on squaring mentioned above we only consider integers $\ell$ where $\ell = u2^{2(m+1)} + w$, $u, w < 2^n$ and $m = n/12 - 5/6$. (Later
The Bulletin of the EATCS

on we show that \( m \) can be enlarged which leads to a larger lower bound.) For this reason we replace the variables \( x_{0/2m}, \ldots, x_{0/2m+1} \) by 0. (See Figure 2 for the composition of the number \( x \).) Afterwards we replace some of the \( x \)-variables and the corresponding \( y \)-variables by constants, where \( y_{m+1} \) is the corresponding \( y \)-variable to \( x_j \), such that a certain part of \( uw \) is equal to a certain part of \( 2^d \cdot w \) for \( d \) suitably chosen. Furthermore, we choose \( w \) in such a way that the assignments to the variables at position \( 3m + 5, \ldots, 6m + 5 \) are the same in the binary representations of \( 4\ell \cdot 2 \) and \( 4\ell \cdot 2 - \left[ \frac{4\ell \cdot 2}{2^{m+1}} \right] \). Moreover, for different integers \( \ell_1 \) and \( \ell_2 \) (which means different assignments to the \( w \)-variables) the assignments to the variables at position \( 3m + 5, \ldots, (7/2)m + 4 \) in the binary representations of \( 4\ell_1 \cdot 2 \) and \( 4\ell_2 \cdot 2 \) are different. (Figure 3 illustrates some of the replacements of the \( y \)-variables.)

Now we make our proof idea more precise. We rename \( [x_{0/2m}, w_{1/2}] \) by \( [u\ell_{m+1}] \) and \( [x_{0/2m}^{m+1}/4, 3] \) by \( [u_{0/2m}] \). If \( \ell = u \cdot 2^{m+1} + w \) the product \( uw \) can be found at position \( 2m + 5, \ldots, 4m + 4 \) in the binary representation of \( 4\ell \cdot 2 \). The crucial step is to choose an appropriate subset of the input variables in order to show that there exists a large strong fooling set. Let \( S := \{w_{0/2}, \ldots, w_{m+1}, y_{3m+5}, \ldots, y_{(7/2)m+4}\} \) and \( T \) be the set of the first \( |T| \) variables according to \( \pi \), where there are \( m/2 \) variables from \( S \), and \( B \) be the set of the remaining variables. Let \( W_{S,T} \) be the \( w \)-variables in \( S \cap T \), \( W_{S,B} \) the \( w \)-variables in \( S \cap B \). Similar the sets \( Y_{S,T} \) and \( Y_{S,B} \) are defined. Using simple counting arguments we can prove that there exists a distance parameter \( d \) such that there are at least \( m/8 \) pairs \( (w_t, y_{2m+5+iv}) \) in \( W_{S,T} \times W_{S,B} \).
The variables $x_{i}/2^{m}, i \in I$, are called free $x$-variables, the variables $y_{n/2^{m+1}}$ and $y_{2^{m+1}+d}, i \in I$, free $y$-variables. The free $x$-variables will play the role of the $a$-variables, the free variables $y_{n/2^{m+1}}, i \in I$, the role of the $c$-, and the remaining free $y$-variables the role of the $b$-variables in the reduction from the function $f_{p}$ mentioned above to $\text{MUL}_{2n-1,n}$. Now we present the reduction. (Figure 4 shows some of the replacements to the inputs $x$ and $y$ of $\text{MUL}_{2n-1,n}$.)

- The variables $y_{m-1}$ and $x_{n-1}$ are set to 1,
- $X_{n/2^{m+1}-1}$ (which corresponds to $w_{n/2^{m+1}}$ and $y_{n/2^{m+1}}$ are set to 1,
- $X_{n/2^{m+1}}$ (which corresponds to $u_{n/2^{m+1}}$ is set to 1, the corresponding variable $Y_{n/2^{m+1}}+d$ is set to 0, $y_{4m+6}+2d$, $d_{n/2^{m+1}}$, $y_{4m+5}$ to 0 (as a result $y_{4m+5} = 2^d$).
- The variables $y_{n/2^{m+1}}$, $y_{n/2^{m+1}}$ are set to 0.
- Besides the free $x$-variables the remaining $x$-variables are replaced by 0.
- Besides the free $y$-variables the remaining $y$-variables are replaced by 1.

What is the effect of the replacements?

- The inputs $x$ and $y$ represent numbers that are at least $2^{n-1}$, since otherwise the function value $\text{MUL}_{2n-1,n}(x, y)$ is 0.
- Since $w_{n/2^{m+1}} = 1$ and $[u_{n/2^{m+1}}] = 2^d$, $4\ell^2$ and $4\ell^2 - \left\lfloor \frac{4\ell^2}{2^m} \right\rfloor$ where $\ell = u \cdot 2^{m+1} + w$, do not differ in one of the bits at position $3m + 5, \ldots, 6m + 5$ of their binary representations.
- Since $X_{n/2^{m+1}} = 1$ and $Y_{n/2^{m+1}} = 1$, $x_{n/2^{m+1}} = \ldots = x_{n/2^{m+1}} = 0$ and $y_{n/2^{m+1}} = \ldots = y_{n/2^{m+1}} = 0$, $[x]_{n/2^{m+1}}^{16} = \left\lfloor \frac{x}{a_{n/2^{m+1}}} \right\rfloor$, $[x]_{n/2^{m+1}}^{16}$ has to be at least $\left\lfloor \frac{y}{a_{n/2^{m+1}}} \right\rfloor$, $\text{MUL}_{2n-1,n}(x, y) = 1$. If $[x]_{n/2^{m+1}}^{16} > [y]_{n/2^{m+1}}^{16}$, $\text{MUL}_{2n-1,n}(x, y) = 1$.
- Since $[y]_{5\ell+5}^{16} = 2^d + w$ and because of the other replacements, $[y]_{5\ell+5}^{16}$ has to be at least $(u \cdot w) \text{ div } 2^m$ for inputs $x$ and $y$, where $\text{MUL}_{2n-1,n}(x, y) = 1$, if $[y]_{5\ell+5}^{16} = 2^{m/2} - 2\ell$ and $[x]_{5\ell+5}^{16} = 2^{m/2-1} + \ell$. 

\[94\]
Therefore, the correctness of our reduction follows from our considerations above. Considering the fact that \( m = n/12 - 5/6 \), we get the result that the OBDD complexity of \( \text{MUL}_{2n-1,n} \) is at least \( \Omega(2^{n/96}) \).

Finally, we present the idea how to improve the lower bound on the OBDD complexity of \( \text{MUL}_{2n-1,n} \) up to \( \Omega(2^{n/60}) \). Up to now we have considered numbers \( \ell \), where \( \ell = u \cdot 2^{2(n+1)} + w \) and \( u, w < 2^m \) with \( m = (n/12) - 5/6 \). Using the fact that in our lower bound proof only the upper half of the bits in the binary representation of \( uw \) is important, \( uw \) div \( 2^{(3/2)m} = 0 \), \( u^2 \) div \( 2^{(7/4)m} = 0 \), and \( u^2 \mod 2^{m/4} = 0 \), we can choose \( \ell = u \cdot 2^m + w \), \( w < 2^m \) and \( u < 2^{(7/8)m} \). As a result we can enlarge \( m \) up to \( (2/15)n \).

\[ \square \]

**4.2 Upper bounds on the OBDD size of the most significant bit of integer multiplication**

In this section we prove an upper bound on the size of OBDDs according to an arbitrary variable order representing the most significant bit of integer multiplication. Afterwards we present the best known upper bound on the size of OBDDs representing \( \text{MUL}_{2n-1,n} \) according to the variable order \( \pi = (x_{n-1}, y_{n-1}, x_{n-2}, y_{n-2}, \ldots, x_0, y_0) \). The results of this section have been presented in [7].

We start our proof of the general upper bound with the investigation of a function \( f \) that is closely related to the most significant bit of integer multiplication. Let \( n \in \mathbb{N} \) be arbitrary but fixed in the rest of the section.
Lemma 4.3. Let $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ be defined as $f(x) := \frac{2^{2n-1}}{c + x}$. For arbitrary $\Delta x, \Delta y > 0$ there exists exactly one value $x \in \mathbb{R}^+$ with $f(x) - f(x + \Delta x) = \Delta y$.

In other words each distance pair $(\Delta x, \Delta y)$ defines uniquely two elements in the definition set.

Next, we consider some modifications of the function $f$.

Definition 4.4. For $c, d \in \mathbb{R}$ and $n \in \mathbb{N}$ we define the function $f_{c,d} : \mathbb{R} \rightarrow \mathbb{R}$ in the following way.

$$f_{c,d}(x) := \frac{2^{2n-1}}{c + x} - d.$$

The function $f_{c,d}$ contains the tuple $(x, y)$ iff $f_{c,d}(x) = y$.

Our proof idea of the upper bound on the size of OBDDs representing $MUL_{2n-1,n}$ is to use the functions $f_{c,d}$ in order to analyze the number of different subfunctions of $MUL_{2n-1,n}$ obtained by replacements of some $x$- and $y$-variables by constants. For this reason we have to relate the functions $f_{c,d}$ to subfunctions of $MUL_{2n-1,n}$.
Clearly there can be several functions $f_{c,d}$, and the length of the numbers could be large. In order to find a small representation for $f_{c,d}$, it is easy to see that each function $f_{c,d}$ can be characterized by two tuples $(x_i,y_i)$ and $(x_i,y_i)$, where $f_{c,d}(x_i) = y_i$ and $x_i,y_i \in \mathbb{R}$ for $i \in \{1,2\}$. Unfortunately, the length of the numbers could be large. In order to find a small representation for $f_{c,d}$ we modify $f_{c,d}$ without changing essentially the corresponding step function.

We start to analyze the effect of moderate modifications of the parameters $c$ and $d$.

**Definition 4.5.** For a given function $f_{c,d}$ and two arbitrary finite sets $A,B \subseteq \mathbb{N}$ and $A,B \neq \emptyset$ the corresponding step function $f^{A,B}_{c,d} : A \to B$ is defined as

$$f^{A,B}_{c,d}(x) := \min\{y \in B | y \geq f_{c,d}(x)\}.$$

The rotation operation is finite.

**Lemma 4.6.** Let $c,d \in \mathbb{R}^+$ and $A,B$ be two arbitrary finite, nonempty subsets of $\mathbb{N}$. Let $y_i$ be the largest element in $B$ that is smaller than $f_{c,d}(x)$ and $\epsilon_i := (y_{i+1} - y_i)$, if $y_i$ is defined, otherwise $\epsilon_i := \infty$. We define $\epsilon_{\min} := \min\{\epsilon_i | x \in A\}$. Then $f^{A,B}_{c+d,\epsilon_{\min}/2,d} = f^{A,B}_{c,d}$.

**Lemma 4.7.** Let $c,d \in \mathbb{R}^+$ and $A,B$ be two arbitrary finite, nonempty subsets of $\mathbb{N}$. For $x \in A$ let $\epsilon_i := f^{A,B}_{c,d}(x) - f_{c,d}(x)$ if $f^{A,B}_{c,d}(x)$ is defined and $\infty$ otherwise. We define $\epsilon_{\min} := \min\{\epsilon_i | x \in A\}$. Then $f^{A,B}_{c+\epsilon_{\min},\epsilon_{\min}} = f^{A,B}_{c,d}$.

Lemma 4.7 tells us that it is allowed to move the graph of the function $f_{c,d}$ upwards, right until it hits its corresponding step function for the first time, without changing the step function.

**Lemma 4.8.** Let $c,d \in \mathbb{R}^+$ and $A,B$ be two arbitrary finite, nonempty subsets of $\mathbb{N}$, such that there exists at least one element $x_0 \in A$ where $f_{c,d}(x_0) = f^{A,B}_{c,d}(x_0)$, and there are at least two elements $x_1,x_2 \in A$ where $\min\{y \in B | y \leq f_{c,d}(x)\} \leq \max\{y \in B, i \in \{1,2\}\}$. We define the following rotation operation for $f_{c,d}$ with respect to $(x_0,y_0)$: decrease $c$ continuously to $c'$ and adjust $d$ to $d'$ at the same time such that $f_{c,d}(x_0) = f_{c',d'}(x_0)$ is always fulfilled until there exists another element $x' \in A$ with $f_{c,d}(x') \in B$.

1. The rotation operation is finite.

2. The function $f^{A,B}_{c,d}$ can be reconstructed from $f_{c,d}$ in the following way:

$$f^{A,B}_{c,d}(x) = \begin{cases} \min\{y \in B | y \geq f_{c,d}(x)\}, & \text{if } x \leq x_0, \\ \min\{y \in B | y > f_{c,d}(x)\}, & \text{if } x > x_0. \end{cases}$$
If we replace \((c, d)\) to \((c', d')\), the curve of the function \(f_{c,d}\) seems visually to rotate to the graph of the function \(f_{c',d'}\), because point \((x_0, y_0)\) stays on the graph, whereas all points left of \(x_0\) are shifted upwards and the other ones downwards. Nevertheless, the graph’s shape does not change since the rotation can be decomposed to a vertical and a horizontal movement (see Figure 5). Therefore, it is still possible to use Lemma 4.3 for the identification of \(f_{c',d'}\).

Now we are able to prove our general upper bound on the \(\pi\)-OBDD size for \(\text{MUL}_{2^{n-1},n}\).

**Theorem 4.9.** Let \(\pi\) be an arbitrary variable order. The \(\pi\)-OBDD size for the representation of \(\text{MUL}_{2^{n-1},n}\) is \(\Theta(2^{n/3})\).

**Proof.** Our aim is to prove an upper bound of \(2^{2(n-i)/(2(n-j)+2)}\) on the number of subfunctions of \(\text{MUL}_{2^{n-1},n}\) obtained by replacements of \(i\) \(x\)- and \(j\) \(y\)-variables by constants.

We assume in the following that \(i \neq 0\) and \(j \neq 0\), since otherwise we are done.

If \(i = n\) an upper bound of \(2^{-j}\) is easy to prove and we are done, similarly an upper bound of \(2^{-i}\) can be shown for \(j = n\). Therefore, we also assume in the following that \(i \neq n\) and \(j \neq n\). Let \(X^S\) be the set of \(i\) arbitrary \(x\)-variables and \(Y^S\) be the set of \(j\) arbitrary \(y\)-variables, \(X_T := \{x_0, \ldots, x_{n-1}\}\backslash X^S\), and \(Y_T := \{y_0, \ldots, y_{n-1}\}\backslash Y^S\).

\(\text{MUL}_{2^{n-1},n}\) answers the question, whether for a given assignment \((a, b)\) of the variables, the product \(\langle a \rangle \cdot \langle b \rangle\) is at least \(2^{2i-1}\). Therefore, the function \(\text{MUL}_{2^{n-1},n}\) can be described by specifying for every possible assignment \(a\) of the \(x\)-variables, the assignment \(b\) of the \(y\)-variables with \(\langle b \rangle = \left\lceil \frac{2^{2i-1}}{2^{n-j}} \right\rceil\). Figure 6 shows \(\text{MUL}_{2^{n-1},n}\), where for a value \(\langle a \rangle\) the smallest corresponding value \(\langle b \rangle\) that fulfills \(\text{MUL}_{2^{n-1},n}\) is dotted. Such pairs of assignments are called significant points. (For sake of simplicity the possible values are at least \(2^{n-1}\) because for smaller numbers the product cannot be at least \(2^{2i-1}\).)

Let \(c := \|a_{X_T}\|\) and \(d := \|b_{Y_T}\|\). We define \(A_T\) as the set of possible values \(\|a_{X_T}\|\) that can be expressed by the variables from \(X_T\). Let \(B_T\) be defined in the same way. \(A_T\) and \(B_T\) are independent of the choice of \(c\) and \(d\), i.e., a grid can be defined for the \(\|a_{X_T}\|\) and \(\|b_{Y_T}\|\) values, which has the same appearance for all possible assignments \(c\) and \(d\). A subfunction of \(\text{MUL}_{2^{n-1},n}\) obtained by replacing the variables in \(X^S\) to \(a_S\) and \(Y^S\) to \(b_S\) can be described by the pairs of \(A_T\) and \(B_T\)-values \((\|a_{X_T}\|, \|b_{Y_T}\|)\), so that \(\|b_{Y_T}\|\) is the minimal value that fulfills \(\|b_{Y_T}\| \geq \frac{2^{2i-1}}{c^j \|a_{X_T}\|} \cdot d\). Therefore, the subfunction of \(\text{MUL}_{2^{n-1},n}\) can be characterized by the step function \(f_{c,d}^{A_T,B_T}\) (see Definition 4.5) for the underlying function \(f_{c,d}\).

Figure 7 shows an example for two different step functions that result from two different assignments to the variables in \(X^S \cup Y^S\).

Since the subfunctions obtained by replacing the variables in \(X^S\) and \(Y^S\) by constants can uniquely be described by their step functions, our aim is to prove...
the existence of a small representation such that the corresponding step function and therefore the corresponding subfunction of $\text{MUL}_{2n-1,n}$ can be reconstructed later on. As each representation implicates at most one possible step function, the number of different representations is an upper bound on the number of different subfunctions.

The idea is to transform the function $f_{c,d}$ in a moderate way into a function $f_{c,d}'$, such that $f_{c,d}'$ contains at least two points from $A_T \times B_T$ and the step function $f_{c,d}' \mid A_T \times B_T$ can easily be obtained from $f_{c,d}$. In the following we assume that for at least two $A_T$-values, the function value $f_{c,d}$ is greater than 0 and smaller or equal to the greatest value in $B_T$. The other cases will be considered later on. If $c$ equals 0, we have to make some extra considerations. Since the function $f_{c,d}$ is not defined for the value $\|a_X\| = 0$, we use Lemma 4.6 to move the graph a tiny distance to the left. As a result we obtain the function $f_{c,d}$ and $f_{c,d}' \mid A_T \times B_T = f_{c,d} \mid A_T \times B_T$.

According to Lemma 4.7 the graph is moved upwards by decreasing the parameter $d$, right until the graph cuts the graph of its step function. Let $f_{c,d}'$ be the resulting function and $f_{c,d}' \mid A_T \times B_T$ its step function. Obviously $f_{c,d}' \mid A_T \times B_T = f_{c,d} \mid A_T \times B_T$. We now have at least one element $p_1 \in A_T$, so that $f_{c,d}'(p_1) = f_{c,d}' \mid A_T \times B_T(p_1) = q_1$.

If $f_{c,d}'$ contains another point $(p_2, q_2) \in A_T \times B_T$, we can be sure that $q_2$ is
not equal to \( q_1 \) because the function is strictly monotonic. In this case we stop the transformation and encode the step function \( f_{c',d'} \) by the triple \( ((p_1, q_1), (p_2, q_2), 1) \) where the last bit indicates that we stopped at this point.

Otherwise we modify the function \( f_{c',d'} \) again to hit a second point of \( A_T \times B_T \). Using Lemma 4.8 the graph is rotated clockwise by decreasing \( c' \) and adjusting \( d' \), so that the point \( (p_1, q_1) \) stays on the graph. We get a new function \( f_{c'',d''} \) and another point \( (p_2, q_2) \in A_T \times B_T \) with \( f_{c'',d''}(p_2) = q_2 \).

Now we have achieved that the function \( f_{c'',d''} \) contains two tuples \( (p_1, q_1) \) and \( (p_2, q_2) \) that can be addressed by the variables in \( X_T \cup Y_T \). The distance between these points is independent of the assignment to the variables in \( X_S \cup Y_S \). In order to apply Lemma 4.3 we have to be sure, that \( (p_1, q_1) \) and \( (p_2, q_2) \) can be used to identify a shifted cutting of the initial graph \( 2n^{-1} \), i.e., \( 2n^{-1} \rightarrow 2n^{-1} - d'' \), with positive numbers in the denominator. The modification of \( d \) is not critical, because it does not have any influence on the denominator. For the values \( c \) we assure at the beginning that \( c \) is greater that 0 (either because \( c = \|a_{S_i}\| \) is greater than 0 or by using \( \epsilon_{min}/2 \)). Just the rotation operation decreases \( c \). But as we continuously check, whether the value of \( f_{c'',d''} \) hits a point in \( A_T \times B_T \), it is impossible that the
function's pole will be translated across any point of the grid. Therefore, Lemma 4.3 can be used to identify the underlying function $f_{c',d'}$ with $(p_1, q_1)$ and $(p_2, q_2)$.

Our last step is now the reconstruction of the original step function $f_{cT, dT}$. If we have just moved the graph upwards without rotating it, then for every $x \in A_T$ the corresponding value of the step function $f_{cT, dT}^{hT}$ is the smallest value of $B_T$ that is at least $f_{c',d'}(x)$. In the other case we can use the second statement of Lemma 4.8 to reconstruct the original step function. Figure 8 illustrates the reconstruction of the step function $f_{cT, dT}^{hT}$.

As we have seen a triple that consists of two points and an additional bit can encode any possible step function that itself represents a subfunction of $\text{MUL}_{2^n-1,n}$ obtained by replacing $i$-x- and $j$-y-variables by constants. As this subfunction can by uniquely reconstructed by this representative, there cannot be two different subfunctions with the same representation. The maximal number of these representation is

$$\frac{2^{2n-2i-2n-2j}}{(p_1, q_1)} \cdot \frac{2^{2n-2i-2n-2j}}{(p_2, q_2)} \cdot \frac{1}{\text{bit} z} = \frac{2^{2n-2i-2(n-1)+1}}{2^{2n-2i-2(n-1)+1}}.$$  

Up to now we have assumed that for at least two $A_T$-values the function $f_{c,d}$ is
greater than 0 and smaller or equal to the greatest value in $B_T$. A subfunction that
is not of this type can be characterized by only one point $(p, q)$ of the step function
$f_{e, c, d}$. Summarizing there are less than $2^{2(n-i+j/2)+j/2}$ different subfunctions.

Obviously there are at most $2^{i+j}$ different subfunctions obtained by the re-
placement of $i + j$ variables by constants. Using the minimum of the two upper
bounds for each layer we obtain the result that the $\pi$-OBDD size for $\text{MUL}_{2n-1,n}$ is
$O(2^{(4/3)n})$ for any variable order $\pi$.

Combining Theorem 4.9 with an upper bound of $2^{i+j}$ on the number of
subfunctions obtained by the replacement of the variables $x_{n-1}, \ldots, x_{n-i}$ and
$y_{n-1}, \ldots, y_{n-i}$ by constants presented in [3], we get the following result.

**Corollary 4.10.** Let $\pi = (x_{n-1}, y_{n-1}, x_{n-2}, y_{n-2}, \ldots, x_0, y_0)$. The $\pi$-OBDD size for
the representation of $\text{MUL}_{2n-1,n}$ is $O(2^{(4/5)n})$.

### 4.3 More general models and the most significant bit of integer
multiplication

Similar to the results presented in [19] for the middle bit of integer multiplication
the lower bound on the OBDD size of the most significant bit can be extended to
arbitrary oblivious binary decision diagrams of linear length. The complexity of
$\text{MUL}_{2n-1,n}$ for more general non-oblivious models than OBDDs is open.

Intuitively the most significant bit of integer multiplication seems to be much
easier than the middle bit. Using the same proof method as described by Wegener
and Woelfel [34] it can be shown that it is impossible to prove a better lower
bound than $\Omega(n^{1/2} / \log n)$ and $\Omega(n^{1/2})$ for the branching program and Boolean
formula size of the most significant bit using Nechiporuk’s technique. Until now
non-trivial lower bounds for the branching program and Boolean formula size are
unknown.

### 5 A comparison between the middle and the most
significant bit of integer multiplication

In this section we finish our considerations with a brief comparison between the
functions $\text{MUL}_{n-1,n}$ and $\text{MUL}_{2n-1,n}$. For the most significant bit replacing a con-
stant number of variables by constants may lead to a constant subfunction but for
the middle bit we can replace almost an arbitrary quarter of the variables for each
factor by constants without obtaining a constant subfunction. The best known
variable order for the most significant bit is $\pi = (x_{n-1}, y_{n-1}, x_{n-2}, \ldots, x_0, y_0)$ and

102
The Bulletin of the EATCS

the $\pi$-OBDD size of $\text{MUL}_{2n-1,n}$ is $O(2^{4/5n})$. For the middle bit of integer multiplication the best known variable order is $\pi' = (x_0, y_0, x_1, \ldots, x_{n-1}, y_{n-1})$ and the $\pi'$-OBDD size of $\text{MUL}_{2n-1,n}$ is $\Theta(2^{6/5n})$. For each variable order there exists an assignment $c$ to the variables of one factor such that the corresponding OBDD size of $\text{MUL}_{c,2n-1,n}$ is $\Omega(2^{n/2})$. In contrast it is not difficult to prove that for each variable order the corresponding OBDD size of $\text{MUL}_{2n-1,n}$ is $O(n^2)$ for each assignment $c$. For the middle bit also large lower bounds for more general BDD models are known whereas exponential lower bounds for non-oblivious models are unknown for the most significant bit.

Conclusion

We have already learned in primary school how to multiply integers, nevertheless, the complexity of integer multiplication is a fascinating subject. Here, we have tried to deepen the knowledge on the set of subfunctions of the most significant bit of integer multiplication in order to obtain the best lower and upper bounds on its OBDD size.

Acknowledgments

I would like to thank Stefan Droste for proofreading a preliminary version of this survey.

References


The Bulletin of the EATCS


One of my favourite talks at the IFIP TCS conference held in Milan in September 2008 was a presentation by Julian Rathke on his joint project with Paweł Sobociński. The aim of this ongoing research effort is to contribute to the development of a general theory for the systematic derivation of a labelled transition semantics for a process calculus from a simpler, non-structural, description of the reduction semantics. Immediately after the conference I invited Julian and Paweł to contribute to the Concurrency Column by offering its readership an accessible account of the status of their work on this very interesting topic and was thrilled by their willingness to write such a piece. The article you are about to read is the result of their efforts and presents an overview of their approach that I trust most readers will enjoy as much as I have done.

I take this opportunity to remind the readers of the Concurrency Column that CONCUR 2009, which will be held in Bologna in the period 1–4 September 2009, promises to be a truly outstanding scientific event. It will be an apt celebration for the 20th birthday of the conference series and I trust that it will be very well attended.
1 Introduction

One of the characteristics of the process calculus approach to studying concurrent systems is the heavy use of operational semantics. This approach has in fact been dominated by the use of labelled transition systems and structural operational semantics [8]. This isn’t too surprising given that the core of concurrency theory lies in understanding interaction; labels rather neatly describe the nature of interaction contributed by each participant. The earlier work on structural operational semantics had no emphasis on labelled transitions [8] and these were largely developed by Milner in order to provide a semantics for CCS [6].

A rather interesting artefact of using labels to define a structural operational semantics is that the labels allow (partial) programs to be compared for equivalence without the need for examining their behaviour in all enclosing contexts. This is because the labels themselves describe the interactions that the (partial) programs make. However, this use of labels is not necessarily what they were designed for and their dual purpose nature can have an unfortunate side-effect in that the labelled transitions which are really designed to structurally define reduction for a language, may yield too fine an equivalence when considering bisimilarity or traces over these labels. This is because structural labels do not always correspond to observable properties of the underlying programming language. We see this in evidence already for languages such as the asynchronous π-calculus [3, 1].

So, it seems that care must be taken when defining a structural operational semantics for one’s new favourite process calculus if one wants to avoid creating too fine an equivalence. Given the proliferation of new favourite process calculi it is clearly worthwhile to have a means by which we can guarantee some properties of the structural operational semantics. In particular, we would like to know that we have defined reduction well. We would also like to know that our notion of equivalence generated from our labelled transition system is natural in some sense. One approach to getting these guarantees, and the approach we ad-
vocate, is to systematically derive a labelled transition semantics from a simpler, non-structural, description of the reduction semantics. This basic idea is not ours as early efforts to do this can be found in [13, 4]. What we present here is our own, ongoing, attempts at solving this problem. The driving philosophy behind [13, 4] which we adopt is that labels should represent, or should be witnessed by, language contexts. In particular, given a process term, consider what contexts that term can be placed in to induce a reduction. The ‘minimal’ such contexts should be considered as labels. This general philosophy is appealing but Sewell’s early results were limited in scope. Leifer and Milner generalised the approach reasonably well by using categorical constructs to identify a suitable notion of minimality. What these approaches lack however is a structurally inductive presentation of the labelled transitions and hence reduction. This is disappointing to us because the initial motivation for labelled transitions is to provide exactly this. Without a structurally defined semantics we lose the compositionality property that the meaning of a process is a function of the meaning of its subterms.

We believe that one needs to take account of the actual term structure in the axioms for the underlying reduction relation to obtain structurally defined, contextually witnessed, labelled transition systems; that is the core of our contribution. By focusing on the axioms rewrite rules themselves as the generators for the labelled transition system we also direct the focus away from the thorny issue of minimality of contexts. Effectively, if the axioms for reduction are specified in a minimal way, then our analysis will generate the minimal contexts as labels.

In this article we describe our approach to the problem by giving a rough overview of the approach and by showing the case studies we have followed. This article does not present the full technical details but is intended to give more of a flavour of the work. We describe how, given a reduction semantics for a language, one goes about deriving a labelled transition system which can be used both as a structurally inductive definition of reduction, and a basis for equivalence. This involves a decomposition of reductions in to the roles of process and context - our development follows this idea by designing labelled transition systems which represent those roles. We make use a simply-typed λ-calculus as a meta-language for manipulating term contexts. This is a technical convenience but also highlights well the role that meta-syntax plays in our development.

2 Overview of the approach

In this section we will give a sketch of our proposed derivation procedure in order to envisage how a general derived lts could be obtained from a starting point of program syntax and reduction rules.

To exemplify the points we will use a running example of one of the rewrite
rules from the ambient calculus of Cardelli and Gordon [2]. This provides a nice example of the structural rewrites which occur in reduction rules which need to be accounted for in the labelled transition system. Interestingly, before [10], providing a labelled transition system for this calculus had proved a challenging task and the state of the art solution [5] had been somewhat ad-hoc, albeit clever, in its design.

To begin with we assume a source language grammar and a simple typing relation for which well-formed programs are given the type $\text{Pr}$. There may be other sorts/types used to describe this relation. We also assume a reduction relation between well-formed programs which describes computation in the source language. We assume that this reduction relation is defined using a number of axioms that describe basic rewrites between terms. These rewrites may then be embedded within larger contexts to define the reduction relation fully. The presentation of the reduction relation may also involve an auxiliary relation of structural equivalence $\equiv$, which allows syntactic manipulation of terms which are invariant across the reduction relation. This auxiliary relation facilitates simpler presentations of reduction. In addition to this data, the language should come equipped with a notion of evaluation context. To describe this, we define the notion of context more precisely:

**Definition 1** (Precontext). Syntactically, precontexts are generated by the grammar obtained by adding a $\sigma$-annotated hole $-\sigma$ for each type $\sigma$ and a constructor for $n$-tuples (for any $n \in \mathbb{N}$) to the source grammar:

$$M ::= \ldots | -\sigma | (M, \ldots, M),$$

In order to type precontexts, we add two additional type rules to extend the assumed simple type relation:

$$\Gamma \vdash [\vec{\tau}] : \vec{\sigma} \quad \Gamma \vdash (V_1, \ldots, V_n) : [\vec{\sigma}],$$

where $[\vec{\sigma}]$ is called an *interface type*. A precontext is then a typeable term of the form $(V_1, \ldots, V_n)$.

**Definition 2** (Context). Suppose that a precontext $(\forall \vec{\tau})[\vec{\sigma}]$ contains $m$ instances of type-annotated holes. A 1-1 enumeration of its holes with natural numbers from 1 to $m$ uniquely determines a word $\vec{i}$ over types, where $\tau_i$ is the type of the $i$th-numbered hole. Syntactically replacing each hole with its number yields a context of type $[\sigma] \to [\rho]$. Ordinary terms of type $\text{Pr}$ will be identified with contexts of type $[] \to \text{Pr}$. Given contexts $f : [\sigma] \to [\rho]$ and $g : [\rho] \to [\sigma]$, there is a context $g \circ f : [\sigma] \to [\sigma]$ which is obtained by substitution of the $i$th component of $f$ for the $i$th hole of $g$. Moreover:
• a language context is a context of type $[P_r] \rightarrow [P_r]$. These will be denoted by $C$;

• evaluation contexts are contexts of type $[P_r] \rightarrow [P_r]$ that are specified as part of the input data to the problem. They represent those contexts in which reductions may freely take place. We will denote these by $D$;

• an interaction context is a particular form of evaluation context which is to be identified by our analysis. They should represent those evaluation contexts that do not inhibit any observations. We will denote these by $E, F$;

For example, in the ambient calculus the evaluation contexts are those built from the parallel, restriction and boxing operators. Reduction is allowed in all such contexts. However, the interaction contexts are those built from just the parallel and restriction operators. This is because the boxing operator may inhibit certain observations. Technically, this is observed by the failure of commutativity of parallel compositions with boxing in the structural equivalence.

To denote substitution of a term $M$ in a language context $C$ we will often write $C \triangleright M$ instead of $C \circ M$. A relation $R$ on terms is said to be a congruence if $MRN$ implies $C[M] \triangleright RC[N]$ for all language contexts $C$.

Now, the novelty in our approach as compared with previous attempts at this problem [13, 4] lies in the use of the structure of terms involved in the reduction relation to derive structural labelled transition systems. So rather than just describing reduction rewrite rules as relations between closed programs of the language, we consider the structural rewrites involved in reduction. In order to do this we analyse the axioms associated with the reduction relation and express them as skeletons. A skeleton is actually a pair of contexts $(l_\alpha n, r_\alpha n)$ that describe the structural changes in passing from $l_\alpha n$ to $r_\alpha n$. Each reduction axiom yields a skeleton. For example, the ambient calculus has a reduction axiom for the ‘out’ prefix of the language, which allows an ambient to exit from a containing ambient. The axiom is:

$$n[\text{out } n \parallel P \parallel R] \rightarrow n[P] \parallel n[R]$$

(\text{out})

To rewrite this as a rule skeleton we need a pair of contexts which describe the structural changes:

$$(l_{\text{out}} n \overset{\text{def}}{=} n[1_N \text{ out } n \parallel 2_{P_r} \parallel 3_{P_r} \parallel 4_{P_r}], r_{\text{out}} n \overset{\text{def}}{=} 1_N[2_{P_r} \parallel 3_{P_r} \parallel n[4_{P_r}])$$

Essentially, we are replacing the parameters in the rule axioms with numbered holes. Note that the parameter $n$ is not replaced though. This particular parameter appears non-linearly in the LHS. This non-linearity is important because we see that a decomposition of the LHS term in to a program and context could result in this
parameter appearing in each part. We need to ensure that any instantiation of this parameter is consistent across such decompositions. For this reason, we leave the parameter instantiated in the skeletons themselves and record such instantiations in the transition labels.

To move from rule skeletons to labelled transitions, we take three steps. First, we identify the ‘program view’ transitions. These are defined inductively on the program structure by considering how the program can be (partially) matched to the λs of a rule skeleton. Next, we define the ‘context view’ transitions, in which the context is allowed to instantiate any missing parameters which have not been matched in the skeleton. Finally, the ‘combined view’ allows us to compose the two previous views.

In order to present our labelled transition systems we will use a meta-syntax for simple syntactic manipulation of terms. The meta-syntax is a simply typed λ-calculus and can be thought of as a primitive system of higher order abstract syntax [7]. In (1) below we extend the base types with function types that will be necessary in order to type terms in the meta-syntax.

\[ σ ::= \ldots | σ → σ \] (1)

The λ-calculus operators added to the source language grammar below constitute the syntactic aspect of the meta-language. Their function is solely to make the structural definition of a labelled transition system possible and they should not be considered as a language extension and have no computational meaning. They are to be thought of as meta-operators on syntactic phrases of the language.

\[ M ::= \ldots | λX : σ. M \mid M(M) \] (2)

In the above, λ-abstraction binds variables and we do not distinguish α-equivalent terms.

Terms in the metalanguage are typed with aid of the standard type rules for simply typed λ given in (3) below to be used in conjunction with the given type rules for the language.

\[
\frac{Γ, X : σ \vdash M : σ'}{Γ \vdash \lambda X : σ. M : σ → σ'} (\lambda) \quad \frac{Γ \vdash M : σ → σ' \quad Γ \vdash N : σ}{Γ \vdash M(N) : σ'} (\text{App})
\] (3)

We quotient the terms of the meta-syntax by the smallest congruence that contains (4) and (5). Substitution is the usual capture-avoiding notion – it is capture avoiding with respect to all binders even those of the underlying language.

\[ (\lambda X : σ. M)(N) = M[N/X] \] (4)

\[ λX : σ. M(X) = M \] (5)

112
In the remainder of the paper, when we write a meta-syntax phrase of type, say \( Pr \), we mean \textit{the syntactic phrase that corresponds to the complete evaluation of the meta-syntactic term}. This technique is very useful because it avoids many bureaucratic difficulties of binding scopes.

### 2.1 Program-view transitions

In order to define the program view transitions we need to understand what it means for a program to match a context: by considering a context as a syntax tree, a match is a structurally equivalent (\( \equiv \)) subtree of this whose root has type \( Pr \). Note that this subtree will clearly have a subset of the numbered holes of the whole context. We are particularly interested in matches which are \textit{minimally active}. This refers to those matches which are minimal with respect to the subtree relation amongst those in the LHS of a skeleton which do not exist as matches in the RHS of that skeleton. Intuitively, the minimal active matches are the smallest parts of the LHS of the skeleton which are structurally transformed when moving to the RHS. Note that in general, there may be several minimal active matches for any given reduction axiom.

For our running example of the 'out' rule for the ambient calculus, there is a single minimal active match:

\[
\text{out } \frac{\text{n.2}}{\lambda x.\{ P \parallel x \parallel \text{Y} \}}
\]

this is active because this subtree does not appear in the RHS of this skeleton. It is minimal because the only subtree of this tree is the hole \( 2_{Pr} \) itself but this subtree is not active as this hole appears in the RHS of the skeleton.

Minimal active matches determine for us the base cases for the inductive definition of our program view transitions. To see why this should be the case, suppose we have a term of type \( Pr \) which is the instantiation of a minimal active match. Then, we know that there must be a context in to which we place this term to cause a reduction. Moreover, the context must interact with the term in some way because the match is active. Minimality tells us that this is won’t be true for any subterms of the program term.

So we begin with the labelled transition axiom:

\[
\text{out } P \longrightarrow^{\text{out} \cdot \frac{\text{n.2}}{\lambda x.\{ P \parallel x \parallel \text{Y} \}}}
\]

where the LHS of the transition is an instantiated minimal active match and the RHS is the RHS of the 'out' rule skeleton \( r_{out}^{\text{n}} \) whose parameters are partially instantiated with those subterms provided by the program and whose remaining parameters are abstracted using the lambda meta-language. The label decorating this transition is chosen to reflect the skeleton used to generate this axiom. The particular instantiation of the non-linear parameter \( n \) is recorded in this label also.
We must now consider how to inductively build upon this axiom using the larger structure of the program term. This is done by considering how the program term matches to the LHS of the skeleton. We have identified three different relevant situations:

(i) a *substructural* modification: the added structure does match part of the skeleton but in such a way that the added structure can be partially matched to a parameter not in the minimal active match. In this case the label of the transition remains unchanged because the context representing the remainder of the LHS of the skeleton remains unchanged. To see an example of this kind of modification consider the rule for extending ‘out’ transitions in a parallel context:

\[
\begin{array}{c}
\text{P} \xrightarrow{\text{out}} T \\
\text{P} Q \xrightarrow{\text{out}} \lambda X. T(QX)
\end{array}
\]

Here, by associativity of parallel composition in the structural equivalence for ambients, the program \( Q \) can partially match the typed hole 3\( \alpha \) of the LHS of the skeleton. The added structure is passed to the appropriate parameter in the LHS of the transition.

(ii) a *superstructural* modification: the added structure of the program term does not match directly into the LHS of the skeleton, but by structural equivalence, the added structure can be commuted to become an evaluation context in which reduction can take place. That is, the added structure allows, but does not take part in the reduction. Therefore the program match and therefore the label, remain unchanged. The added structure is added to the LHS of the transition at top level.

For example, consider how the ‘out’ rule of the ambient calculus is enlarged in a \( \nu m \) context:

\[
\begin{array}{c}
\text{P} \xrightarrow{\text{out}} T \\
\nu m P \xrightarrow{\text{out}} \nu m T(X)
\end{array}
\]

where \( \nu m T \overset{\text{def}}{=} \lambda \bar{X}. \nu m T(\bar{X}) \)

(iii) an *observational* modification: the extra structure fully matches a larger part of the skeleton. In this case, the closing context for the skeleton is smaller and we reflect this by a change in the transition label. In fact, once enough structure is added to cover the entire LHS of a skeleton (leaving a trivial context) a \( \tau \)-labelled transition is derived.

For example, consider how the ‘out’ rule of the ambient calculus is extended
in the ambient boxing structure:

\[
\begin{align*}
P & \xrightarrow{\text{[in]}} U \\
mP & \xrightarrow{\text{[out]}} T(0)(m)
\end{align*}
\]

Here we see that the program term fully matches the part of the skeleton: \(1\text{in} \text{out} n.2_{\text{Pr}} \parallel 3_{\text{Pr}}\) and thus the parameters represented by these holes must be instantiated from the program term. In this case, we use the ambient name \(m\) and the nil ambient (the unit for parallel composition up to structural equivalence). Notice also the change in the label on the transition.

For any given language, identifying exactly which kind of modification each structural extension of a term is will depend on properties of the structural equivalence relation. In particular, the associativity and commutativity of the language operators are important. To appreciate the example rule modifications above it is worth considering the structural equivalence relation of the ambient calculus: \(\nu\)-binders commute with both parallel and boxing operators, but parallel does not commute with boxing.

For our running example, in addition to the structurally defined rules already shown we also have the following rules:

\[
\begin{align*}
P & \xrightarrow{\text{[in]}} \lambda Y. U(Q \parallel Y) \\
P & \xrightarrow{\text{[out]}} \nu m U \\
& \xrightarrow{\text{[in]}} n[P] \xrightarrow{T(0)(n)} \\
\end{align*}
\]

We can see that these are substructural, superstructural and observational modifications respectively. Note that, in the final rule, a \(\tau\)-labelled transition is generated as the full \(\text{in}\) of the skeleton is matched by the program term.

We have looked at an example skeleton in which there was a single minimal active match. As mentioned above, in general there may be more than one. In fact, by considering the reduction rule:

\[
\begin{align*}
in[n.P][Q][R] & \rightarrow n[mP][Q][R]
\end{align*}
\]

of the ambient calculus, and its associated skeleton:

\[
( P^m_{\text{in}} \overset{\text{def}}{=} 1n[ \text{in}\ n.2_{\text{Pr}} \parallel 3_{\text{Pr}}] \parallel n[4_{\text{Pr}}], \ r^m_{\text{in}} \overset{\text{def}}{=} n[1n[2_{\text{Pr}} \parallel 3_{\text{Pr}}] \parallel 4_{\text{Pr}}])
\]

we can observe two minimal active matches for this rule:

\(\text{in}\ n.2_{\text{Pr}}\) and \(n[4_{\text{Pr}}]\)

For such rules we have a more complicated situation. Starting with either minimal active match, we can, as described above, give an lts axiom by using the minimal
active match as the source and the (partially) instantiated, lambda-abstracted \( \text{lhs} \) of the skeleton as the target. Structure can then be added by extending the match as above. But what of the other minimal active match? If we follow the same procedure and give an axiom whose source is the minimal active match and whose target is also the \( \text{lhs} \) of the skeleton - if we are to cover the whole \( \text{lhs} \) of the skeleton, at some point in the structural rules we will need to combine the two minimal active matches. However, in our two putative axioms we have included the \( \text{lhs} \) of the skeleton in result of the transitions, and it is not obvious how to “merge” the two. Our solution is to use co-actions, borrowing continuation-passing style. For the second minimal active match we give an axiom that instead of using the actual skeleton in the target of the transition we use an abstract skeleton. This abstract skeleton is made concrete at the point the two minimal matches are structurally combined. To see an example of this we provide a fully derived program-view labelled transition system for the ambient calculus in Figure 1. Notice the use of an abstract skeleton in axioms (cols) and (contr) and notice the merging of minimal matches in rules (inTuc) and (orTuc).

Of course, the use of co-actions is a specialised and non-uniform solution for reduction rules with just two minimal active matches. In general, and for a uniform treatment, it is possible to give all its axioms using abstract skeletons. The actual skeletons would then be provided at the point a \( \tau \)-labelled transition is generated. This approach is the more general systematic one and suits mechanisation but leads to less familiar lts rules.

### 2.2 Context-view transitions

So far we have presented a recipe for deriving structural labelled transition rules according to the program terms’ structure. Of course, in our view in which reductions are caused by interactions between context and program, we must also account for the context’s contribution to a reduction. This will take the form of instantiation of the parameters of the \( \text{lhs} \) of a skeleton which are not provided by the matched program term. The context view is therefore straightforward and can be described using the general purpose rule

\[
\begin{array}{c}
\text{\( \mathcal{M} \): \( \phi \) } \\
\text{(ctx)} \\
\text{\( \mathcal{N}, \phi, P \xrightarrow{\sigma} (\mathcal{N}, \phi, p), M \) }
\end{array}
\]

This essentially states that the context can provide any well-typed instantiations of the missing parameters. This in itself is not especially interesting but it is the combination of this rule with the program-view rules which generate the completed labelled transitions.

---

*BEATCS no 98*  THE EATCS COLUMNS

---

116
Figure 1: Process-view fragment (C). By convention τ → Pr, N → Pr, → Pr, U → Pr, A → Pr, Pr 1 → Pr. Symmetric rules (a[[1]]) omitted. When T = Pr P we use T[Q] → T[Q] and \( \text{vm} \) T \( \overset{\text{def}}{=} \text{vm} T[\text{Pr}] \).
2.3 Combined-view transitions

The combined view transitions form a labelled transition system of complete actions. It is this complete actions system that we use to define bisimilarity between programs. In its simplest form we just need to juxtapose the program-view and context-view transitions and allow for complete program-view $\tau$-actions. This gives the generic rules:

\[
\begin{align*}
P & \xrightarrow{\alpha} C \\
A & \xrightarrow{\delta_j} A' \\
\hat{M} \downarrow \quad & \Rightarrow \quad \Rightarrow \quad \Rightarrow \quad \Rightarrow \quad \\
P & \xrightarrow{\tau} C' \\
P' & \xrightarrow{\tau} P''
\end{align*}
\]

where $\xrightarrow{\alpha}$ and $\xrightarrow{\delta_j}$ refer to transitions of the program- and context-views respectively. Slightly more complicated is the rule for combining co-actions from the program view. In this case we must also provide the skeleton of the rule:

\[
\begin{align*}
P & \xrightarrow{\alpha} C \\
A & \xrightarrow{\delta_j} A' \\
\hat{M} \downarrow \quad & \Rightarrow \quad \Rightarrow \quad \Rightarrow \quad \Rightarrow \quad \\
\lambda & \xrightarrow{\pi} C' \\
\lambda & \xrightarrow{\pi} \lambda \\
\hat{M} \downarrow \quad & \Rightarrow \quad \Rightarrow \quad \Rightarrow \quad \Rightarrow \quad \\
P & \xrightarrow{\tau} C' \\
P' & \xrightarrow{\tau} P''
\end{align*}
\]

where $r^\pi_\alpha$ is the rhs of the skeleton for the reduction rule associated with $\alpha$ whose parameters have been suitably permuted.

3 Another example: the $\pi$-calculus

We have used the running example of the ambient calculus above and, we hope, this demonstrates the approach well. We have also applied the technique with some degree of success to the $\pi$-calculus [9].

Some nice features of the $\pi$-calculus as an example is that it highlights the use of the lambda-calculus as a meta-language for expressing the reduction semantics and skeletons. For example, the only reduction rule in the $\pi$-calculus traditionally takes the form

\[
a!n.P \parallel a?x.Q \rightarrow P \parallel Q[n/x]
\]

where $x$ is defined to be bound in $a?x.Q$ and a capture-free substitution is made in the target of the rule. We express this reduction in our meta-language as

\[
a!n.P \parallel a?A \rightarrow P \parallel A(n)
\]

where $A : N \rightarrow Pr$. We can then generate the skeleton for this rule as

\[
(a!1N.2Pr \parallel a?3N\rightarrow Pr , 2Pr \parallel 3N\rightarrow Pr(1N))
\]
The Bulletin of the EATCS

Given this, there are clearly two minimal active matches in the LHS:

\[ a!_1 N, 2 Pr \quad \text{and} \quad a?_3 N \rightarrow Pr \]

Using the co-action approach (arbitrarily choosing output as the main axiom), we obtain program-view axioms:

\[
\frac{a!_n P \rightarrow \lambda X : (N \rightarrow Pr) (P(X(n)))}{(n\alpha v)}
\]

and

\[
\frac{a?_A Z : ((N \rightarrow Pr) \rightarrow Pr) Z(A)}{(rbo)}
\]

there are (not shown) superstructural modifications for these rules for \( || \) and \(\nu\) contexts and the communication rule:

\[
\frac{P \rightarrow_a C U Q \quad Q \rightarrow_a C \lambda T}{P||Q \rightarrow_a C \lambda T(U)} (\tau a)
\]

We argue in [9] that this provides a better structural account of the labelled transition system of the \(\pi\)-calculus than the more traditional one [12]. In addition to this, the complete action system provides a bisimilarity equivalence which coincides neatly with contextually defined equivalences, even for \(\pi\)-calculus variants e.g. \(\pi\)-calculus without the matching operator, the asynchronous \(\pi\)-calculus, and the higher-order \(\pi\)-calculus.

4 Soundness, Completeness and Observability

In this section we consider the properties that we want our derived labelled transition systems to satisfy. As a bare minimum we expect that the program-view transitions provide a structurally inductive definition of the reduction relation of the language:

**Property 3** (\(\tau\) and Reduction). If \( P \rightarrow^\tau P' \) then \( P \rightarrow P' \). Conversely, if \( P \rightarrow P' \) then \( \exists P'' : P'' \equiv P' \) such that \( P \rightarrow^\tau P'' \).

For our example derived systems this property is straightforward to prove using structural induction.

The more interesting question about our labelled transition systems is to what extent (strong) bisimilarity over complete actions \(\sim_{C,\beta}\) coincides with a more standard equivalence between programs. For process calculi such as ambients and \(\pi\)-calculus we can use the touchstone equivalence of reduction barbed congruences as a point of comparison. Reduction barbed congruence \(\simeq\) depends on the
choice of an observation predicate (or barb) for the language but beyond that is
defined as the largest reduction preserving, barb preserving, congruence relation. If the above property holds for our language then bisimilarity will be reduction preserving. We need to check on a per-example basis if our bisimilarity is barb-preserving, and this leaves us with congruence.

**Property 4 (Congruence).** If $P \sim_{CA} Q$ then $C[P] \sim_{CA} C[Q]$ for all language contexts $C$.

A proof of congruence of our bisimilarity for any given language is still work in progress, but we have proved it, using a very straightforward inductive proof, in our example systems. We are confident that a general proof can be established due to the careful construction of the labelled transition system. In essence, by definition there are labelled transitions provided for every possible (partial) context which can cause interaction with a program term. In order for congruence to fail, there must be an enclosing context which provokes different reactions for the bisimilar programs $P$ and $Q$.

The main obstacles to a general congruence result are the use of higher-order meta-language to describe reductions and the analysis of the structural equivalence for a given language to determine which rules are sub-, super- or observational modifications.

If we can establish barb-preservation and congruence, we will have soundness:

**Property 5 (Soundness).** If $P \sim_{CA} Q$ then $P \equiv Q$.

The converse property, ‘Completeness’, is less clear.

**Property 6 (Completeness).** If $P \equiv Q$ then $P \sim_{CA} Q$.

In general, the derived labelled transition system will not be complete. Indeed, the labelled transition system presented above for the ambient calculus is sound but not complete for reduction barbed congruence in that setting. The failure of completeness is a subtle issue which depends on power of the contexts to observe state changes in the program due to interaction. For example, contexts do not have the general ability to observe the reception of a value in the asynchronous $\pi$-calculus because the interacting context can only asynchronously send a message to be received. Without complicity from the receiving program in sending an $ack$, there is no way to tell whether the message has been received or not.

This analysis of observability is beyond the scope of our project but nonetheless we can partially address the issue. A known approach to dealing with incompleteness in the asynchronous setting is to use so-called Honda-Tokoro rules [3]. These are rules which obscure the observability of certain labelled actions by providing alternative means of generating transitions decorated with the action. For
example, we add the rules

\[
\begin{align*}
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \qa
References


THE DISTRIBUTED COMPUTING COLUMN

BY

PANAGIOTA FATOUROU

Department of Computer Science, University of Crete
P.O. Box 2208 GR-714 09 Heraklion, Crete, Greece
and
Institute of Computer Science (ICS)
Foundation for Research and Technology (FORTH)
N. Plastira 100. Vassilika Vouton
GR-700 13 Heraklion, Crete, Greece
faturu@csd.uoc.gr

TRANSACTIONAL MEMORY: A PRIMER FOR THEORISTS

Maurice Herlihy
Computer Science Department
Brown University
mph@cs.brown.edu

Abstract
This article gives a brief overview of Transactional Memory, and speculates about some open research questions.

1 Introduction

Although I cannot be entirely sure, I am convinced that paper got in by the skin of its teeth. Several people, including program committee members, came up to me afterwards and referred to this work as an “idea paper”. In the architecture community, which values quantification above all else, calling something an idea paper, while not exactly fighting words, falls well short of being a compliment. Nevertheless, the paper was accepted, and in 2008 it won the “ISCA Influential Paper” award, given to the paper deemed, fifteen years later, to have been the most influential.

The fifteen-year lag is essential. Five years after publication, that paper had few, if any citations. Even ten years later, while citations had suddenly started to appear, it seems unlikely that transactional memory would have been considered one of the greatest hits of 1993. Fifteen years later, however, transactional memory had entered the mainstream. Papers on the subject have appeared in conferences on areas as diverse as architecture, programming languages, compilers, and distributed computing.

The reason for this gradual change from neglect to interest is clear: the rise of the (multicore) machines. Multicore changes everything. Before, research in concurrency was respected, but not taken seriously as an influence on computing practice. In those days, Moore’s Law guaranteed that clock speeds would effectively double in speed every few years, so there was little incentive to worry about parallelization. Today, by contrast, Moore’s Law ensures that the number of cores will grow for the foreseeable future, but that clock speeds will remain roughly flat.

As a consequence, today’s purveyors of fine hardware and software products now have substantial incentives to worry about parallelization. Many believe that classical approaches to parallelization, ranging from low-level locks and condition variables, to high-level parallelizing compilers, are unequal to the task. Transactional memory is an alternative programming model intended to address these shortcomings.

Transactional memory supports a computational model in which each thread executes a sequence of transactions. A transaction is a sequence of steps executed by a single thread. Transactions are atomic: each transaction either commits (it takes effect) or aborts (its effects are discarded). Transactions are linearizable [14]: they appear to take effect in a one-at-a-time order. When the transaction is complete, the thread tries to commit the transaction. If the commit succeeds, the transaction’s operations take effect; otherwise, they are discarded.

These memory transactions satisfy the same formal serializability and atomicity properties as database-style transactions, but they are intended to be used very differently. Unlike database transactions, memory transactions are short-lived activities that access a relatively small number of objects in primary memory. The effects of database transactions are persistent, and committing a transaction involves backing up changes on a disk. Memory transactions are not persistent, and
As an aside, there are even more radical approaches to concurrent programming, such as "shared-nothing" models based on message-passing or pure functional programming. While these approaches contain many valuable and influential ideas, they are unlikely to have a direct impact in the near future. Like Esperanto, the Dvorak keyboard, or Naturism, they have compelling justifications in the abstract, and while they may work well in particular circumstances, they are unlikely to become mainstream in time to help with the current crisis. Instead, I expect mainstream programming will co-opt their best ideas. For example, I think “mostly functional” languages (such as transactional Haskell [11]) that distinguish statically between private and shared objects (but allow both) will have a broad influence on near-term language design. On the other hand, the question whether communication should occur through messages or other shared objects strikes me as a distinction without a difference, since one model is easily simulated by the other.

This article is an idiosyncratic and opinionated reflection on open research questions that might be of interest to the Theoretic Computer Science community. Many of the questions I raise are not posed as theory questions: they may seem to be the property of other research communities, such as compilers, operating systems, programming languages, electrical engineering, and so on. Part of the challenge is to articulate a precise model, precise problems, and precise solutions.

Research in Transactional Memory can be divided into three broad problem areas: software engineering, highly-concurrent data structures, and implementation techniques. This subdivision is somewhat arbitrary, since some topics arguably fit in several areas, some in none, but it keeps things organized.

2 Concurrent Programming

Before Transactional Memory, there were locks. Whenever one thread needed to modify a shared data object, it would lock that object for the duration of its method call, ensuring that no other thread could access the object while it was locked.

First, programmers must decide between coarse-grained locking, in which a large data structure is protected by a single lock, and fine-grained locking, in which a lock is associated with each component of the data structure. Coarse-grained locking is simple, but permits little or no concurrency, thereby preventing the program from exploiting multiple processing cores. By contrast, fine-grained locking is substantially more complicated because of the need to ensure that threads acquire all necessary locks (and only those, for good performance), and because of the need to avoid deadlock when acquiring multiple locks. The decision is further complicated by the fact that the best engineering solution may
be platform-dependent, varying with different machine sizes, workloads, and so on, making it difficult to write code that is both scalable and portable.

Second, conventional locking provides poor support for code composition and reuse. For example, consider a lock-based hash table that provides atomic add() and remove() methods. Ideally, it should be easy to move an element atomically from one table to another, but this kind of composition simply does not work. If the table methods synchronize internally, then there is no way to acquire and hold both locks simultaneously. If the tables export their locks, then modularity and safety are compromised.

Finally, such basic issues as the mapping from locks to data, — that is, which locks protect which data, and the order in which locks must be acquired and released — are all based on convention, and violations are notoriously difficult to detect and debug. For these and other reasons, today’s software practices make concurrent programs too difficult to develop, debug, understand, and maintain.

Here is how Transactional Memory addresses these problems: Under locking, critical sections conflict if they lock the same lock. Under Transactional Memory, by contrast, transactions conflict only if they actually share data at run-time, reducing the incentive for shrinking synchronization granularity. While some Transactional Memory implementations are subject to deadlock, once the deadlock is detected (or suspected), it is straightforward to roll back application-level transactions in a transparent manner. Transactional memory thus relieves the programmer of worry about deadlock. (Open nested transactions [21], discussed elsewhere, are a regrettable exception to this rule.) Finally, transactions eliminate the need to rely on conventions identifying which locks protect which data.

Nevertheless, TM is not a “silver bullet” that magically dissolves all obstacles to concurrent programming. The transformation from locking to transactions has hazards. For example, it is possible to devise a correct lock-based program that deadlocks if locks are replaced by transactions. It is also possible to devise two transactions that behave correctly in isolation, but that deadlock if they are encompassed in a larger transaction. There are ambiguities associated with interactions between transactional and non-transactional threads. Finally, what does it mean for a transaction to abort, and what can an aborted transaction do? The following examples are adapted from Blundell et al. [3].

Figure 1 illustrates why it is not possible to blindly replace lock-based critical sections with transactions. If Threads 0 and 1 run concurrently, their critical sections synchronize on different locks, so updates to the Boolean variables $a$ and $b$ can proceed concurrently. Threads 2 and 3 are identical to Threads 0 and 1, except that the critical sections have been replaced by transactions. As a result, either Thread 2 observes Thread 3’s update to $a$, or Thread 3 observes Thread 2’s update to $b$, but not both. It follows that the program deadlocks.

This particular program is (arguably) pathological, and the authors perhaps
volatile boolean a = false;
volatile boolean b = false;
Object x = new Object();
Object y = new Object();

/* Thread 0 */ /* Thread 1 */
synchronized(x) {
    synchronized(y) {
        while (!a) {} a = true;
        while (!b) {} b = true;
    }
}

/* Thread 2 */ /* Thread 3 */
atomic {
    atomic {
        while (!a) {} a = true;
        while (!b) {} b = true;
    }
}

Figure 1: Locks not Equivalent to Transactions

get what they deserve. The real reason this issue is important is that it affects
the design of optimizing compilers that might convert legacy critical sections to
transactions.

Clearly, transactions are isolated from one another: no transaction can observe
the partial of effects other transactions. How are transactions isolated from activi-
ties of non-transactional threads? Figure 2 shows an example of two threads, one
atomic, one not. Under strong isolation, when Thread 1 reads x, it cannot read 0,
and division by x will never cause an exception. Under weak isolation, however,
Thread 1 might read 0, and the subsequent division could cause an exception.

We have seen that one of the problems with locking is lack of composability:
one cannot, in general, compose two lock-based programs. Transaction-based
programs are compositional with respect to safety properties, but not always with
respect to liveness properties. Figure 3 shows three threads. Here we use the retry
construct [11] which aborts and (sometime later) restarts the calling transaction.
If Thread 0 and Thread 2 run together, Thread 2’s transaction can be interleaved
between Thread 0’s two transactions. Thread 1 differs from Thread 2 only by
encompassing both of Thread 0’s transactions in a single transaction. If Thread 1
and Thread 2 run together, they will deadlock.

One might view this anomaly as saying that replacing locks with transactions
replaces a model where almost no composition is possible with one where certain
```c
int x = 1;

// thread 0 */ // thread 1 */
atomic{
    x = 0;            int y = 100 / x;
    x = 1;
}
```

Figure 2: Weak vs Strong Isolation

```c
boolean a = false;
boolean b = false;

// thread 0 */ // thread 1 */ // thread 2 */
atomic {
    atomic{
        atomic{
            a = true;       a = true;       if (!a) retry ;
        }
        atomic{
            b = true;
        }
    }
    atomic{
        if (!b) retry ;
    }
}
```

Figure 3: Composition
pathological compositions still present problems. The deeper problem, however, is that optimizing compilers cannot obliviously coalesce adjacent transactions to reduce per-transaction overheads. There are many open research questions concerning the kinds of compiler optimizations that are possible for transactional programs.

A zombie transaction is one that continues to run after it aborts. Zombie transactions are dangerous because in some TM implementations, they may observe inconsistent states, including partial effects of other transactions. Although zombies never commit, they may still have unfortunate side-effects such as infinite loops, dividing by zero, and so on. Opacity [9] is a correctness condition that (among other things) prevents zombies from misbehaving.

One way to tie many of these questions together is an approach called single global lock semantics [19], where transactions have the same effects as critical sections protected by a single (implicit) global lock. Nevertheless, researchers [18] have observed that this model has its own drawbacks.

**Research Questions**

- Devise a reasonably effective way to decide when it is safe for a compiler to turn a critical section into a transaction.
- Devise a reasonably effective way to decide when it is safe for a compiler to merge multiple transactions.
- Devise an efficient algorithm to provide strong isolation in STMs, or prove one doesn’t exist.
- Devise static checks to ensure there are no such transactional/non-transactional races, perhaps using algorithms similar to those used by existing race-detection tools [23].
- The Theory community has made substantial contributions to understanding the implications of weak memory models for conventional multiprocessors (for example, [15, 7, 17]). Do the same for transactional models of computation. (See Grossman et al. [8] for a list of questions.)
- Devise general proof rules for showing that a TM implementation supports opacity [9]. Are there other correctness conditions of interest?

### 3 Highly-Concurrent Data Structures

Back in 1993, we considered TM as a way to make lock-free algorithms accessible to everyday programmers. Today, most of the research in this area is
concerned with making general-purpose concurrent programs easier to write and easier to reason about. Nevertheless, I think the question of how to design highly-concurrent data structures (lock-free or otherwise) in a transactional model is an important and still under-explored area.

Concurrent data structures are data abstractions. They provide clients with simple sequential specifications that can be understood and exploited by nonspecialists. For example, a map data structure would link keys to values, or a queue data structure would be a sequence of values. An operation such as binding a key to a value or enqueuing a value should be *linearizable*: it should appear to happen instantaneously while the method is in progress. On the inside, however, these data structures are typically quite complex, engineered by specialists to match the characteristics of the underlying platform.

From a practical perspective, shared data structures are the “ball bearings” of concurrent applications: they are where otherwise independent parts interact. Like ball bearings, if these objects become hot-spots, then individually well-engineered components will not combine to form a well-engineered system. *Amdahl’s Law* [13, p.13] states that granularity matters in synchronization: even short sequential sections can substantially reduce the scalability of otherwise well-designed concurrent systems.

To illustrate why transactions provide an attractive alternative to conventional locks and condition variables, consider the problem of constructing a concurrent FIFO queue that permits one thread to enqueue items at the tail of the queue at the same time another thread dequeues items from the head of the queue, while the queue is non-empty. Any problem so easy to state, and that arises so naturally in practice, should have a simple and understandable solution. In fact, solving this problem with conventional tools requires substantial cleverness [20]. By contrast, it is almost trivial to solve this problem using transactions. Figure 4 shows how the queue’s enqueue method might look in a language that provides direct support for transactions (for example, see Harris et al. [10]). It consists of little more than enclosing sequential code in a transaction block, and handling an exception if the transaction aborts. In practice, of course, a complete implementation would include more details (such as how often to retry a failed transaction), but even so, this concurrent queue implementation by itself is not a publishable result.

Nevertheless, many research challenges remain. For example, almost all transactional memory systems, both hardware and software, synchronize on the basis of *read/write conflicts*. As a transaction executes, it records the locations (or objects) it read in a *read set*, and the memory locations (or objects) it wrote in a *write set*. Two transactions *conflict* if one transaction’s read or write set intersects the other’s write set. Conflicting transactions cannot both commit. Conflict *detection* can be eager (detected before it occurs) or lazy (detected afterwards). Conflict
class Queue<T> {
    QNode head;
    Qnode tail;
    public void enq(T x) {
        atomic {
            Qnode q = new Qnode(x);
            tail .next = q;
            tail = q;
        }
        catch (AbortException e) {...}
    }
    ...
}

Figure 4: Transactional queue code fragment

resolution (deciding which transactions to abort) can be implemented in a variety of ways.

Synchronizing via read/write conflicts has one substantial advantage: it can be done automatically without programmer participation. It also has a substantial disadvantage: it can severely and unnecessarily restrict concurrency for certain shared objects. If these objects are “hot-spots”, then the performance of the system as a whole may suffer.

For example, consider a mutable set of integers that provides add(x), remove(x) and contains(x) methods with the obvious meanings. Suppose we implement the set as a sorted linked list: each list node has two fields, an integer value and a node reference next. List nodes are sorted by value, and values are not duplicated. Integer x is in the set if and only if a list node has value field x. The add(x) method reads along the list until it encounters the first value greater than or equal to x. Assuming x is absent, it creates a node to hold x, and links that node into the list.

Suppose a set has state {1, 3, 5}. Transaction A is about to add 2 to the set and transaction B is about to add 4. Since neither transaction’s pending method call depends on the other’s, there is no inherent reason they cannot run concurrently. Nevertheless, calls to add(2) and add(4) do conflict in the list implementation, because no matter how A and B’s steps are interleaved, A must write to a node read by B. This conflict is quite serious because transactional delays can be quite long. Unlike conflicts between short-term locks, where the delay is typically bounded by a statically-defined critical section, if transaction A is blocked by B, then A is blocked while B completes an arbitrarily long sequence of steps.

Recently, my students and I have started looking at exploiting algebraic prop-
erties of data types to escape the restrictions imposed by read-write synchronization. We proposed Transactional Boosting [12], a methodology for taking a large class of highly-concurrent linearizable objects not designed with transactions in mind, and transforming them into highly-concurrent objects that can be used in transactional memory systems.

Start with a base object, a thread-safe linearizable concurrent object, perhaps taken from a highly-optimized non-transactional library such as, for example, java.util.concurrent. We treat such an object as a black box, relying only on the object’s specification that characterizes its abstract state (for example, it is a set of integers), and how its methods affect the state (for example, add(x) ensures that x is in the set). We also require certain regularity conditions (basically, that methods have inverses).

The specification for a linearizable base object defines an abstract state (such as a set of integers), and a concrete state (such as a linked list). Each method is usually specified by a precondition (describing the object’s abstract state before invoking the method) and a postcondition, describing the object’s abstract state afterwards, as well as the method’s return value.

Informally, two method invocations commute if applying them in either order leaves the object in the same state and returns the same response. In a set, for example, add(x) commutes with add(y) if x and y are distinct. Absence of commutativity is the basis of how transactional boosting performs conflict detection.

For synchronization, we associate an abstract lock [21] with each invocation of a boosted object. Simplifying somewhat, two abstract locks conflict if their invocations do not commute. Abstract locks ensure that non-commuting method calls never occur concurrently. Before a transaction calls a method, it must acquire that method’s abstract lock. The caller is delayed while any other transaction holds a conflicting lock (timeouts avoid deadlock). Once it acquires the lock, the transaction makes the call, relying on the base linearizable object implementation to take care of thread-level synchronization. In the integer set example, the abstract locks for add(x) and add(y) do not conflict when x and y are distinct, so these calls can proceed in parallel.

A method call m has inverse m’ if applying m’ immediately after m undoes the effects of m. For example, a method call that adds x to a set not containing x has as inverse the method call that removes x from the set. A method call that adds x to a set already containing x has a trivial inverse, since the set’s state is unchanged.

When inverses are known, recovery can be done at the granularity of method calls. As a transaction executes, it logs an inverse for each method call in a thread-local log. If the transaction commits, the log is discarded, and the transaction’s locks are released. However, if the transaction aborts, the transaction revisits the log entries in reverse order executing each inverse. (A transaction that added x to the set would call remove(x).) When every inverse has been executed, the
transaction releases its locks.
Algebraic approaches to synchronization have the following advantages: they permit substantially more concurrency than conventional read-write synchronization; they permit re-use of highly-optimized algorithms, written by experts, in a transactional context; and instead of keeping track of read and write sets (which may be huge), it is only necessary to log pointers to inverse methods (closures), a much smaller data footprint.

Research Questions

• Is it possible to detect automatically whether two method calls commute?

• Is it possible to check (using verification or model checking) a programmer’s assertion that two method calls commute?

• Same questions for inverses.

• Are there weaker notions of conflict than commutativity? One can devise scenarios where non-commutative operations could proceed concurrently. For example, if a transaction compares the value of two counters, it may not matter if the counters change, as long as the comparison is unaffected.

• Boosting updates the object in place, and undoes the effects on abort. Is there a useful variation in which method calls are applied only on commit?

Open nested transactions [21] (ONT) have been proposed as a way to implement highly-concurrent transactional objects. With ONT, a nested transaction can be designated as open. If an open transaction commits, its effects immediately become visible to all other transactions. That is, an open nested transaction can commit (and make its modifications visible) before it is known whether the parent transaction will commit or abort. To compensate, each open nested transaction can register an abort handler, a method that is run if the parent aborts. The abort handler is intended to perform an application-specific compensating action (for example, deleting files created by an open nested transaction).

Although open nested transactions address a very important problem, I think the idea is fundamentally flawed. For example, abort handlers can deadlock. Normally, one clear deadlocked transactions by aborting and rolling some of them back, but how do you clear transactions that deadlock in the middle of rolling back? Moreover, it is left undefined what happens if an open nested transaction’s read set intersects a parent’s write set, or its write set intersects a parent’s read or write set. Ironically, it is also too weak in the sense that certain common problems, such as traversing linked-lists or B-trees, simply do not lend themselves to open nested transactions.
Research Questions

- In general, is it possible to transform open nested transactions into a safe and useful mechanism?

- More specifically, can we place restrictions on abort handlers to avoid deadlock?

- Can we describe, in an implementation-independent way, the semantics of a program in which an open nested transaction’s read set intersects its parent’s write set, or its write set intersects its parent’s read or write set?

- If not, can we devise a compile-time or run-time test to avoid such anomalies?

- Even assuming we can avoid all such anomalies, can we describe the circumstances under which this mechanism yields correct and desired results?

4 Implementation

The other promise of transactional memory is that it might provide better performance than conventional models. This issue is quite complex, and easily misunderstood.

Bloom filters and counting Bloom filters are used to avoid synchronization costs in transactional memory implementations such as Bulk [4] and LogTM Signature Edition [24], and VTM [22]. They also appear in the Dreadlocks deadlock detection algorithm [16].

Another example of a highly-concurrent data structure well-adapted to transactional memory implementations is the scalable non-zero indicator (SNZI) [6], a concurrent counter that provides a operations to increment, decrement, and test whether the counter is zero (without returning the actual value). Such weak semantics is important, because it admits a highly scalable, cache-friendly implementation. It central to the run-time data structures used in the SkySTM [5] transactional memory system. I conjecture that many equally useful concurrent data structures are yet to be discovered.

Research Questions

- Bloom filters replace an expensive exact check with a fast check that admits false positives. Are there other synchronization problems where such a trade-off makes sense?
Energy consumption is an important area that has received little attention from the Theory community. There are many challenges, including a lack of a universally-accepted model. Nevertheless, this area is going to become increasingly important as power-hungry devices such as mobile phones, GPS devices, and game consoles become more and more common. Eventually, such devices will affect every aspect of modern life, and their power consumption profiles will have a broad economic impact.

Research Questions

- **How does speculative synchronization affect energy consumption?** Many TM systems rely on speculation for performance, but the effect of speculation on power remains poorly understood, both empirically and analytically.
- **How does non-speculative synchronization affect energy consumption?** Modern multicore systems will require a mix of speculative and non-speculative synchronization. For example, much remains to be done before we understand the energy implications of locking algorithms (devise scalable, energy-efficient spin-lock algorithms), and at the systems level (devise energy-efficient lock-based transactions?). In particular, there are no lower bounds.
- **How does task-level scheduling affect energy consumption?** Researchers have examined the problem of decomposing software into tasks, using provably efficient software scheduling algorithms such as work-stealing to provide concurrency [1, 2]. If energy consumption complements or replaces latency as the desired objective, can one devise similar scheduling algorithms for power based on voltage or frequency scaling, or powering down unneeded devices?
- **How do data structures affect energy consumption?** There has substantial work on highly-concurrent, cache-friendly data structures, naturally focusing on performance, not energy consumption. High-performance data structures often save energy, but not always. Little is known how the choice of algorithms for data structures such as hash tables, queues, or skip lists affects energy consumption.

5 Conclusions

In my experience, Theorists are often reluctant to address questions relating to multicores and concurrency. This reluctance may stem from memories of the
once-popular PRAM model, which produced many elegant algorithms, but with a disappointing impact on practice. It may also stem from a lack of clean problem statements and models, but a degree of conceptual messiness is exactly what characterizes new and exciting areas. The glory of Theory is its proven ability to impose order on other fields’ chaos, as Turing Machines once imposed order on the design of mechanical computing devices.

6 Acknowledgments

Victor Luchangco and Youla Fatourou made many valuable suggestions and corrections.

References

Memory system performance and correctness, pages 62–69, New York, NY, USA, 2006. ACM.


THE FORMAL SPECIFICATION COLUMN

BY

HARTMUT EHRLIG

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

COSPAN DPO APPROACH:
AN ALTERNATIVE FOR DPO GRAPH TRANSFORMATIONS

Hartmut Ehrig, Frank Hermann, Ulrike Prange

Technische Universität Berlin, Germany
ehrlig|frank|uprange@cs.tu-berlin.de

Abstract

The DPO approach for graph transformations is based on productions \( p = (L \leftarrow K \rightarrow R) \) and direct transformations defined by two pushouts, where, roughly spoken, in the first pushout all items in \( L \setminus K \) are deleted and in the second one all items \( R \setminus K \) are added, while those items in \( K \) are preserved. Intuitively, \( K \) is the intersection of \( L \) and \( R \) and, formally, \( p = (L \leftarrow K \rightarrow R) \) is a span of graph morphisms.

In this paper, we consider productions \( \overline{p} = (L \rightarrow \overline{K} \leftarrow R) \) which are cospanps of graph morphisms, and \( \overline{K} \) corresponds to the union of \( L \) and \( R \). As before, direct transformations are defined by double pushouts, but now the first pushout adds all items in \( \overline{K} \setminus L \) and the second one deletes \( \overline{K} \setminus R \). This basic idea can be extended to an alternative graph transformation approach, called cospan DPO approach. Key notions of the classical DPO approach...
can be reformulated in the cospan DPO approach and our main result shows in which way corresponding concepts and results are equivalent.

Introduction

The DPO approach for graph transformation has been introduced in [1] and is today one of the most prominent graph transformation approaches concerning theory and applications (see [2, 5, 6] and [7]). It is based on the idea of gluing graphs along designated subgraphs, which can be formalized by the idea of pushouts in the category of graphs. More precisely, a production \( p = (L \leftarrow K \rightarrow R) \) consists of graphs \( L, K, \) and \( R, \) called left hand side, gluing, and right hand side, respectively, and two injective graph morphisms \( l \) and \( r. \) A direct transformation \( G \Rightarrow H \) via \( (p, m) \) with match \( m : L \rightarrow G \) is defined by two pushouts (1) and (2), where in pushout (1) all items \( m(L \setminus K) \) in \( G \) are deleted leading to graph \( D, \) and in pushout (2) all items in \( R \setminus K \) are added leading to graph \( H. \) Given \( p \) and \( m : L \rightarrow G \) the context graph \( D \) can only be constructed if a suitable gluing condition is satisfied.

Various modifications of this “classical” DPO concept have been studied in the literature, e.g. the double-pullback approach, where (1) and (2) are pullbacks and not necessarily pushouts [4], the sesqui-pushout approach, where production morphisms may be non-injective and (1) is a certain pullback, but not necessarily a pushout [3], and the DPO approach in adhesive categories [8] or weak adhesive HLR categories [7], where the category of graphs is replaced by suitable other categories, like the category of labeled graphs, typed graphs, hypergraphs, attributed graphs, Petri nets, or algebraic high-level nets.

Moreover, the DPO approach has been implemented in the AGG system for typed attributed graphs including simulation and analysis of graph transformation systems [9].

From the implementation point of view, however, it is sometimes easier to add the new items first and then delete (some of) the old items in the second step. In some sense, this would mean to construct first the pushout (2) and then the pushout (1) in the diagram above. However, this is not possible if production \( p \) and match \( m \) are given as above. But if we replace \( p \) by a cospan \( \overline{p} = (L \rightarrow K \leftarrow R), \) where \( K \) corresponds to the union of \( L \) and \( R, \) we can consider pushouts (3) and
(4) to define a “direct transformation” $G \Rightarrow H$ via $\bar{p}$ and match $m$, called cospan direct transformation. This means that we first glue together $\overline{R}$ and $G$ along $\overline{L}$ in pushout (3), which corresponds to adding $\overline{K} \setminus L$, and in the second step delete $\overline{K} \setminus R$ in pushout (4).

At first glance, this looks strange because we are not sure whether the pushout complement construction leading to pushout (4) exists. In fact, similar to the classical DPO approach, we now need a “cospan gluing condition” in order to construct pushout (4).

In this paper, we will analyse whether it is possible to develop a cospan DPO approach similar to the classical DPO approach and how far both are equivalent. In Section 1, we introduce some basic concepts and results for the cospan DPO approach for graphs. In Section 2, we extend this approach to other categories and establish the relationship to the classical DPO approach which allows us to give indirect proofs for the results in Section 1. In Section 3, we conclude and give an overview of future work.

1 Basic Concepts of the Cospan DPO Approach for Graphs

In this section, we introduce some basic concepts and results for the cospan DPO approach for graphs. We do not give direct proofs for these results, but they can be obtained from the equivalence to the classical DPO approach shown in Section 2.

**Definition 1.1** (Cospan Production and Direct Transformation). A cospan production, short production, $\bar{p} = (L \xrightarrow{i} \overline{K} \xleftarrow{r} R)$ consists of graphs $L$, $\overline{K}$, and $R$, and injective graph morphisms $i$ and $r$ which are jointly surjective.

Given a cospan production $\bar{p}$ and a match $m : L \rightarrow G$, a (cospan) direct transformation $G \Rightarrow H$ via $(\bar{p}, m)$ consists of the following two pushouts (1) and (2), where items are added by pushout (1) and deleted by pushout (2).
In order to construct pushout (2), we need the following cospan gluing condition, where the boundary of a morphism $f : A \to A'$ is an inclusion from the smallest subgraph of $A$ such that a pushout complement exists. For a morphism $f : A \to A'$ we want to construct a boundary $b : B \to A$, a boundary object $B$ and a context $C$ leading to a pushout, where roughly spoken, $A'$ is the gluing of $C$ and $A$ along the boundary object.

**Definition 1.2 (Cospan Gluing Condition).** Let $B \xrightarrow{b} L$ be the boundary of the match $m : L \to G$, then $m$ satisfies the cospan gluing condition w.r.t. $\bar{p} = (L \xrightarrow{l} K \leftarrow R)$ if there is a morphism $\bar{b} : B \to R$ such that $\bar{p} \circ \bar{b} = l \circ b$.

**Remark.** Note, that $B \xrightarrow{b} L \xrightarrow{i} K$ is also the boundary of $\bar{k}$ in pushout (1), so that the classical gluing condition is satisfied for $\bar{k}$ w.r.t. $\bar{p}$, which allows to construct the context graph $H$ in pushout (2).

Analogously to the classical case, cospan direct transformations can be constructed iff the cospan gluing condition is satisfied, and are unique up to isomorphism.

**Fact 1.3 (Construction and Uniqueness of Direct Transformations).** Given a cospan production $\bar{p}$ and a match $m : L \to G$ then there is a direct transformation $G \xrightarrow{\Rightarrow} H$ via $(\bar{p}, m)$ iff the cospan gluing condition is satisfied for $m$ w.r.t. $\bar{p}$. Moreover, in this case $D$ and $H$ in pushouts (1) and (2) above are uniquely determined up to isomorphism.

Once we know how to construct direct transformations in the cospan DPO approach we define as usual transformations as sequences of direct transformations. Now we can study under which conditions we are able to obtain the basic results of the classical DPO approach, like the Local Church-Rosser, Parallelism, Concurrency and Embedding Theorems, also in the cospan DPO approach. In this section, we only consider the Local Church-Rosser Theorem which is based on parallel and sequential independence.
Definition 1.4 (Cospan Parallel Independence). Two direct transformations $G \Rightarrow H_1$ via $(\overline{p}_1, m_1)$ and $G \Rightarrow H_2$ via $(\overline{p}_2, m_2)$ are called (cospan) parallel independent if there are morphisms $m'_1 : L_1 \to H_2$ and $m'_2 : L_2 \to H_1$ such that $y_2 \circ m'_1 = x_2 \circ m_1$ and $y_1 \circ m'_2 = x_1 \circ m_2$.

Remark. Sequential independence is defined in a similar way.

The existence of $m'_1$ and $m'_2$ means that we obtain a match from $L_1$ of $\overline{p}_1$ to $H_2$ and a match from $L_2$ of $\overline{p}_2$ to $H_1$. These matches will allow us to construct direct transformations $H_2 \Rightarrow H$ via $(\overline{p}_1, m'_1)$ and $H_1 \Rightarrow H$ via $(\overline{p}_2, m'_2)$.

Theorem 1.5 (Local Church-Rosser Theorem). Given parallel independent direct transformations $G \Rightarrow H_1$ via $(\overline{p}_1, m_1)$ and $G \Rightarrow H_2$ via $(\overline{p}_2, m_2)$ there is a graph $H$ and direct transformations $H_2 \Rightarrow H$ via $(\overline{p}_1, m'_1)$ and $H_1 \Rightarrow H$ via $(\overline{p}_2, m'_2)$ such that the sequences become sequentially independent, and vice versa.

2 Equivalence of Classical and Cospan DPO Approach

The basic concepts and results of Section 1 can be reformulated for adhesive high-level replacement (HLR) systems based on (weak) adhesive HLR categories $(\mathcal{C}, \mathcal{M})$ (see [7]). For this purpose, we replace graphs by the objects of the category $\mathcal{C}$, injective graph morphisms by morphisms of a morphism class $\mathcal{M}$ of monomorphisms, and jointly surjective graph morphisms by jointly epimorphic morphisms. Thus, a cospan production $p = (L \xrightarrow{\overline{l}} K \leftarrow R)$ consists of objects $L$, $K$, and $R$ in $\mathcal{C}$, where $\overline{l}$ and $\overline{r}$ are $\mathcal{M}$-morphisms and jointly epimorphic.

It is possible to weaken the condition of jointly epimorphic morphisms of a cospan production. A cospan of $\mathcal{M}$-morphisms is called a generalized production.
Definition 2.1 (Generalized Cospan Production). A generalized cospan production $p = (L \xrightarrow{l} K \xleftarrow{r} R)$ consists of objects $L$, $K$, and $R$, and $\mathcal{M}$-morphisms $\tilde{l}$ and $\tilde{r}$.

Since we do not require jointly epimorphic morphisms for the cospan gluing condition or the construction of pushouts, a cospan direct transformation over a generalized production exists and is defined analogously to the normal cospan production. Moreover, for every generalized cospan production $p$ there is a cospan production $\overline{p}$ such that $G \Rightarrow H$ via $(\overline{p}, m)$ if and only if $G \Rightarrow H$ via $(p, m)$. This implies that it is sufficient to consider normal cospan productions instead of generalized ones.

Definition 2.2 (Closure of Generalized Cospan Production). Given a generalized cospan production $p = (L \xrightarrow{l} K \xleftarrow{r} R)$ then the closure of $p$ is a cospan production $\overline{p} = (L \xrightarrow{\bar{l}} \overline{K} \xleftarrow{\bar{r}} R)$ such that (1) is the pullback of $\tilde{l}$ and $\tilde{r}$, and (2) is the pushout of $l$ and $r$.

![Diagram](image)

Theorem 2.3 (Equivalence of $p$ and $\overline{p}$). In a weak adhesive HLR category with effective pushouts, given a generalized cospan production $p = (L \xrightarrow{l} K \xleftarrow{r} R)$, its closure $\overline{p} = (L \xrightarrow{\bar{l}} \overline{K} \xleftarrow{\bar{r}} R)$, and a match $m : L \rightarrow G$, then we have that

$$G \Rightarrow H \text{ via } (\overline{p}, m) \iff G \Rightarrow H \text{ via } (p, m).$$

Remark. Effective pushouts means that for a pullback (1) and a pushout (2) as above, with all morphisms in $\mathcal{M}$, the induced morphism $\overline{K} \rightarrow \overline{K}$ is also an $\mathcal{M}$-morphism.

For a cospan production $\overline{p}$ with jointly epimorphic morphisms, the closure of $\overline{p}$ is the cospan production $\overline{p}$ itself (up to isomorphism), since a pullback over jointly epimorphic $\mathcal{M}$-morphisms is already a pushout in weak adhesive HLR categories with effective pushouts.

Proof. Due to (2) being a pushout, (1) being a pullback and effective pushouts there is an induced morphism $\overline{K} \rightarrow \overline{K}$ which is an $\mathcal{M}$-morphism.
“⇒” Given \( G \Rightarrow H \) via \((p, m)\) then we have the back faces as pushouts in the following cube. We construct the pushout \( \overline{D} \) over \( \overline{D} \leftarrow \overline{K} \rightarrow \overline{K} \) and obtain the cospan direct transformation \( G \Rightarrow H \) via \((\overline{p}, m)\) by pushout composition.

\[
\begin{array}{c}
\begin{tikzpicture}[scale=0.7, baseline=(current bounding box.center)]
  \node (L) at (0,0) {$L$};
  \node (R) at (2,0) {$R$};
  \node (K) at (1,-2) {$K$};
  \node (D) at (1,-4) {$D$};
  \node (G) at (-1,-4) {$G$};
  \node (H) at (3,-4) {$H$};
  \draw[->] (L) -- (K);
  \draw[->] (R) -- (K);
  \draw[->] (K) -- (D);
  \draw[->] (D) -- (G);
  \draw[->] (D) -- (H);
\end{tikzpicture}
\end{array}
\]

“⇐” Given \( G \Rightarrow H \) via \((\overline{p}, m)\) we have the front faces as pushouts in the above cube. We construct \( D \) as pushout of \( G \leftarrow L \rightarrow \overline{K} \), get an induced morphism \( \overline{D} \rightarrow \overline{D} \) and by pushout decomposition \( \overline{D} \) is the pushout of \( \overline{D} \leftarrow \overline{K} \rightarrow \overline{K} \). Similarly, we can construct a pushout \( \overline{D} \) in the right hand side of the cube, and by uniqueness of pushout complements with \( K \rightarrow \overline{K} \in M \) we have that \( D \) and \( \overline{D} \) are isomorphic leading to the cospan direct transformation \( G \Rightarrow H \) via \((\overline{p}, m)\).

Starting with a classical production \( p = (L \leftarrow K \rightarrow R) \) we obtain an adjoint cospan production \( \overline{p} = (L \rightarrow \overline{K} \leftarrow R) \) by pushout construction in (1), and vice versa \( p \) is obtained by \( \overline{p} \) by pullback construction as in the second step of the construction of the closure.

**Definition 2.4 (Adjoint Productions).** A classical production \( p = (L \leftarrow K \rightarrow R) \) and a cospan production \( \overline{p} = (L \rightarrow \overline{K} \leftarrow R) \) are adjoint to each other if diagram (1) is both pushout and pullback.

\[
\begin{array}{c}
\begin{tikzpicture}[scale=0.7, baseline=(current bounding box.center)]
  \node (L) at (0,0) {$L$};
  \node (R) at (2,0) {$R$};
  \node (K) at (1,-2) {$K$};
  \draw[->] (L) -- (K);
  \draw[->] (R) -- (K);
  \draw[->] (K) -- (L);
\end{tikzpicture}
\end{array}
\]
The most interesting question now is how cospan transformations and classical transformations are related to each other. In the following, we show that direct transformations and hence also transformation sequences correspond to each other uniquely (up to isomorphism) for graphs and weak adhesive HLR categories.

**Theorem 2.5 (Equivalence of Cospan and Classical DPO Transformation).** Given adjoint productions \( p = (L \xleftarrow{l} K \rightarrow R) \) and \( \overline{p} = (L \rightarrow \overline{K} \xleftarrow{r} R) \), and a match \( m : L \rightarrow G \) then we have the following equivalence of direct transformations in the cospan and in the classical DPO approach:

\[
G \xRightarrow{p,m} H \iff G \Rightarrow H \text{ via } (p,m).
\]

**Proof.** \( \Rightarrow \) Given \( G \xRightarrow{p,m} H \), we have the front faces and the top face as pushouts and pullbacks in the following cube. Now we construct \( D \) as pullback in the bottom face, and all morphisms in the top and bottom faces are \( M \)-morphisms. The bottom pullback implies a morphism \( K \rightarrow D \) such that the cube commutes, and the cube pushout-pullback lemma (see [7]) implies that also the back faces are pushouts. These pushouts lead to the direct transformation \( G \Rightarrow H \text{ via } (p,m) \).

\[ \]

\( \Leftarrow \) Given \( G \Rightarrow H \text{ via } (p,m) \), we have the back faces and the top face of the above cube as pushouts and pullbacks. Now we construct \( \overline{D} \) as pushout in the bottom face, and get an induced morphism \( \overline{K} \rightarrow \overline{D} \) such that the cube commutes. By pushout composition and decomposition also the front faces are pushouts. These pushouts lead to the cospan direct transformation \( G \xRightarrow{\overline{p},m} H \). \( \square \)

This result leads to the equivalence of the cospan and the classical DPO approach.

**Theorem 2.6 (Equivalence of Cospan and Classical DPO Approach).** For each concept and result in the classical DPO approach there is an adjoint concept and result in the cospan DPO approach, and vice versa.
Proof Idea. The equivalence is based on the adjointness of productions (see Def. 2.4) and the equivalence of direct transformations defined by the pushout-pullback cube in the proof of Thm. 2.5.

In the following, we illustrate Thm. 2.6 for the concepts of gluing condition and parallel independence and the corresponding results in both approaches.

**Fact 2.7** (Equivalence of Gluing Conditions). Given adjoint productions \( p = (L \xleftarrow{l} K \xrightarrow{r} R) \) and \( \overline{p} = (L \xrightarrow{\overline{l}} K \xleftarrow{\overline{r}} R) \), and a match \( m : L \to G \) then \( m \) satisfies the cospan gluing condition w.r.t. \( \overline{p} \) (see Def. 1.2) if and only if \( m \) satisfies the classical gluing condition w.r.t. \( p \).

**Proof.** For the boundary \( B \) of \( m \), we have to show the equivalence of the classical gluing condition (\( C \)) \( \exists b^* : B \to K : l \circ b^* = b \) and the cospan gluing condition (\( \overline{C} \)) \( \exists b^* : B \to R : \overline{r} \circ b^* = \overline{l} \circ b \).

\[
\begin{align*}
&\text{“(C) } \Rightarrow \text{(C}): \text{ Define } \overline{b^*} := r \circ b^*, \text{ then we have that } \overline{r} \circ \overline{b^*} = \overline{l} \circ l \circ b^* = \overline{l} \circ b. \\
&\text{“(\overline{C}) } \Rightarrow \text{(C)}: \text{ (C) and (1) being a pullback implies that there is a unique } b^* \text{ with } r \circ b^* = \overline{b^*} \text{ and } l \circ b^* = b. 
\end{align*}
\]

Note that Fact 1.3 in Section 1 is the adjoint result corresponding to the well-known construction and uniqueness of direct transformations in the classical DPO approach (see [7]). The proof of Fact 1.3 follows directly from Thm. 2.5 and Fact 2.7, but could also be given directly in the cospan DPO approach.

**Fact 2.8** (Equivalence of Parallel Independence). Two cospan direct transformations \( G \Rightarrow H_i \) via \( (\overline{p}_i, m_i) \) for \( i = 1, 2 \) are parallel independent (see Def. 1.4) if and only if the adjoint direct transformations \( G \Rightarrow H_i \) via \( (p_i, m_i) \) are parallel independent in the classical DPO approach.

**Proof.** Given the cospan direct transformations \( G \Rightarrow H_i \) and the adjoint direct transformations \( G \Rightarrow H_i \) for \( i = 1, 2 \) as shown below, we have to show that the conditions for parallel independence in the classical case (\( C \)) \( \exists i : L_2 \to D_1 \):
\( f_1 \circ i = m_2 \land \exists j : L_1 \to D_2 : f_2 \circ j = m_1 \) and in the cospan approach \((C)\) \( \exists m'_1 : L_1 \to H_2 : y_2 \circ m'_1 = x_2 \circ m_1 \land \exists m'_2 : L_2 \to H_1 : y_1 \circ m'_2 = x_1 \circ m_2 \) are equivalent.

\[ (C) \Rightarrow (\overline{C}) \] Define \( m'_2 := g_1 \circ i, \) then we have that \( y_1 \circ m'_2 = y_1 \circ g_1 \circ i = x_1 \circ f_1 \circ i = x_1 \circ m_2, \) and analogously \( m'_1 := g_2 \circ j \) leading to \((\overline{C}).\)

\[ (\overline{C}) \Rightarrow (C) \] The left bottom face being a pullback and \((\overline{C})\) implies that there exists a unique \( i \) such that \( g_1 \circ i = m'_2 \) and \( f_1 \circ i = m_2, \) and analogously there is \( j \) leading to \((C).\)

\[ \square \]

Thm. 1.5 in Section 1 is the adjoint result corresponding to the well-known local Church-Rosser Theorem in the classical DPO approach (see [7]). The proof of Thm. 1.5 follows directly from Thm. 2.6, Fact 2.8, and the similar equivalence of sequential independence, but could also be given directly in the cospan DPO approach.

3 Conclusion

In this paper, we have shown that similar to the classical DPO approach based on productions as spans \( p = (L \leftarrow K \rightarrow R) \) there is an alternative cospan DPO approach based on productions as cospans \( \overline{p} = (L \rightarrow K \leftarrow R). \) We have presented some basic concepts and results in this alternative approach in Section 1 and have shown in Section 2 in which sense both approaches are equivalent.

It remains open to analyse the benefits of the cospan DPO approach in more detail from the theoretical and the practical point of view. Especially it is interesting to note that Thm. 2.6 corresponds in some sense to the duality principle in category theory, where spans are replaced by cospans but not pushouts by pullbacks.
References


**THE LOGIC IN COMPUTER SCIENCE COLUMN**

**BY**

**YURI GUREVICH**

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

---

**THE LOGIC OF INFONS**

Yuri Gurevich   Itay Neeman*

---

**Abstract**

Infons are pieces of information. In the context of access control the logic of infons is a conservative extension of intuitionistic logic. Distributed Knowledge Authorization Language uses additional unary connectives “p said” and “p implied” where p ranges over principals. Here we investigate infon logic and a narrow but useful primal fragment of it. In both cases, we develop model theory and analyze the derivability problem: Does the given query follow from the given hypotheses? Our more involved technical results are on primal infon logic. We construct an algorithm for the multiple derivability problem: Which of the given queries follow from the given hypotheses? Given a bound on the quotation depth of the hypotheses, the algorithm works in linear time. We quickly discuss the significance of this result for access control.

---

*Department of Mathematics, UCLA, Los Angeles, CA 90095, USA
The Bulletin of the EATCS

1 Introduction

Infons are pieces of information, e.g. John has the right to read File 13 of the given computer system. The notion of infon is basic in DKAL, Distributed Knowledge Authorization Language, that we have been developing [8, 9, 10].

Quisani\(^1\): How are infons different from statements?

Authors\(^2\): We never ask whether an infon is true or false. Instead we ask whether it is known to relevant principals. For example, does the owner of File 13 know that John has the right to read the file? Does the administrator of the system know? Does John know? Besides the notion of infon is purely semantical.

Q: Did you invent the term?

A: No, though for a while we thought we did. The term has been used in situation theory where the name is “intended to emphasize the fact that ‘infons’ are semantic objects, not syntactic representations” [7, page 22]. But infons are assigned truth values in situation theory, so our usage of the term is different.

One may study the algebra of infons. Order \(a \leq b\) if the information in \(a\) is a part of that in \(b\). At the bottom of that partial order are uninformative infons carrying no information whatsoever. There is a natural union operation \(a + b\) on infons. You know \(a + b\) if you know \(a\) and you know \(b\). Infon \(a + b\) is the least upper bound of \(a\) and \(b\). But one has to be careful with the information order because of the omniscience problem well known in epistemic logic. The partial order is intractable. Indeed, valid logic statements are uninformative. In [8] we used a rule-based feasible approximation of the true information order.

The right tool to deal with infons is logic. The addition operation \(a + b\) can be thought as conjunction \(a \land b\). And the implication connective is natural. If you know \(a\) and you know \(a \rightarrow b\) then you know \(b\). In terms of the information order, \(a \rightarrow b\) is the least solution \(x\) of the inequality \(a + x \geq b\).

Q: Give an example of implication.

A: You have the right to fish in this river if you have an appropriate licence from the state.

\(^1\)Quisani is an inquisitive friend of ours.

\(^2\)Speaking one at a time.
Implication allowed us to simplify DKAL [10]. For example “principal $p$ is trusted on saying infon $x$,” used to be a primitive notion in DKAL subject to a stipulation “$x \leq (p \text{ said } x) + (p \text{ trusted on saying } x).$” Now we define “$p$ trusted on saying $x$” as “$(p \text{ said } x) \rightarrow x.” The stipulation follows. It turns out that infon logic is a conservative extension of (disjunction-free, propositional) intuitionistic logic. In addition to conjunction and implication, infon logic has two unary connectives “$p$ said” and “$p$ implied” for any principal $p$. The number of principals is unbounded.

**Q:** How is infon logic related to authorization?

**A:** In DKAL, a principal uses infon logic to derive consequences from (i) his own knowledge assertions and (ii) the communications from other principals.

**Q:** How are “said” and “implied” different?

**A:** A principal $p$ may just say $x$ to $q$. Then (if the communication reaches $q$ and $q$ accepts it but let us ignore all that) $q$ learns infon $p$ said $x$. However $p$ may condition his saying $x$ on $q$ knowing $y$. Then $q$ learns infon $y \rightarrow (p \text{ implied } x)$.

**Q:** Give me an example.

**A:** Suppose that Alice and Bob work in the same company, and Charlie is an outside visitor. If Alice tells Bob that Charlie can read the latest company letter then Bob learns this: Alice said Charlie can read the letter. But if Alice tells Bob that Charlie can read the letter provided that Charlie signed the non-disclosure agreement (NDA) then Bob learns this: if Charlie signed the NDA then Alice implied that Charlie can read the letter.

**Q:** What does this buy you? Suppose that Bob knows the proviso that Charlie signed the NDA. Then he learns that Alice implied that Charlie can read the letter. Shouldn’t he learn that Alice said that Charlie can read the letter?

**A:** If one is not careful, a proviso can be used for undesired delegation and even a probing attack [10, §7]. To get from knowing $p$ said $x$ to knowing $x$, $q$ needs to trust $p$ on saying $x$. To get from knowing $p$ implied $x$ to knowing $x$, $q$ needs to trust $p$ on implying $x$ which is a stronger condition.

The derivability problem (whether given formulas $\Gamma$ entail another given formula $x$) for intuitionistic logic may seem to be intractable. Even the termination of the proof search does not seem to be guaranteed. But there are intuitionistic calculi with the subformula property: if $\Gamma$ entails $x$ then there is a derivation of $x$
from $\Gamma$ that uses only subformulas of formulas $\Gamma \cup \{x\}$. This helps to establish that the problem is solvable in polynomial space. The problem is in fact polynomial space complete [16].

Starting with a known sequent calculus for intuitionistic logic which is sound and complete with respect to the standard Kripke semantics and which has the subformula property, we extend the calculus and semantics to infon logic and prove that the extended calculus is sound and complete and has the subformula property. The derivability problem remains polynomial space complete in the worst case. This does not make infon logic useless. It works in great many practical cases. Typical cases are far from the worst ones.

Further, we identified a narrow primal fragment of infon logic that is surprisingly expressive. For the sake of contrast, the original infon logic may be called full. We modify the sequent calculus for full infon logic to fit primal infon logic. The new calculus has a version of subformula property: instead of subformulas we speak of relatives of theirs which we call local (to the given hypotheses and query) formulas. The new calculus is sound and complete with respect to Kripke semantics adjusted in a simple and natural way. The definition of when an implication $x \rightarrow y$ holds in a world $w$ becomes non-deterministic and that’s it.

Q: What about primal intuitionistic logic? I mean the intersection of primal infon logic with intuitionistic logic. It’s got to be known. Intuitionistic logic has been researched so thoroughly for such a long time.

A: Well, the only references of relevance that we know are proof-theoretic papers [1] and [2]. Their “semi-implication” is our primal implication. But even the two papers have not been published yet; we learned of them when we presented infon logic at Tel Aviv University. Primal intuitionistic logic is weak indeed. That may explain the lack of attention.

The more involved technical results of this paper are related to the derivability problem for primal infon logic.

**Definition 1.1.** The *multiple derivability problem* $\text{MD}(L)$ for a logic $L$ is to compute, given formulas $x_1, \ldots, x_m$ (the hypotheses) and $y_1, \ldots, y_n$ (the queries), which of the queries are derivable from the hypotheses.

We construct an algorithm that solves the multiple derivability problem for primal infon logic. We stratify primal logic according to the quotation depth of infons. For example, both infons “Alice said that Bob implied that John has the right to read File 13” and “Bob said that John has the right to read File 13” belong to stratum 2 but only the second belongs to stratum 1. At each stratum, our algorithm works in linear time. (Our computation model is the usual random access machine, as in [6].)
In applications quotation depth tends to be small. Many authorization languages do not even allow nesting the pronouncements of principals. This applies to all Datalog-based authorization languages that we know.

Q: The lowest stratum where pronouncements is nested is stratum 2. Is stratum 2 of primal logic of any use for authorization purposes?

A: Very much so. In our experience, stratum 2 suffices for most purposes.

Q: Is there any evidence beyond your own experience?

A: An authorization language SecPAL [3] expresses many important access control scenarios. A rather natural translation of SecPAL to DKAL uses only stratum 2 of primal infon logic.

This paper is self-contained. We do not presume that the reader is familiar with DKAL or intuitionistic logic.

Acknowledgments

Conversations with Nikolaj Bjørner, Andreas Blass and Grigori Mints have been most helpful. Last-minute remarks of Sergio Mera were useful as well.

2 Sequent calculi for full and primal infon logic

In DKAL, infons are expressed by means of terms, and formulas have the form “principal knows infon”. Primitive terms have the form $t_0 A(t_1, \ldots, A_k)$ where $A$ is an attribute name, each $t_i$ is a term in the sense of first-order logic, and $t_0$ is of type Principal. Compound terms are built from primitive ones by means of functions $\land$ and $\rightarrow$ of type Infon $\times$ Infon $\rightarrow$ Infon and functions said and implied of type Principal $\times$ Infon $\rightarrow$ Infon.

This paper is devoted to infon logic. (Readers interested primarily in DKAL and not so much in infon logic *per se* may want to go directly to [10].) Here we treat infon terms as propositional formulas. We use symbol $\top$ to represent an uninformative infon. We presume an infinite vocabulary of infon variables and another infinite vocabulary of (the names of) principals. Formulas are built from infon variables and the infon constant $\top$ by the following means.

- Conjunction. If $x, y$ are formulas then so is $x \land y$.

- Implication. If $x, y$ are formulas then so is $x \rightarrow y$. 
Two unary connectives $p$ said and $p$ implied for every principal $p$. If $x$ is a formula then so are $p$ said $x$ and $p$ implied $x$.

In the sequel, formulas are by default these formulas. Formulas that do not involve the unary connectives will be called quotation-free.

Some of our algorithms take formulas or sequences of formulas as inputs. In this connection, we presume that the syntax of formulas is such that an occurrence of a subformula $y$ in a formula $x$ is uniquely defined by the position of the first symbol of $y$ in $x$.

### 2.1 Sequent calculus for full infon logic

Our sequent calculus SCF for full infon logic is essentially an extension of the disjunction-free version of the intuitionistic propositional system NJp [14, §2.2]. A sequent has the form $\Gamma \vdash x$ where $x$ is a formula and $\Gamma$ is a set of formulas written as a sequence. Here are the axioms and rules of inference.

#### Axioms

\[
\begin{align*}
(\top) & \quad \top \vdash \top \\
(x2x) & \quad x \vdash x
\end{align*}
\]

#### Inference rules

\[
\begin{align*}
\text{(Premise Inflation)} & \quad \Gamma \vdash y \quad \Gamma, x \vdash y \\
\text{(\land E)} & \quad \frac{\Gamma \vdash x \land y}{\Gamma} \\
\text{(\land I)} & \quad \frac{\Gamma \vdash x \quad \Gamma \vdash y}{\Gamma \vdash x \land y} \\
\text{(\rightarrow E)} & \quad \frac{\Gamma \vdash x \quad \Gamma \vdash x \rightarrow y}{\Gamma \vdash y} \\
\text{(\rightarrow I)} & \quad \frac{\Gamma, x \vdash y}{\Gamma \vdash x \rightarrow y}
\end{align*}
\]

155
Here E and I allude to “elimination” and “introduction” respectively. If \( \Delta = \{x_1, \ldots, x_n\} \) then “\( q \) said \( \Delta \)” is the set \( \{q \) said \( x_i) : i = 1, \ldots, n\} \) and “\( q \) implied \( \Delta \)” is the set \( \{q \) implied \( x_i) : i = 1, \ldots, n\} \).

**Corollary 2.1.** \( q \) said \( x \) \( \vdash \) \( q \) implied \( x \).

**Proof**  Apply rule (Implied) to \( x \vdash x \). \( \square \)

**Corollary 2.2** (Transitivity). If \( \Gamma \vdash x \) and \( \Gamma, x \vdash y \) then \( \Gamma \vdash y \).

**Proof**  By \((\to I)\), we have \( \Gamma \vdash (x \to y) \). It remains to apply \((\to E)\). \( \square \)

**Q:** I guess \( \top \) is the propositional constant usually called “true.”

**A:** This is a reasonable point of view as far as this paper is concerned.

**Q:** Is there any semantical difference between \( \top \) and “true?”

**A:** Yes, \( \top \) is a piece of information known to all principals. It is natural to assume that this infon is true but we don’t have to lump all true infons together. Some true infons may be informative.

**Q:** Rule (Implied) is too complicated. Turn Corollary 2.1 into an axiom and then replace (Implied) with a simpler rule \( \Delta \vdash y \quad \frac{}{q \text{ implied } \Delta \vdash q \text{ implied } y} \).

**A:** Calculus SCF has the subformula property: If a sequent \( \Gamma \vdash x \) is provable then it has a proof that uses only subformulas of the sequent formulas; see Theorem 3.2 below. The modified system loses the subformula property. For example any proof of sequent

\[ s = [q \text{ said } r \text{ said } x \vdash q \text{ implied } r \text{ implied } x]. \]

involves additional formulas.
2.2 Sequent calculus for primal infon logic

We introduce primal infon logic by means of a sequent calculus SCP obtained from the calculus SCF for full infon logic by replacing the implication-introduction rule \((\rightarrow I)\) with two rules

\[
\frac{\Gamma \vdash y}{\Gamma \vdash (x \rightarrow y)} \quad (\rightarrow IW)
\]

\[
\frac{\Gamma \vdash x, \; \Gamma, x \vdash y}{\Gamma \vdash y} \quad (\text{Trans})
\]

Rule \((\rightarrow IW)\) is a weaker (hence the “W”) implication-introduction rule. Rule \((\text{Trans})\) reflects Corollary 2.2.

Lemma 2.3. In either calculus, if \(\Gamma \cup \{x_1, \ldots, x_n\} \vdash y\) and \(\Gamma \vdash x_1, \ldots, \Gamma \vdash x_n\) then \(\Gamma \vdash y\).

Proof Induction on \(n\). If \(n = 0\), the lemma is trivial. Suppose that \(n > 0\) and the lemma has been proved for \(n - 1\). Let \(\Gamma' = \Gamma \cup \{x_1, \ldots, x_{n-1}\}\). Use \((\text{Premise Inflation})\) to derive \(\Gamma' \vdash x_n\) from the last premise. The first premise is \(\Gamma', x_n \vdash y\). Now use \((\text{Trans})\) and then the induction hypothesis. □

Q: Rule \((\rightarrow IW)\) is ridiculous. Why do you need the implication if you already have the conclusion?

A: A principal may know the conclusion \(y\) but may be willing to share only an implication \(x \rightarrow y\) with another principal. Besides the implication \(x \rightarrow y\) may be needed for derivation as it may occur as the premise of another implication.

3 Semantics, soundness and completeness, subformula property

3.1 Semantics for full infon logic

Recall that a quasi-order \(\leq\) is a binary relation that is reflexive and transitive. A subset \(U\) of a quasi-ordered set \((W, \leq)\) is a cone if \(u \in U\) and \(u \leq v\) imply \(v \in U\) for all \(u, v\) in \(W\). A Kripke structure for propositional intuitionistic logic [12] can be defined as a triple \((W, \leq, C)\) where
W is a nonempty set whose elements are traditionally called worlds,
\( \leq \) is a quasi-order of W, and
\( C \) assigns a cone to every infon variable.

By induction, every quotation-free formula \( z \) is assigned a cone \( C(z) \) of worlds:

\[ K4. \quad C(\top) = W. \]
\[ K5. \quad C(x \land y) = C(x) \cap C(y). \]
\[ K6. \quad C(x \rightarrow y) = \{ u : C(x) \cap \{ v : v \geq u \} \subseteq C(y) \}. \]

Here \( \top \) plays the traditional role of propositional constant “true.” It is easy to check, by induction on formula \( z \), that every \( C(z) \) is indeed a cone.

We extend this definition to accommodate full infon logic. A Kripke structure for full infon logic is a quintuple \((W, \leq, C, S, I)\) where \( W, \leq, C \) are as above and where \( S \) and \( I \) assign binary relations \( S_q \) and \( I_q \) over \( W \) respectively to every principal \( q \) in such a way that the following two requirements are satisfied.

\[ K7. \quad I_q \subseteq S_q. \]
\[ K8. \quad \text{If } u \leq w \text{ and } w S_q v \text{ then } u S_q v, \text{ and the same for } I_q. \]

The cone map \( C \) is extended to all formulas by means of clauses \( K4–K6 \) and the following two clauses.

\[ K9. \quad C(q \text{ said } x) = \{ u : \{ v : u S_q v \} \subseteq C(x) \}. \]
\[ K10. \quad C(q \text{ implied } x) = \{ u : \{ v : u I_q v \} \subseteq C(x) \}. \]

If \( u \in C(z) \), we say that \( z \) holds in \( u \) and that \( u \) models \( z \), and we write \( u \models z \).

Again, it is easy to check, by induction on formula \( z \), that every \( C(z) \) is indeed a cone. We consider here only the case when \( z = (q \text{ said } x) \). Suppose that \( u \in C(z) \) and \( u \leq w \). By \( K9 \), \( \{ v : u S_q v \} \subseteq C(z) \) and we need to show that \( \{ v : w S_q v \} \subseteq C(z) \). This follows from \( K8 \).

The cone map \( C \) extends naturally to sets of formulas and to sequents:

- \( C(\Gamma) = \bigcap_{x \in \Gamma} C(x) \).
- \( C(\Gamma \vdash y) = \{ u : C(\Gamma) \cap \{ v : v \geq u \} \subseteq C(y) \} \).

The Kripke structure itself \textit{models} a sequent \( s = [\Gamma \vdash y] \) if \( C(s) = W \) which is equivalent to \( C(\Gamma) \subseteq C(y) \). A sequent \( s \) is \textit{valid} if every Kripke structure models \( s \).

\textbf{Corollary 3.1.} \( C(q \text{ said } x) \subseteq C(q \text{ implied } x) \)

\textit{Proof} By \( K9 \) and \( K10 \), it suffices to show that, for every \( u \), \( \{ v : u I_q v \} \subseteq \{ v : u S_q v \} \). This is exactly \( K7 \). \qed
3.2 Semantics for primal infon logic

The definition of Kripke structures for primal infon logic is similar to that for full infon logic except that the deterministic requirement K4 is replaced with the following non-deterministic requirement.

K6W. \( C(x \to y) \) is an arbitrary cone subject to constraint

\[
C(y) \subseteq C(x \to y) \subseteq \{ u : C(x) \cap \{ v : v \geq u \} \subseteq C(y) \}.
\]

3.3 Soundness and completeness

Theorem 3.2. In the case of either infon logic, full or primal, the following claims are equivalent for any sequent \( s \).

1. \( s \) is provable.
2. \( s \) is valid.
3. Every finite Kripke structure models \( s \).
4. There is a proof of \( s \) that uses only subformulas of \( s \).

Proof. We deal with both logics at once. We prove that \((1) \implies (2) \implies (3) \implies (4) \implies (1)\). Implications \((4) \implies (1)\), and \((2) \implies (3)\) are obvious.

\((1) \implies (2)\). Let \( K = (W, \leq, C, S, I) \) be an arbitrary Kripke structure. By induction on the given derivation of sequent \( s \) we show that \( K \) models \( s \). If \( s \) is the \((\top)\) axiom, use K4. If \( s \) is an \((x_2x)\) axiom, the desired \( C(x) \subseteq C(x) \) is obvious.

Suppose that \( s \) is not an axiom. Let \( u \) be a world in the cone of the premise of \( s \). We need to show that \( u \) models the conclusion of \( s \). Consider the last step in the given derivation of \( s \). Several cases arise. The case of (Premise Inflation) is obvious. The cases \((\wedge E)\) and \((\wedge I)\) are obvious as well; just use K5.

\((\to E)\). By the induction hypothesis, \( u \) models \( x \) as well as \( x \to y \). By K6W (use the second inclusion of the constraint), \( u \) models \( y \).

\((\to I)\). This case is relevant for the full but not primal infon logic. By K6, it suffices to check that \( v \models x \) implies \( v \models y \) for all \( v \geq u \). Suppose \( v \models x \). By the induction hypothesis, \( v \models y \).

\((\to IW)\). This case is relevant for primal infon logic. By the induction hypothesis, \( u \) models \( y \). By K6W (use the first inclusion of the constraint), \( u \) models \( x \to y \).

(Trans). This case is relevant for primal infon logic. By the choice of \( u \), we have \( u \in C(\Gamma) \). By the induction hypothesis applied to the first premise, \( u \in C(x) \). Now we apply the induction hypothesis to the second premise: \( u \in C(y) \).
(S) and (I). These two cases are similar. We consider only (I). By the choice of \( u \), we have \( u \in C(q \text{ said } \Delta_1) \cap C(q \text{ implied } \Delta_2) \). By Corollary 3.1, \( u \in C(q \text{ implied } (\Delta_1 \cup \Delta_2)) \) which, by K10, is equivalent to this: \( v \in C(\Delta_1 \cup \Delta_2) \) for all \( v \) with \( u \vdash v \). For any such \( v \), by the induction hypothesis, \( v \in C(y) \). By K10, \( u \in C(q \text{ implied } y) \).

(3) \( \implies (4) \). Assuming that (4) fails, we construct a finite model \( K = (W, \leq, C, S, I) \) for sequent \( x \). Call a formula \textit{native} if it is a subformula of \( x \). A \textit{native theory} \( v \) is any set of native formulas closed under native deduction in the following sense: \( u \) contains every native formula \( x \) such that sequent \( u \vdash x \) is provable using only native formulas.

The quasi-order \((W, \leq)\) of \( K \) is the set of native theories ordered by inclusion. Set \( u \vdash v \) true if \( v \) contains every formula \( x \) such that \( (q \text{ implied } x) \) belongs to \( u \) or \( (q \text{ said } x) \) belongs to \( u \). Set \( u \preceq v \) true if \( v \) contains every formula \( x \) such that \( q \text{ said } x \) belongs to \( u \). Requirements K1–K2 and K7–K8 are obviously satisfied. It remains to define the cone map \( C \). In the case of full infon logic, we could have defined \( C \) on the variables and then used clauses K4–K6 and K9–K10 to extend \( C \) to compound formulas. For the uniformity of the proof, we choose a different route.

If \( z \) is a native formula or \( \top \) define \( C(z) = \{ u : z \in u \} \), so that \( u \vdash z \) if and only if \( z \in u \). Clearly requirements K3 and K4 are satisfied. Now use clauses K5–K6 and K9–K10 to extend \( C \) to the remaining formulas composed from native formulas and \( \top \) by means of connectives \( \land, \rightarrow, q \text{ said }, q \text{ implied} \). To complete the definition of \( K \), we check that requirements K5, K6 (resp. K6W) and K9–K10 on \( C(z) \) are satisfied for any native formula \( z \). This is done by induction on \( z \). The base of induction, when \( z \) is a variable, is trivial. The induction step splits into several cases.

Case K5. Using the definition of \( C \) on native formulas and the fact that every world is closed under native deduction, we have

\[
\begin{align*}
  u \in C(x \land y) & \iff (x \land y) \in u \\
  x \in u \land y \in u & \iff u \in C(x) \cap C(y).
\end{align*}
\]

Case K6. First we prove that \( C(x \rightarrow y) \subseteq \{ u : C(x) \cap \{ v : v \geq u \} \subseteq C(y) \} \). This part is relevant to both infon logics. Suppose that \( u \) contains \( x \rightarrow y \). If a native theory \( v \geq u \) also contains \( x \) then it contains \( y \).

Second we prove that \( \{ u : C(x) \cap \{ v : v \geq u \} \subseteq C(y) \} \subseteq C(x \rightarrow y) \). This part is relevant only to full infon logic. Pick an arbitrary world \( u \) such that \( C(x) \cap \{ v : v \geq u \} \subseteq C(y) \). We claim that sequent \( u, x \vdash y \) is provable. Otherwise, there is a native theory \( v \) that includes \( u \), contains \( x \) but does not contain \( y \) which contradicts the choice of \( u \). By \((\rightarrow I)\) with \( \Gamma = u \), we have that \( u \) contains \( x \rightarrow y \).
Case K6W. This case is relevant for primal infon logic. The right inclusion has been already proven. To prove the left inclusion, suppose that $u \in C(y)$, so that $u \vdash y$ is provable. By $(\rightarrow IW)$, $u \vdash (x \rightarrow y)$ is provable, and so $u \in C(x \rightarrow y)$.

Cases K9 and K10. The two cases are similar; we consider only case K9. Consider an arbitrary world $u$. First suppose that $u \in C(q \text{ said } x)$, that is $u$ contains $(q \text{ said } x)$. We need to prove that any $v$ with $u S_q v$ contains $x$. This follows from the definition of $S_q$. Second suppose that $\{ v : u S_q v \} \subseteq C(x)$, that is every native theory $v$ with $u S_q v$ contains $x$. Let $\Delta$ be the set of formulas $y$ such that $u$ contains $(q \text{ said } y)$, and let $\Delta'$ be the least native theory that includes $\Delta$. By the definition of $S_q$, we have $u S_q \Delta'$. Then $\Delta'$ contains $x$, so that the sequent $\Delta' \vdash x$ is provable. By rule (Said), sequent $u \vdash (q \text{ said } x)$ is provable, and so $u$ contains $q \text{ said } x$.

Thus $K$ is a legitimate Kripke structure. Finally let $s$ be $\Gamma \vdash x$, and let $\Gamma^\ast$ be the least native theory. By the assumption that (4) fails, $\Gamma^\ast$ does not contain $x$. It follows that $s$ fails in $K$.  

4 Full infon logic: complexity

Theorem 4.1. The validity problem (whether a given formula is valid) for full infon logic is polynomial-space complete.

The rest of this section is devoted to proving the theorem. The quotation-free fragment of our sequent calculus for full infon logic is a calculus for a fragment of propositional intuitionistic logic whose validity problem is pspace (that is polynomial space) hard [16]. Hence our problem is pspace hard. It remains to show that the validity problem for full infon logic is pspace.

We use the following idea that goes back to Ladner [13] who proved that the validity problem for some modal logics is pspace. Instead of checking validity, check whether the given formula can be refuted “in a tree-like model structure,” which may be exponentially large but where the branches have only polynomial length and thus “can be constructed one branch at a time.” The idea was developed in a variety of publications, in particular by Halpern and Moses [11] and by Schröder and Pattinson [15]. The latter paper is quite recent, has a survey of related literature and will delight the fans of category theory.

Instead of constructing a tree-like model structure and examining it branch by branch, we use games; this helps to avoid bookkeeping. Recall that pspace equals alternating polynomial time [5]. We show that the unprovability problem for the sequent calculus SCF for full infon logic, whether a given sequent is unprovable, is solvable in alternating polynomial time.

We start with a few auxiliary definitions. A sequent $s = [\Gamma \vdash \varphi]$ self-refuting if $\Gamma$ is closed under $s$-native deduction but does not contain $\varphi$. A formula is potent
if it has the form \( x \rightarrow y \) or \( p \text{ said } x \) or \( p \text{ implied } x \). Define quotation depth in the obvious way:

\[
\begin{align*}
\text{QD}(z) &= 0 \text{ if } z \text{ is a variable or } T, \\
\text{QD}(x \land y) &= \text{QD}(x \rightarrow y) = \max(\text{QD}(x), \text{QD}(y)), \\
\text{QD}(p \text{ said } x) &= \text{QD}(p \text{ implied } x) = 1 + \text{QD}(x), \\
\text{QD}(\Gamma \vdash z) &= \max(\text{QD}(x) : x \in \Gamma \lor x = z).
\end{align*}
\]

**The game** Given a sequent \( s_0 = [\Gamma_0 \vdash \varphi_0] \), we define a game \( G(s_0) \) between two players: Refuter, who intends to refute \( s \), and Challenger. Let \( n \) be the length of \( s_0 \) and \( d \) the quotation depth \( \text{QD}(s_0) \). Notice that the number of \( s_0 \)-native formulas is \( \leq n \).

**Phase 0.** Refuter starts the game by guessing a sequent \( s_1 = [\Gamma_1 \vdash \varphi_1] \) such that \( \Gamma_1 \supseteq \Gamma_0 \) and \( \varphi_1 = \varphi_0 \); Refuter claims that \( s_1 \) is self-refuting.

**Winning condition.** For any \( k > 0 \), the state of the game after phase \( k - 1 \) (unless the game terminated earlier) is given by a sequent \( s_k = [\Gamma_k \vdash \varphi_k] \). The game continues further provided the following conditions are satisfied.

1. **R1** \( \top \in \Gamma_k \), and \( \varphi \notin \Gamma_k \), and every formula in \( \Gamma_k \) is \( s_{k-1} \)-native.
2. **R2** If \( x \land y \) is in \( \Gamma_k \) then both \( x \) and \( y \) are in \( \Gamma_k \).
3. **R3** If \( x, y \) are in \( \Gamma_k \) and \( x \land y \) is \( s_k \)-native then \( x \land y \) is in \( \Gamma_k \).
4. **R4** If \( x \) is in \( \Gamma_k \) and \( x \rightarrow y \) is in \( \Gamma_k \) then \( y \) is in \( \Gamma_k \).

**C1** There is an \( s_k \) native potent formula outside of \( \Gamma_k \).

Otherwise the game terminates. If at least one of the conditions R1–R4 fails then Challenger wins. If conditions R1–R4 hold but condition C1 fails, then Refuter wins.

**Phase number \( k > 0 \).** In state given by sequent \( s_k = [\Gamma_k, \varphi_k] \), Challenger chooses a potent \( s_k \)-native formula \( z \) outside of \( \Gamma_k \).

**Case** \( z = (x \rightarrow y) \). Refuter guesses a sequent \( s_{k+1} = [\Gamma_{k+1} \vdash y] \) with \( \Gamma_{k+1} \supseteq \Gamma_k \cup \{x\} \).

**Case** \( z = (p \text{ said } y) \). Let \( \Delta \) be the set of formulas \( x \) such that formula \( p \text{ said } x \) belongs to \( \Gamma_k \). Refuter guesses a sequent \( s_{k+1} = [\Gamma_{k+1} \vdash y] \) where \( \Gamma_{k+1} \) consists of formulas native to \( \Delta \cup \{y\} \) and \( \Gamma_{k+1} \supseteq \Delta \).

**Case** \( z = (p \text{ implied } y) \). Let \( \Delta \) be the set of formulas \( x \) such that either \( (p \text{ said } x) \) or \( (p \text{ implied } x) \) belongs to \( \Gamma_k \). Refuter guesses a sequent \( s_{k+1} = [\Gamma_{k+1} \vdash y] \) where \( \Gamma_{k+1} \) consists of formulas native to \( \Delta \cup \{y\} \) and \( \Gamma_{k+1} \supseteq \Delta \).
The Bulletin of the EATCS

That completes the description of the game.

**Termination** The game terminates in at most $1 + (d + 1)n$ phases. Indeed, at every phase $k$ where Challenger chooses a said-formula or implied-formula, we have $QD(s_{k}+1) < QD(s_{k})$. Thus there can be at most $d$ such cases. And every stretch of phases where Challenger chooses implications can contain at most $n$ phases as at each such phase the set of hypotheses grows but there can only be $\leq n$ hypotheses as there are only $\leq n$ $s_{0}$-native formulas. The only exception is the very first stretch because the set of hypotheses may not grow during the very first phase.

**Lemma 4.2.**
1. If $s_{0}$ is refutable then Refuter has a winning strategy in $G(s_{0})$.
2. If $s_{0}$ is valid then Challenger has a winning strategy in $G(s_{0})$.

**Proof**
1. Assuming that $s_{0} = [\Gamma_{0} \vdash \varphi_{0}]$ is refutable, we construct a winning strategy for Refuter. At phase 0, Refuter chooses $\Gamma_{1}$ to be the least $s_{0}$-native theory that includes $\Gamma_{0}$, and he chooses $\varphi_{1} = \varphi_{0}$ so that sequent $s_{1} = [\Gamma_{1} \vdash \varphi_{1}]$ is self-refuting.

Suppose that $k > 0$, phase $k - 1$ has been executed, the current sequent $s_{k}$ is self-refuting. The game continues, and Challenger chose a formula $z$. We show that Refuter can reply with a self-refuting sequent $s_{k+1} = [\Gamma_{k+1} \vdash \varphi_{k+1}]$ whose form depends on the form of $z$. But first notice that $s_{k}$ is not valid because it is self-refuting and condition R1 holds. Let $K$ be the counter-model constructed in the proof of Theorem 3.2 with $s_{k}$ playing the role of $s$. Since $s_{k}$ is self-refuting, $\Gamma_{k}$ is an $s_{k}$-native theory and thus a world in $K$.

Case $z = (x \rightarrow y)$. Since $(x \rightarrow y)$ does not belong to $s_{k}$-native theory $\Gamma_{k}$, there is a world $\Gamma_{k+1} \supseteq \Gamma_{k}$ in $K$ that contains $x$ but not $y$. The desired $s_{k} = [\Gamma_{k+1} \vdash y]$.

Case $z = (p \text{ said } y)$. Let $\Delta$ be as in the definition of phase $k > 0$. Taking into account the rule (Said) of calculus SCF for full infon logic, we see that sequent $t = \Delta \vdash y$ is unprovable. The desired $\Gamma_{k+1}$ is the least $t$-native theory that includes $\Delta$ and the desired $\varphi_{k+1} = y$.

Case $z = (p \text{ implied } y)$ is similar to the previous one.

2. Suppose that $s_{0}$ is valid and thus provable in calculus SCF for full infon logic. We construct a winning strategy for Challenger. Suppose that $k > 0$, phase $k - 1$ has been executed and the current sequent $s_{k}$ is provable. If at least one of the clauses R1–R4 in winning definition fails then Challenger wins. Suppose that all four clauses hold. We show that Challenger can choose a potent $s_{k}$-native formula in such a way that every legal response $s_{k+1}$ of Refuter is provable.

Since $s_{k}$ is provable and condition R1 holds, $\Gamma_{k}$ is not closed under $s_{k}$-native deduction. Hence there exist $s_{k}$-native formulas $\psi$ outside of $\Gamma_{k}$ such that sequent $[\Gamma_{k} \vdash \psi]$ is provable in SCF. Challenger should choose a formula $z$ among such formulas $\psi$ subject to an additional constraint: the proof $P$ of sequent $t = [\Gamma_{k} \vdash$
should be minimal possible. Clearly \( t \) is not an axiom of SCF. Let \( R \) be the
inference rule used to obtain \( t \) in \( P \). Taking into account that conditions R2–R4 hold, \( R \) cannot be (Premise Inflation), \((\land E)\), \((\land I)\) or \((\rightarrow E)\). We consider the
remaining cases. In all those remaining cases formula \( z \) chosen by Challenger is
potent.

Case \( R = (\rightarrow I) \), so that \( z \) has the form \( x \rightarrow y \). Refuter guesses a sequent \( s_{k+1} = [\Gamma_{k+1} \vdash y] \) with \( \Gamma_{k+1} \supseteq \Gamma_k \cup \{x\} \). But the premise of (our application of) \( R \) is sequent \( \Gamma_k, x \vdash y \). So \( s_{k+1} \) is provable.

Case \( R = (\text{Said}) \), so that \( z \) has the form \( p \text{ said } x \). Let \( \Delta \) be as in the definition
of phase \( k \). Refuter guesses a sequent \( s_{k+1} = [\Gamma_{k+1} \vdash y] \) where \( \Gamma_{k+1} \) consists of
formulas native to \( \Delta \cup \{y\} \) and \( \Gamma_{k+1} \supseteq \Delta \). But the premise of \( R \) is sequent \( \Delta \vdash y \). \( s_{k+1} \) is provable.

Case \( R = (\text{Implied}) \) is similar to the previous one.

Thus the question whether a given sequent \( s_0 \) is valid or refutable can be de-
cided in alternating polynomial time. That concludes the proof of Theorem 4.1.

## 5 Hilbert-type calculus for primal infon logic

The rest of this paper is devoted to primal infon logic. We introduce a Hilbert-type
calculus \( \mathcal{H} \) for the logic.

Let \( \text{told} \) with or without a subscript range over \{implied, said\}. A string \( \pi \)
of the form \( q_1 \text{told}_1 q_2 \text{told}_2 \ldots q_k \text{told}_k \) is a quotation prefix; the length \( k \) of \( \pi \)
may be zero. Let \( \text{pref} \) with or without a subscript range over quotation prefixes.
If \( x \) is a formula

\[
q_1 \text{told}_1 q_2 \text{told}_2 \ldots q_m \text{told}_m y
\]

where \( y \) is a variable, conjunction or implication then every quotation prefix
\( (q_1 \text{told}_1 q_2 \text{told}_2 \ldots q_k \text{told}_k) \) with \( k \leq m \) is a quotation prefix of \( x \) so that
\( (q_1 \text{told}_1 q_2 \text{told}_2 \ldots q_m \text{told}_m) \) is the maximal quotation prefix of \( x \).

We say that \( \text{pref}_1 \) is dominated by \( \text{pref}_2 \) and write \( \text{pref}_1 \leq \text{pref}_2 \) if \( \text{pref}_1 \)
is the result of replacing some (possibly none) occurrences of \( \text{implied} \) in \( \text{pref}_2 \)
with said. Now we are ready to give the axioms and rules of \( \mathcal{H} \).

**Axioms:**

\[
\text{pref} \top
\]

**Inference rules:**

\[
\frac{\text{pref}_2 x}{\text{pref}_1 x} \quad \text{where} \quad \text{pref}_1 \leq \text{pref}_2
\]
The Bulletin of the EATCS

\[
\begin{array}{c}
(\text{Pref } \land \text{E}) \quad \frac{\text{pref}(x \land y)}{\text{pref } x} \quad \frac{\text{pref}(x \land y)}{\text{pref } y} \\
(\text{Pref } \land \text{I}) \quad \frac{\text{pref } x \quad \text{pref } y}{\text{pref } (x \land y)} \\
(\text{Pref } \to \text{E}) \quad \frac{\text{pref } x \quad \text{pref } (x \to y)}{\text{pref } y} \\
(\text{Pref } \to \text{I}) \quad \frac{\text{pref } y}{\text{pref } (x \to y)}
\end{array}
\]

A derivation of a formula \( \varphi \) from hypotheses \( \Gamma \) in calculus \( \mathcal{H} \) is a sequence \( x_1, \ldots, x_n \) of formulas where \( x_n = \varphi \) and every \( x_i \) is an axiom, a hypothesis or the result of applying one of the rules to earlier members. \( \mathcal{H} \) does not have the subformula property but it has a similar property.

**Q:** How do you know that \( \mathcal{H} \) does not have the subformula property?

**A:** Here is a simple example. Hypotheses

\[
p \text{ said } x, \quad p \text{ said } (x \to y), \quad p \text{ said } (y \to z)
\]

entail \( p \text{ said } z \). Indeed, the first two hypotheses entail formula \( p \text{ said } y \); that formula and the third hypotheses entail \( p \text{ said } z \). That derivation and indeed all derivations of \( p \text{ said } z \) from the hypotheses contain formula \( p \text{ said } y \) which is not a subformula of the hypotheses or the conclusion.

**Definition 5.1.** The **components** of a formula \( z \) are defined by induction:

- \( z \) is a component of \( z \), and
- if \( \text{pref } (x \land y) \) is a component of \( z \) or if \( \text{pref } (x \to y) \) is a component of \( z \), then \( \text{pref } x \) and \( \text{pref } y \) are components of \( z \).

The components of a set \( \Delta \) of formulas are the components of the formulas in \( \Delta \).

**Lemma 5.2** (Components). If \( x \) is a component of \( z \) then \( p \text{ told } x \) is a component of \( p \text{ told } z \).

**Proof** Induction on the definition of components. \( \square \)

Further, a formula \( x \) is dominated by a formula \( y \) if \( x = \text{pref}_1 z \) and \( y = \text{pref}_2 z \) for some \( z \) and \( \text{pref}_1 \leq \text{pref}_2 \).
Definition 5.3 (Local). A formula $x$ is local to a formula $z$ if it is dominated by a component of $z$. Formula $x$ is local to a set $\Delta$ of formulas if it is local to a formula in $\Delta$. A prefix $\text{pref}$ is local to $\Delta$ if some formula $\text{pref} \ y$ is local to $\Delta$. A derivation $x_1, \ldots, x_n$ of $\varphi$ from $\Gamma$ in calculus $\mathcal{H}$ is local if every formula $x_i$ is local to set $\Gamma \cap \{\varphi\}$.

Recall that SCP is the sequent calculus of §2.2 for primal infon logic.

Theorem 5.4. For every sequent $s = [\Gamma \vdash \varphi]$, the following claims are equivalent.

1. $s$ is derivable in SCP.
2. There is a derivation of $s$ in SCP where every formula is a subformula of $s$.
3. $s$ is valid.
4. $\varphi$ is derivable from $\Gamma$ in $\mathcal{H}$.
5. There is a local derivation of $\varphi$ from $\Gamma$ in $\mathcal{H}$.

Proof (1), (2) and (3) are equivalent by Theorem 3.2. (5) obviously implies (4). It suffices to prove that (4) implies (1) and (2) implies (5).

(4) implies (1). By induction on a given derivation of $\varphi$ from $\Gamma$ in $\mathcal{H}$, we prove $\Gamma \vdash \varphi$ in SCP. If $\varphi$ is an axiom $\text{pref} \top$ of $\mathcal{H}$, start with the axiom $\top$ of SCP; repeatedly apply rules (Said) or (Implied) to obtain sequent $\emptyset \vdash (\text{pref} \top)$; then repeatedly apply (Premise Inflation) to obtain sequent $\Gamma \vdash \text{pref} \top$. If $\varphi$ is a hypothesis, use axiom ($x2x$) of SCP and then repeatedly apply (Premise Inflation). Otherwise several cases arise according to the last step in the given derivation of $\varphi$. We consider only the case when rule (Pref→E) was applied at the last step; other cases are similar. By the induction hypothesis, $\Gamma \vdash \text{pref} x$ and $\Gamma \vdash \text{pref}(x \rightarrow y)$ are provable. By Lemma 2.3, it suffices to prove the sequent

$$\Gamma, \text{pref} x, \text{pref}(x \rightarrow y) \vdash \text{pref} y.$$  

Start with an obviously provable sequent $x, (x \rightarrow y) \vdash y$ and apply rules (Said), (Implied) to obtain sequent

$$\text{pref} x, \text{pref}(x \rightarrow y) \vdash \text{pref} y;$$

and then repeatedly apply (Premise Inflation).

(2) implies (5). By induction on a given proof $P$ of sequent $s = [\Gamma \vdash \varphi]$ in SCP that uses only subformulas of $s$, we construct a local derivation of $\varphi$ from $\Gamma$ in $\mathcal{H}$. The cases when $s$ is an axiom are obvious. Otherwise several cases arise according to the last step in $P$. We consider here only two cases.
(→E) Suppose that rule (→E) was applied in the last step of \( P \). By the induction hypotheses, we have local derivations of \( x \) and of \( x \rightarrow y \) from \( \Gamma \) in \( \mathcal{H} \). It remains to apply rule (Pref→) with the empty pref.

(Implied) Suppose that rule (Implied) was applied in the last step of \( P \), so that \( \Gamma \) has the form \((q \text{ said } \Delta_1) \cup (q \text{ implied } \Delta_2)\) and \( \varphi \) has the form \( q \text{ implied } \varphi_0 \). By the induction hypothesis, there is a local derivation \( D \) of \( \varphi_0 \) from \( \Delta_1 \cup \Delta_2 \) in \( \mathcal{H} \). Without loss of generality, \( D \) has the form

\[
\Delta_1, \Delta_2, \text{Tail}
\]

so that first \( \Delta_1 \) formulas are listed, then \( \Delta_2 \) formulas are listed, and then the remaining tail formulas \( z \) are listed. Let \( D' \) be

\[
q \text{ implied } \Delta_1, \ q \text{ implied } \Delta_2, \ q \text{ implied } \text{Tail}.
\]

\( D' \) is a derivation of \( \varphi \) from \( \Gamma' = (q \text{ implied } \Delta_1) \cup (q \text{ implied } \Delta_2) \) in \( \mathcal{H} \). Indeed, if a tail formula \( z \) of \( D \) is an axiom or is obtained from earlier members of \( D \) by means of a rule \( R \) then \( (q \text{ implied } z) \) is an axiom or is obtained from the corresponding members of \( D' \) by rule \( R \). Furthermore, \( D' \) is a local derivation. Indeed, if a tail formula \( x \) of \( D \) is dominated by a component \( y \) of \( \Delta_1 \cup \Delta_2 \cup \{\varphi_0\} \) then \( q \text{ implied } x \) is dominated by \( q \text{ implied } y \) which, by Lemma 5.2, is a component of \( \Gamma' \cup \{\varphi\} \). Further, let \( D'' \) be \((q \text{ said } \Delta_1)\) followed by \( D'\):

\[
q \text{ said } \Delta_1, \ q \text{ implied } \Delta_1, \ q \text{ implied } \Delta_2, \ q \text{ implied } \text{Tail}.
\]

\( D'' \) is the desired local derivation of \( \varphi \) from \( \Gamma \) in \( \mathcal{H} \). First we check that \( D'' \) is a derivation. Since \( D' \) is a derivation of \( \varphi \) from \( \Gamma' \); it remains only to notice that \( q \text{ implied } \Delta_1 \) is obtained from \( q \text{ said } \Delta_1 \) by repeated applications of rule (Pref S2I). In fact, derivation \( D'' \) is local to sequent \( s \). Indeed formulas in blocks \((q \text{ said } \Delta_1)\) and \((q \text{ implied } \Delta_2)\) are local because they are hypotheses. Formulas in block \((q \text{ implied } \Delta_1)\) are local as they are dominated by hypotheses \( q \text{ said } \Delta_1 \). And, since \( D' \) is local to \( \Gamma' \cup \{\varphi\} \), every formula \( z \) in \((q \text{ implied } \text{Tail})\) is dominated either by a component of \( \varphi \) or else by a component of \( \Gamma' \) which is dominated by the corresponding component of \( \Gamma \).

\[\square\]

Q: Do you really need local formulas in addition to the components of \( \Gamma \cup \{\varphi\} \)? Maybe there is a derivation of \( \varphi \) from \( \Gamma \) that uses only the components whenever \( \varphi \) is derivable from \( \Gamma \).

A: Here is a counter-example to your conjecture. Let \( x, y, z \) be distinct infon variables, and let \( \Gamma \) be

- \( p \text{ implied } x \).
• *p* said \((x \rightarrow y)\),

• *p* said \((y \rightarrow z)\).

Formula \(\varphi = (p \text{ implied } z)\) is derivable from \(\Gamma\) but any derivation involves local formula \((p \text{ implied } y)\) that is not a component of \(\Gamma \cup \{\varphi\}\). Furthermore, if \(y\) is not a variable and \(y = (q \text{ said } y')\) then any derivation of \(\varphi\) from \(\Gamma\) involves a quotation prefix \((p \text{ implied } q \text{ said})\) that is not a prefix of any component.

6 Primal intuitionistic logic

To logicians, primal intuitionistic logic may be of interest in its own right. In this connection, in §6.1, we specialize the relevant results above to the case of primal intuitionistic logic. In §6.2 we construct a linear time algorithm for the multiple derivation problem (defined in §1) for primal intuitionistic logic. The algorithm will be generalized in the next section. For brevity primal intuitionistic logic will be called PC which is an allusion to “Primal Constructive logic” as intuitionistic logic is also known as constructive.

6.1 Syntax and semantics

PC formulas are built from variables and constant \(\top\) by means of conjunction and implication. A PC sequent calculus is obtained from the sequent calculus for primal infon logic by removing inference rules (Said) and (Implied):

**Axioms**

\[
\begin{align*}
(\top) & \quad \Gamma \vdash \top \\
(x2x) & \quad x \vdash x
\end{align*}
\]

**Inference rules**

\[
\begin{align*}
\text{(Premise Inflation)} & \quad \frac{\Gamma \vdash y}{\Gamma,x \vdash y} \\
\text{(\&E)} & \quad \begin{array}{c}
\frac{\Gamma \vdash x \land y}{\Gamma \vdash x} \\
\frac{\Gamma \vdash x \land y}{\Gamma \vdash y}
\end{array}
\end{align*}
\]

168
Kripke structures for PC are triples \((W, \leq, C)\) subject to conditions K1–K5 in §2.1 and condition K6W in §2.2. Theorem 3.2 remains true. More exactly, for every PC sequent \(s\) the clauses 1–4 of Theorem 3.2 are equivalent.

A Hilbert type calculus \(\mathcal{H}_{PC}\) for PC is a simplification of calculus \(\mathcal{H}\) of the previous section:

**Axiom:** \([\top]\)

**Inference rules**

\[\begin{align*}
(\land e) & \quad \frac{x \land y}{x} \quad \frac{x \land y}{y} \\
(\land i) & \quad \frac{x}{x \land y} \\
(\rightarrow e) & \quad \frac{x}{x \rightarrow y} \\
(\rightarrow i) & \quad \frac{y}{x \rightarrow y}
\end{align*}\]

In the PC case, the components of a formula \(z\) are exactly the subformulas of \(z\), and the local formulas of \(z\) are exactly the subformulas of \(z\). Theorem 3.2 simplifies as follows.

**Theorem 6.1.** For any PC sequent \(s = [\Gamma \vdash \varphi]\), the following are equivalent: (i) \(s\) is derivable in the PC sequent calculus, (ii) there is a derivation of \(s\) in the PC sequent calculus where every formula is a subformula of \(s\), (iii) \(s\) is valid, (iv) \(\varphi\)
is derivable from $\Gamma$ in $\mathcal{H}_{PC}$, and (v) there is a derivation of $\varphi$ from $\Gamma$ in $\mathcal{H}_{PC}$ where every formula is a subformulas of $s$.

### 6.2 Linear time theorem for primal constructive logic

**Theorem 6.2.** There is a linear time algorithm for the multiple derivability problem for primal constructive logic. Given hypotheses $\Gamma$ and queries $Q$, the algorithm determines which of the queries in $Q$ follow from the hypotheses $\Gamma$.

**Proof** Formulas local to (that is subformulas of) $\Gamma \cup Q$ will be simply called local. By Theorem 6.1, we may restrict attention to derivations where all formulas are local. The idea is to compute all local consequences of $\Gamma$. This is obviously sufficient.

Without loss of generality we may assume that $\top$ does not occur in $\Gamma \cup Q$. It follows that $\top$ does not occur in any local formula. Let $n$ be the length of the input sequence $\Gamma, Q$. The key $K(y)$ of a local formula $y$ is the position of the first symbol of the first occurrence of $y$ in the input sequence.

**Parse tree** Run a parser on the input string producing a parse tree. The subtrees of the hypotheses and the queries hang directly under the root. Each node of the parse tree has a label that is or represents (according to the lexical analyzer) a variable or connective. The label length is $O(\log(n))$ due to the lexical analysis phase of parsing. Extra tags mark hypotheses and queries; such a tag is not a part of the official label.

By induction define the locutions $L(u)$ of nodes $u$. If $u$ is a node with children $u_1, \ldots, u_k$, then

$$L(u) = \text{Label}(u)(L(u_1), \ldots, L(u_k)).$$

The locutions are being introduced for use in our analysis of the algorithm; the algorithm will not have the time to produce all the locutions.

Each node $u$ is associated with a particular occurrence of $L(u)$ in the input string. The initial position of $L(u)$ in the input string is the key $K(u)$ of $u$. Nodes $u$ and $v$ are homonyms if $L(u) = L(v)$. A node with the least key in its homonymy class is a homonymy original. A homonymy original $u$ represents the locution $L(u)$; its key is the key of the locution. We presume that the nodes $u$ come with homonymy pointers $H(u)$ initially set to nil.

**Homonymy originals** Run the Cai-Paige algorithm [4] on the parse tree. The algorithm partitions the (keys of) nodes $u$ into buckets $B_\ell$, according to the height $\ell$ of $u$, that is the height (or depth) of the subtree rooted at $u$. Furthermore, every bucket is ordered according to the lexicographic order of locutions $L(u)$. Note
that two homonyms have the same height and thus belong to the same bucket. Homonyms are ordered according their keys. The Cai-Paige algorithm sets every homonym pointer \( H(u) \) to the homonymy original of \( u \). This wonderful algorithm runs in linear time.

### Preprocessing

Create a table \( T \) of records indexed by the homonymy originals \( u \) such that \( L(u) \) is a formula. A record \( T(u) \) has five fields. One of them is the status field \( S(u) \) with values in the set \( \{1, 2, 3\} \). The status field is dynamic; its value may change as our algorithm runs. All other fields are static; once created their values remain immutable.

The status field \( S(u) \) determines the current status of the formula \( L(u) \). Formulas of status 1, 2, 3 will be called raw, pending, processed respectively. A raw formula has not been derived. A pending formula has been derived but remains a subject of some processing. A processed formula has been derived and processed. Initially every \( S(u) = 1 \). Traverse the parse tree setting the status \( S(u) \) to 2 if \( L(u) \) is tagged as a hypothesis.

The static fields of the record \( T(u) \) are \((\wedge, \text{left}), (\wedge, \text{right}), (\rightarrow, \text{left}) \) and \((\rightarrow, \text{right})\). Each entry in these fields is a sequence of (the keys of) nodes. To compute those sequences, traverse the parse tree in the depth-first manner. If the current node \( v \) is the left child of a node \( v' \) with label “\( \wedge \)” then append \( H(v') \) to the \((\wedge, \text{left})\) sequence of \( H(v) \) and move on. The cases where \( v \) is the right child of \( v' \) or the label of \( v' \) is \( \rightarrow \) or both are treated similarly.

### Processing

Walk through the table \( T \) and process every pending formula \( L(u) \) in turn. When the processing of \( L(u) \) is finished do the following.

- Set \( S(u) = 3 \) indicating that \( L(u) \) is processed.
- If \( u \) is tagged as a query then output \( K(u) \).

The processing consists of firing, one after another, the inference rules applicable to \( L(u) \).

Rule (\( \wedge e \)) requires that \( L(u) \) has the form \( x \wedge y \). If any of the formulas \( x, y \) is raw, make it pending. More exactly, let \( u_1, u_2 \) be the left and right children of \( u \). If \( S(H(u_i)) = 1 \), set \( S(H(u_i)) = 2 \).

Rule (\( \wedge i \)) may be applied in two ways depending on whether we view \( L(u) \) as the left or right premise of the rule. The two cases — call them the left and right cases — are similar; we describe here only the left case where \( L(u) \) is the left conjunct \( x \) of a formula \( x \wedge y \). For any such \( y \) that has been derived but \( x \wedge y \) is still raw, make \( x \wedge y \) pending. More exactly, for every \( v \) in the \((\wedge, \text{left})\) sequence of \( T(u) \)
do the following. Note that \( L(v) = L(u) \land L(w) \) where \( w \) is the right child of \( v \). If \( S(H(w)) > 1 \) and \( S(v) = 1 \), set \( S(v) = 2 \).

Rule \((\rightarrow e)\) also may be applied in two ways depending on whether we view \( L(u) \) as the left or right premise of the rule. This time around the two cases — call them the left and right cases — are quite different. The left case is similar to the left case in the application of rule \((\land i)\) above. For every \( v \) in the \((\rightarrow, \text{left})\) sequence of \( T(u) \) do the following. Note that \( L(v) = L(u) \lor L(w) \) where \( w \) is the right child of \( v \). If \( S(v) > 1 \) and \( S(H(w)) = 1 \), set \( S(H(w)) = 2 \). The right case requires that \( L(u) \) has the form \( x \rightarrow y \). If \( x \) has been derived and \( y \) is raw, make \( y \) pending. More exactly, let \( u_1, u_2 \) be the left and right children of \( u \). If \( S(H(u_1)) > 1 \) and \( S(H(u_2)) = 1 \), set \( S(H(u_2)) = 2 \).

Rule \((\rightarrow i)\) has only one premise \( y = L(u) \). Any raw formula of the form \( x \rightarrow y \) becomes pending. More exactly, for every \( v \) in the \((\rightarrow, \text{right})\) sequence of \( T(u) \) do the following. Note that \( L(v) = L(w) \rightarrow L(u) \) where \( w \) is the homonymy original of the left child of \( v \). If \( S(v) = 1 \), set \( S(v) = 2 \).

This completes the algorithm.

**Proof of correctness** Note that every pending formula becomes processed. Obviously only provable formulas become pending. To prove the correctness of the algorithm, it suffices to prove that every provable formula \( L(v) \) becomes pending. We do that by induction on the proof length of \( L(v) \). If \( L(v) \) is a hypothesis, it becomes pending when the table is formed. Otherwise several cases arise depending on the rule used at last step of the given proof of \( L(v) \). All cases are pretty obvious. We consider just one of those cases.

Case \((\land i)\) where \( L(v) = x \land y \) and \( x, y \) have been derived earlier. By symmetry we may assume without loss of generality that \( x \) became pending first. Formula \( y \) is \( L(u) \) for some \( u \). But then \( v \) occurs in the \((\land, \text{right})\) sequence of \( T(u) \). Accordingly \( L(v) \) becomes pending as a result of applying the right case of rule \((\land i)\) in the processing of \( L(u) \).

**Time complexity** It remains to check that the algorithm is indeed linear time. Obviously the parse-tree and table stages are linear time. The only question is whether the processing stage is linear time. Instead we claim something else. Note that the processing of a pending formula \( L(u) \) is done in a fixed finite number of phases. At each phase, we make some number \( k(u) \) of attempts to apply a particular inference rule viewing \( L(u) \) as one of the premises. Each attempt takes bounded time. The number of attempts is not bounded. It suffices to prove, however, that \( \sum_u k(u) = O(n) \).
The proof is similar for all phases. Here we consider only the phase when we attempt to apply rule (\(\wedge e\)) with \(L(u)\) as the left premise \(x\). In this phase the number \(k(u)\) is the length of the \((\wedge, \text{left})\) sequence of \(u\). But the \((\wedge, \text{left})\) sequences for distinct records are disjoint. And so \(\sum u k(u) \leq n\). \(\square\)

7 Linear time theorem for primal infon logic

We return to primal infon logic. We will be working with fragments of the Hilbert-type calculus \(\mathcal{H}\) for primal infon logic. To recall \(\mathcal{H}\), see §5.

In §4 we gave the obvious definition of the quotation depth of formulas. In the case of primal infon logic, another definition of quotation depth is more pertinent.

**Definition 7.1** (Primal quotation depth). The primal quotation depth of formulas is defined by induction:

- \(\delta(x) = 0\) if \(x\) is a variable.
- \(\delta(p \ \text{told} \ x) = 1 + \delta(x)\).
- \(\delta(x \ \wedge \ y) = \max\{\delta(x), \delta(y)\}\).
- \(\delta(x \ \rightarrow \ y) = \delta(y)\)

Further \(\delta(\Gamma) = \max_{\alpha \in \Gamma} \delta(\alpha)\) for any set \(\Gamma\) of formulas. \(\square\)

Recall Definition 1.1 of the multiple derivability problem \(\text{MD}(L)\) for a logic \(L\).

**Theorem 7.2.** For every natural number \(d\), there is a linear time algorithm for the multiple derivability problem for primal infon logic restricted to formulas \(z\) with \(\delta(z) \leq d\).

**Q:** The definition of primal quotation depth is strange. The clause \(\delta(x \ \rightarrow \ y) = \delta(y)\) ignores \(\delta(x)\). If \(\text{foo}\) is a quotation-free formula then \(\delta(p \ \text{said} \ q \ \text{said} \ \text{foo}) \rightarrow \text{foo} = 0\). The formula involves quotation but its primal quotation depth is zero.

**A:** Well, the strange definition makes the theorem stronger.

**Q:** Does the additional strength matter?

**A:** It does. For example it allows us to interpret authorization logic SecPAL [3] in the stratum of depth \(\leq 2\) of primal infon logics [10].
### 7.1 Reduction to Main Lemma

**Definition 7.3 (Regular).** A formula \( x \) is regular if it has no occurrences of \( \top \). A derivation is regular if all its formulas are regular. \( \mathcal{R} \) is the fragment of \( \mathcal{H} \) obtained by removing \( \top \) from the language and removing the axioms from the calculus.

Two formulas are **equivalent** if each of them entails the other in primal infor logic.

**Lemma 7.4.**

1. For any formula \( z \) there is an equivalent formula \( z' \) that is either an axiom or regular.
2. There is a linear time algorithm that, given a formula \( z \), computes the equivalent formula \( z' \).
3. Any local derivation of a regular formula from regular hypotheses is regular.
4. There is a linear-time reduction of \( \text{MD}(\mathcal{H}) \) to \( \text{MD}(\mathcal{R}) \).

**Proof**

1. Easy induction on \( z \). We consider only the case \( z = x \to a \) in the induction step where \( a \) is an axiom. In that case \( z' \) may be \( a \). Indeed \( z \vdash a \) because \( a \) is an axiom, and \( a \vdash z \) by (Pref→I).
2. Execute the **regularization algorithm** implicit in the proof of 1. One appropriate data structure is parse tree. Even if \( z \) is given as a string, in linear time you can construct the parse tree of \( z \) and then execute the regularization algorithm.
3. By the definition of local derivation.
4. Apply the regularization algorithm to the hypotheses and queries. If a modified hypothesis is an axiom, remove it. If a modified query is an axiom, mark it derivable and remove it. \( \square \)

**Lemma 7.5.** Theorem 5.4 remains true if sequent \( s \) is assumed to be regular and if \( \mathcal{H} \) is replaced with \( \mathcal{R} \).

**Proof**

The proof of Theorem 5.4 needs only two minor modifications, in fact simplifications. In the proof that (4) implies (1), ignore the case where \( \phi \) is an axiom of \( \mathcal{H} \). Similarly, in the proof that (2) implies (5), in the proof that \( D' \) is a derivation, ignore the case where the tail formula \( z \) is an axiom of \( \mathcal{H} \). \( \square \)

**Lemma 7.6.**

1. For any inference rule in \( \mathcal{R} \), the primal quotation depth of the conclusion is bounded by the primal quotation depth of the premise(s).
2. If \( \Gamma \) entails \( \phi \) in \( \mathcal{R} \) then \( \delta(\phi) \leq \delta(\Gamma) \).
3. The restriction \( \mathcal{R}_d \) of \( \mathcal{R} \) to formulas \( z \) with \( \delta(z) \leq d \) is a calculus in its own right; all inference rules of \( \mathcal{R} \) produce formulas in \( \mathcal{R}_d \) when applied to formulas in \( \mathcal{R}_d \).
Proof Claim 1 is obvious but note that the unusual clause $\delta(x \rightarrow y) = \delta(y)$ in the definition of primal quotation depth is used in the case of rule (Pref $\rightarrow$I). Claims 2 and 3 are obvious as well.

**Main Lemma 7.7.** For every natural number $d$, there is a linear time algorithm for the multiple derivation problem $MD(R_d)$ for $R_d$.

Clearly Theorem 7.2 follows from Main Lemma. We prove Main Lemma in the next subsection.

### 7.2 Proof of Main Lemma

Given a positive integer $d$, we construct a linear time algorithm for $MD(L_d)$ by modifying the linear-time algorithm of §5. Recall that $\text{told}$ with or without a subscript ranges over \{said, implied\}, and $\text{pref}$ with or without a subscript ranges over quotation prefixes $q_1 \text{told}_1 \ldots q_i \text{told}_i$ where $k$ is the length of $\text{pref}$. We say that a quotation prefix is relevant if its length is $\leq d$.

**Parsing** Parse the input as in §6.2 except that now we have additional labels $p \text{told}$. There is a problem, however. In §6.2, every local formula was the location $L(u)$ of some node $u$ of the parse tree. Now it is not necessarily the case. To remedy the situation, we graft extra nodes into the parse tree. This is not the optimal remedy but it simplifies the exposition.

By induction on nodes $u$, define $\text{pref}(u)$. If $u$ is the root, then $\text{pref}(u)$ is empty. Suppose that $v$ is the parent of $u$. If the label of $v$ is of the form $p \text{told}$ then $\text{pref}(u) = \text{pref}(v)$ appended with the label of $v$; otherwise $\text{pref}(u) = \text{pref}(v)$. Let $C(u)$ be the formula $\text{pref}(u) L(u)$. It is easy to check that every $C(u)$ is a component and that every component is $C(u)$ for some $u$. Call node $u$ relevant if (i) $u$ is not the root of the parse tree so that $L(u)$ is a formula and (ii) $\text{pref}(u)$ is relevant.

The *fan* $F(\text{pref})$ of a prefix $\text{pref}$ is a tree of prefixes. It contains all prefixes $\text{pref}' \leq \text{pref}$. Further, it contains a prefix $q_1 \text{told}_1 \ldots q_i \text{told}_i$ of length $i$ if it contains a prefix $q_1 \text{told}_1 \ldots q_i \text{told}_i q_{i+1} \text{told}_{i+1}$ of length $i + 1$ which is a parent of $q_1 \text{told}_1 \ldots q_i \text{told}_i$. Every node in $F(\text{pref})$ has at most two children and at most one parent. If $d = 2$, then $F(\text{pref})$ contains at most 7 nodes. Now turn $F(\text{pref})$ upside down, and let $F'(\text{pref})$ be the result. Normally our trees go downward so that the root is at the top. $F'(\text{pref})$ grows upward.

Traverse the parse tree in the depth-first manner and graft a fresh copy $F(u)$ of $F'(\text{pref}(u))$ at every relevant node $u$. There is a one-to-one correspondence $\xi : F(u) \longrightarrow F'(\text{pref}(u))$. If $v \in F(u)$ and $\xi(v)$ is a prefix $q_1 \text{told}_1 \ldots q_i \text{told}_i$ of length $i > 0$ then the label of $v$ is $q_i \text{told}_i$, and the key of $v$ is the pair...
(Key(u), told₁ ... toldₖ)

The keys are ordered lexicographically. We do not distinguish between Key(u) and the pair (Key(u), s) where string s is empty. Every node u of the resulting parse structure has at most three parents, and the nodes ≤ u form a tree.

Remark 7.8. For every relevant original node u, the formula pref(u) L(u) is a component of Γ ∪ Q. If pref ≤ pref(u) then pref L(u) is local. Every local formula is obtained this way. Thus the parse structure has a node for every local formula (and for some non-local formulas).

Homonymy originals As in §6.2, run the Cai-Paige algorithm on the parse structure and compute homonymy pointers H(u).

Preprocessing Let T₁ be the table T constructed in §6.2. T₁ is a one-dimensional table of records T₁(u) indexed by nodes u such that u is a homonymy original. Now, in addition to T₁, we construct a sparse two-dimensional table T₂. The rows of T₂ are indexed by original nodes u, and the columns are indexed by relevant prefixes π. If there is a node v with L(v) = π L(u), then T₂(u, π) = {H(v)}; otherwise T₂(u, π) = ∅. A traversal around the parse structure suffices to fill in table T₂.

Remark 7.9. We graft nodes and then put only some of them into table T₂. This is not the most efficient way to do things. One can forgo grafting, construct table T₂ directly, and refer to the table instead of to grafted nodes in the following processing. We thought that grafting would simplify the exposition.

Remark 7.10. It is more efficient to combine preprocessing with parsing.

Processing As in §6.2, we walk through the table T and process every pending formula L(u) in turn. The processing consists of firing, one after another, the inference rules applicable to L(u). The case of rule

\[
\frac{\text{pref}_2 \ x}{\text{pref}_1 \ x}
\]

is new. Suppose that L(u) = pref₂ x and let pref₁ ≤ pref₂. The descendant u₀ of u with locution x has an ancestor v with locution pref₁ x. If H(v) is raw, make it pending.

As far as the remaining rules are concerned, the situation is similar to that in §6.2. For example, consider the application of the rule

\[
\frac{\text{pref}(x \land y)}{\text{pref} \ x}
\]
The Bulletin of the EATCS

to a formula $L(u) = \text{pref}(x \land y)$. Find the descendant $u_0$ of $u$ with $L(u_0) = x \land y$. The left child of $u_0$ has an ancestor $v$ with locution $\text{pref } x$. If $H(v)$ is raw, make it pending.

For a more interesting example, consider the application of rule

$$
\frac{\text{pref } x \quad \text{pref } y}{\text{pref } (x \land y)}
$$

to a formula $L(u) = \text{pref } x$. Find the descendant $u_0$ of $u$ with $L(u) = x$. For each $v$ in the $(\land, \text{left})$ field of record $T_i(H(u_0))$ do the following.

Check the entry $T_2(v, \text{pref})$. If it is empty, do nothing. Otherwise let $w$ be the node in the entry. The locution of $v$ is $x \land y$, and the locution of $w$ is $\text{pref } (x \land y)$. Let $w_0$ be the descendant of $w$ with locution $x \land y$. The right child of $w_0$ has an ancestor $w'$ with locution $L(w') = \text{pref } y$. If the status of $H(w')$ is $> 1$, so that $\text{pref } y$ has been proved, but the status of $w$ is $1$, so that $\text{pref } (x \land y)$ has not been proved, then set the status of $w$ to $2$; otherwise do nothing.

Correctness and time complexity The proof of the correctness of the algorithm and the analysis of time complexity of §6.2 survive with minor changes.

References


REPORTS FROM CONFERENCES
REPORT ON BCTCS 2009

Artur Czumaj, Sara Kalvala, Steve Matthews

The British Colloquium for Theoretical Computer Science (BCTCS) is an annual meeting for researchers to present their work in a wide range of areas including algorithms, logic, and the design of programming languages (www.bctcs.ac.uk) This event has always been particularly special for the emphasis it gives to supporting PhD students to present their own work and meet internationally respected researchers. For many years the UK Engineering and Physical Research Council (EPSRC) has supported students attending, and the London Mathematical Society (LMS) support a special invited speaker.

BCTCS 2009 took place from April 6-9th, hosted by the Computer Science Department of the University of Warwick and the Centre for Discrete Mathematics and its Applications (DIMAP). 97 participants produced the high quality programme of research in theoretical computer science for which the BCTCS is well known, enriched by talks from the invited distinguished speakers:

- Noga Alon, Tel Aviv University (LMS Keynote Speaker in Discrete Maths),
- Paul Goldberg, University of Liverpool,
- Andy Gordon, Microsoft Research,
- Jane Hillston, University of Edinburgh,
- Alistair Sinclair, University of California at Berkeley,
- Bill Wadge, University of Victoria

For more information on the event itself please see the web site www.dcs.warwick.ac.uk/events/bctcs. Included below are abstracts for each of the invited and contributed talks.

BCTCS 2010 will take place at the University of Edinburgh, from April 6-9th, hosted by the Computer Science Department of the University of Edinburgh, with lead organiser Dr. Julian Bradfield.
Abstracts: Invited Talks

COMBINATORIAL REASONING IN INFORMATION THEORY
Noga Alon, Tel Aviv University

Combinatorial arguments have played a crucial role in the investigation of several surprising phenomena in Information Theory. After a brief discussion of some of these results I will describe a recent example, based on joint papers with Lubetzky and Stav, and with Hasidim and Weinstein, in which properties of graph powers, colorings of Cayley graphs, and the chromatic numbers of Kneser graphs are applied in the study of a broadcasting problem with side information.

PRINCIPLES AND APPLICATIONS OF REFINEMENT TYPES
Andy Gordon, Microsoft Research

A refinement type is a type qualified by a logical constraint; an example is the type of even numbers, that is, the type of integers qualified by the is-an-even-number constraint. Although this idea has been known in the research community for some time, it has been assumed impractical, because of the difficulties of constraint solving. But recent advances in automated reasoning have overturned this conventional wisdom, and transformed the idea into a practical design principle. I will present a primer on the design, implementation, and application of refinement types. I will explain:

- How a range of diverse features may be unified as instances of the general idea of refinement types.
- How a static checker for the Oslo modeling language M allows us to check for security errors in server configurations; intended constraints on configurations are expressed with refinement types, so that configuration validation reduces to type checking.
- How we statically check integrity and secrecy properties of security critical code, such as an implementation of the CardSpace security protocol, using a system of refinement types for the F# programming language.

The lectures in this series are based on recent research with my esteemed colleagues Karthik Bhargavan, Gavin Bierman, and Cédric Fournet of MSR Cambridge, and David Langworthy of the Microsoft Connected Systems Division; much of our work relies on the excellent Z3 automated theorem prover developed by Nikolaj Bjorner and Leonardo de Moura of MSR Redmond.

RECENT PROGRESS IN COMPUTING APPROXIMATE NASH EQUILIBRIA
Paul Goldberg, University of Liverpool

In Game Theory, the Nash Equilibrium is the most prominent and well-known solution concept, mainly due to the famous result of John Nash showing that every game is guaranteed to have such a solution. Recent results suggest however that in the worst case, Nash equilibria may be computationally hard to find. A subsequent line of research has considered the problem of searching for a weaker version
of this solution, an "approximate" Nash equilibrium. In this talk I will give an overview of some of these results, and discuss future directions.

PHASE TRANSITIONS AND MIXING TIMES
Alistair Sinclair, University of California at Berkeley

Recent work on random satisfiability and other problems has raised the possibility of deep connections between phase transitions (as studied in physics) and computational complexity. Markov chain Monte Carlo algorithms provide one of the most compelling examples to date of this connection. Roughly speaking, the physical notion of a phase transition frequently has a computational manifestation in the form of a sudden jump in the mixing time.

In this talk I will illustrate various aspects of the above phenomenon, with special emphasis on the classical Ising model. No knowledge of statistical physics will be assumed.

STOCHASTIC PROCESS ALGEBRA: BRINGING PERFORMANCE TO LIFE
Jane Hillston, University of Edinburgh

Stochastic process algebras emerged in the early 1990s as a novel formal description technique for performance modelling based on continuous time Markov chains (CTMCs). Enhancing classical process algebras with information about the expected duration of actions, stochastic process algebras have a clear CTMC semantics yet offer compositionality and formal manipulation techniques which are not available when performance models are constructed directly at the CTMC level. Over the ensuing decade stochastic process algebras such as PEPA enjoyed significant success in modelling and analysing a range of computer systems including software and communication systems. In the last eight years there has been considerable interest in applying these formalisms to biological applications, particularly for modelling the dynamics of intracellular processes. In this tutorial I will give an overview of stochastic process algebras and explain the attractions and challenges of using them in these diverse areas of application.

INFINITESIMAL LOGIC
Bill Wadge, University of Victoria

Infinitesimal logic is a multivalued logic in which there are discrete levels of truth and falsehood. At the top we have the Gold standard of absolute truth, but also, directly below it, Silver truth as well. Silver truth is much less true than Gold truth - in fact, infinitely less true. But Silver truth is still true, and infinitely more true than Bronze truth, just below it; and so on. At the bottom we have Gold (complete) falsity, then the much less false Silver falsity, then Bronze falsity, and so on, with a neutral value right in the middle.

This logic was first developed to give a semantics for negation as failure—for example, the negation-as-failure of Silver truth is Bronze falsity. More generally, infinitesimal logic allows us to avoid many of the paradoxes of self-reference.
On the practical side, this logic gives a semantics to constraints (such as database queries) with preferences. The most preferable answers to a query are those that satisfy the constraints at the Gold level, followed by those that satisfy it at the Silver level, and so on. Queries can be constructed with the usual Boolean connectives but also with preference sensitive alternatives; two of which can be thought of as "and, if possible" and "or, failing that".

**Abstracts: Contributed Talks**

**PTAS FOR THE k-TOUR COVER PROBLEM ON THE EUCLIDEAN PLANE FOR MODERATELY LARGE VALUES OF k**

Anna Adamaszek in joint work with Artur Czumaj and Andrzej Lingas

We are given a set $P$ of $n$ points in the Euclidean plane and a distinguished point $O$ called the origin. A $k$-tour cover is a set of tours covering all points from $P$, such that each tour starts and ends in the origin and covers at most $k$ points from $P$. The objective of the $k$-tour cover problem is to find a $k$-tour cover which minimizes the total length of the tours.

This problem is known to be $NP$-hard. It is also known to admit constant factor approximation algorithms for all values of $k$ and even a polynomial-time approximation scheme (PTAS) for small values of $k$, i.e., $k = O(\log n / \log \log n)$.

I will present a new PTAS for all values of $k \leq 2^{\log^a n}$, where $\delta = \delta(\epsilon)$. The PTAS is based on a novel reduction of the $k$-tour cover problem with a set of $n$ points to a small set of instances of the problem, each with $O((k/\epsilon)^{O(1)})$ points.

**FORMAL SIMULATION OF SUPERVISED COMPONENTRY MODELS AND THEIR EXECUTION**

Djihed Afifi

We consider the modelling of systems that can adapt their behaviour at runtime in response to external and internal stimuli. A logical framework for such evolvable systems is introduced in [1]. In this framework, an evolvable system is modelled as a tree of components. Each component is modelled as a first order logic theory with constraints, actions and a state modelled as ground atoms. A component may be standalone or may be formed by a special pairing of supervisor and supervisee components. In this pair, the supervisor has access to the supervisee’s theory and so it can trigger an evolution by altering its sub-components, its constraints or its state.

A simulator tool for this framework is under development. The simulator accepts a logical specification and executes a sequence of instructions. During execution, the component’s theories must remain internally consistent. Actions
pre-conditions must be met before firing actions. New constraints or components
must not render the component’s state conflicting.

To ensure this, the component’s theories are rewritten together with theorems
that test the logical obligations. These theorems are then proved on the fly using
automated theorem provers. The simulator currently invokes provers that support
the TPTP [2] format, such as iProver, Vampire and Paradox, with plans to support
other provers, such as PVS, in the future.

In this talk I will present some of the theoretical and practical issues aris-
ing from the simulation. Different theorem provers accept first order logics with
different extensions while the logic used in this framework is a typed FOL. This
necessitates theory translation. Other issues include ensuring state persistence and
generating minimum models.

Available from: http://www.cs.man.ac.uk/~david/evolution/evolution.html


SPANNING CONNECTIVITY GAMES
Haris Aziz in joint work with Oded Lachish, Mike Paterson and Rahul Savani

The Banzhaf index, Shapley-Shubik and other voting power indices measure
the importance of a player in a coalitional game. We consider a simple coalitional
game called the spanning connectivity game (SCG) based on an undirected multi-
graph, where edges are players. We examine the computational complexity of
computing the voting power indices of edges in the SCG. It is shown that com-
puting Banzhaf values is #P-complete and computing Shapley-Shubik indices or
values is NP-hard for SCGs. Interestingly, Holler indices and Deegan-Packel in-
dices can be computed in polynomial time. Among other results, it is proved that
Banzhaf indices can be computed in polynomial time for graphs with bounded
tree-width. It is also shown that for any reasonable representation of a simple
game, a polynomial time algorithm to compute the Shapley-Shubik indices im-
plies a polynomial time algorithm to compute the Banzhaf indices. This answers
(positively) an open question of whether computing Shapley-Shubik indices for a
simple game represented by the set of minimal winning coalitions is NP-hard.

COMPUTATION OF THE INDEX OF A COMPONENT OF NASH EQUILIBRIA FOR
BIMATRIX GAMES
Anne Balthasar

In game theory, the index of a component of Nash equilibria is an important
topological notion which can be used to characterize certain properties of such
a component. For example, an equilibrium component is hyperstable, i.e. does
not vanish under certain manipulations of the game, if and only if its index is
non-zero.

For non-degenerate bimatrix games, the calculation of the index of an equilibrium is straightforward via an explicit formula, which boils down to computing determinants of square matrices. However, in degenerate cases, this formula fails to make sense, and index computation then amounts to calculating the (relatively complex) topological degree of a function. To resolve this difficulty we present an algorithm for the computation of the index in degenerate bimatrix games.

MULTIPROCESSOR SPEED SCALING FOR JOBS WITH ARBITRARY SIZES AND DEADLINES

Paul C. Bell in joint work with Prudence Wong

Energy consumption has become an important concern in the design of modern processors, not only for battery-operated mobile devices with single processors but also for server farms or laptops with multi-core processors. A popular technology to reduce energy usage is dynamic speed scaling where the processor can vary its speed dynamically. The power consumption is modelled by $s^\alpha$ when the processor runs at speed $s$, where $\alpha$ is typically 3 in reality. Running a job slower saves energy, yet it takes longer to finish the job. The study of speed scaling was initiated by Yao et al., see [1]. They studied deadline scheduling on a single processor in which jobs with arbitrary sizes and deadlines arrive online and the aim is to finish all jobs by their deadlines using the minimum amount of energy.

Albers et al. have extended the study to the multiprocessor setting in the special cases of unit-size jobs or jobs with agreeable deadlines (jobs arriving earlier have earlier deadlines), and presented constant competitive algorithms for both cases [2]. In the multiprocessor setting, in addition to determining processor speed, a job dispatching algorithm is required to assign jobs to processors. We will present results concerning generalized problems where jobs have arbitrary sizes and arbitrary deadlines. We propose a non-migratory job dispatching algorithm, called DCRR, and show that DCRR is $O(\log P)$-competitive, where $P$ is the ratio between the maximum and minimum job size.


RECENT THEORETICAL AND PRACTICAL DEVELOPMENTS WITH BIGRAPHS

Clive Blackwell

Bigraphs [1] are a process calculus based on category theory. A bigraph is composed of a link and place graph with the same nodes, but different edges. The link graph represents logical communication (as in the pi calculus) and the place
graph models physical locality (as in the ambient calculus).

Bigraphs can be used to model security with additional reaction rules to model cryptographic and other security mechanisms, which is analogous to the extension of the pi-calculus to the spi-calculus. A system and its users are modelled as bigraphs, where bigraph reaction or rewriting rules private to the defender model the possible security mechanisms, and inverse rules model the access rights of users. The defender’s objectives can be defined by bigraph invariants such as preventing unauthorised access to critical nodes, preserving nodes and channels that represent security boundaries, and limiting access to protective reaction rules.

The advantages of bigraphs, compared to most other process calculi, are the explicit representation of location and system structure that are fundamental to modelling many aspects of systems realistically. Protection mechanisms cannot remove weaknesses; only transform them. Bigraphs model protection mechanisms that control access to protected resources using private communication channels from inaccessible locations. This allows a more general and realistic adversary model than many process calculi that abstract away location. An adversary can gain new powers by acquiring access to the locations of reaction rules situated within the bigraph.

We also illustrate some recent theoretical developments of bigraphs. Several graph algorithms such as transitive closure remain polynomial time when we consider bigraphs as special types of DAGs. We need to make some reasonable constraints on the reaction rules to ensure their finite application, and store system state using special attribute nodes to avoid the unnecessary application of reaction rules.


THREE WAYS TO TEST IRREDUCIBILITY
Richard P. Brent in joint work with Paul Zimmermann

We consider several algorithms for testing irreducibility of sparse polynomials over finite fields of small characteristic. The algorithms that are fastest in theory turn out not to be best in practice. A hybrid algorithm that combines the classical approach with modular composition is suggested. As an application, we describe a search for irreducible trinomials (over GF(2)) whose degree is a large Mersenne exponent.

SAFETY DOES NOT CONSTRAIN EXPRESSIVITY FOR WORD-LANGUAGES
Christopher Broadbent

Higher-order recursion schemes are systems of rewrite rules that can generate infinite trees and word-languages. Higher-order pushdown automata (HOPDA), which were first introduced by Maslov [5], are devices equipped with a stack that itself contains stacks of stacks... of stacks. If recursion schemes are restricted by a
constraint known as safety, then they have the same expressive power as HOPDA [2, 4]. Arbitrary recursion schemes require collapsible pushdown automata [3].

For word-languages we have shown that for any order-\(n\) unsafe recursion scheme there is an equivalent order-\((n + 1)\) safe recursion scheme. The argument goes via automata and extends the idea of Aehlig \textit{et al.} used at order-2 [1]. We extend recursion schemes with ‘exception handling’, which bears some connections with safety. Our argument shows that order-\(n\) recursion schemes with exceptions are equi-expressive with order-\((n + 1)\) recursion schemes without.

This talk focuses on introducing higher-order (collapsible) pushdown automata together with recursion schemes (with exceptions). We will briefly sketch the idea of Aehlig \textit{et al.’s} proof at order-2 and the difficulties that must be surmounted to obtain our generalisation.


LONGEST PREVIOUS REVERSE FACTOR
Supaporn Chairungsee

Data compression is useful in data communication over a low-bandwidth channel and for storing documents efficiently. Lempel-Ziv factorisations yield well-known powerful technique for data compression. The Longest Previous Factor (LPF) table of a string provides an efficient way to deal with the LZ77 factorisation. It is even more efficient if the Longest Previous Reverse Factor (LPRF) table is used as it captures more information on the string. We describe new algorithms for computing the LPRF of a string by using three data structures that are suffix trie, suffix tree and suffix automaton. The last two algorithms run in linear time on a fixed size alphabet.

ALGORITHMS FOR RANDOM \(k\)-SAT
Amin Coja-Oghlan

The \(k\)-SAT problem is well known to be NP-hard for any \(k \geq 3\). Among the (empirically) most challenging benchmark instances for this problem are randomly generated \(k\)-SAT formulas. In this talk I will survey various algorithms for coping with these instances. These are either simple combinatorial heuristics, algorithms based on backtracking (“DPLL”), randomized algorithms, or message passing algorithms. The quality of these algorithms can be measured in terms of the constraint density (number of clauses divided by number of variables) up to
The Bulletin of the EATCS

which the algorithm typically succeeds. In addition, I’ll present a new algorithm
that succeeds up to a constraint density that has been suggested as a natural barrier
for efficient algorithms to find satisfying assignments.

GENERATING AND COUNTING EULER TOURS OF RANDOM REGULAR DIGRAPHS
Páidí Creed

A graph (resp. digraph) is said to be Eulerian iff all vertices are of even degree
(resp. all vertices have the same in-degree and out-degree). Every Eulerian graph
G has a set of circuits in which each edge is used exactly once, known as the
Euler tours of G. This talk is about the closely related problems of counting and
sampling Euler tours of an Eulerian graph.

For any directed Eulerian graph G, we can count the number of Euler tours
of G in polynomial time and there exist several polynomial time algo-
rithms for sampling from the uniform distribution on the Euler tours of G. However,
the complexity of counting Euler tours of an undirected Eulerian graph is \#P-
complete. Moreover, none of the sampling algorithms for the di-
rected case can be applied to sampling Euler tours of an undirected Eulerian
graph.

In this talk, I present a result characterising the asymptotic distribution of the
number of Euler tours of a random r-in/r-out graph. This can be used to show
that a simple algorithm can be used to sample or approximately count Euler tours
of almost every r-in/r-out graph in expected polynomial time. This algorithm
can also be used to sample or approximately count Euler tours of an undirected
Eulerian graph, and I will briefly mention work underway towards generalising
this result to the undirected case. In both cases, the approach used is the method
of conditioning on small subgraph counts pioneered by Robinson and Wormald
in their proof that almost every regular graph admits a Hamiltonian cycle. In
particular, this can be seen as an analogue of Frieze et al’s work on generating and
counting Hamiltonian cycles of random regular graphs.

THE COMPLEXITY OF COUNTING INDEPENDENT SETS MODULO k, WITH APPLI-
CATIONS TO CSP
John Faben

In 1979, Valiant introduced the complexity class \( \oplus P \), the problem of counting
the number of solutions to NP problems modulo two, and has since proved some
completeness results. In this talk we define the notion of completeness for count-
ing modulo integers other than 2, and we consider the complexity of counting the
number of independent sets in a graph in this sense. In fact, we will prove that this
problem is \( \#_k P \)-complete for all \( k \) even if the graphs are restricted to be bipartite.

This was a preliminary result in proving a dichotomy theorem for the com-
plexity of counting the number of solutions to Boolean Constraint Satisfaction
Problems modulo integers. That result builds on an earlier paper of Creignou and
Hermann which gave a counting dichotomy for these types of problem, and the
dichotomy itself is almost identical. Specifically, we have found that counting the number of solutions to a Boolean Constraint Satisfaction Problem can be done in polynomial time if all the relations are affine. Otherwise, except for one special case with $k = 2$, it is $\#P$-complete.

**LCP ALGORITHMS FOR DISCOUNTED GAMES**  
*John Fearnley in joint work with Marcin Jurdziński and Rahul Savani*

We study the performance of Lemke’s algorithm and the Cottle-Dantzig algorithm for P-Matrix LCPs is studied for the instances produced by the reduction from discounted games given by Jurdziński and Savani. Both algorithms are described purely in terms of the original discounted game, bypassing the reduction itself. A discounted game is given for which both the algorithms take an exponential number of steps, indicating that the algorithms perform no better for discounted games than they do for general P-matrix LCPs.

**DESCRIPTIVE COMPLEXITY OF OPTIMISATION PROBLEMS**  
*James Gate in joint work with Iain Stewart*

The field of Descriptive Complexity is the bridge between finite model theory and algorithmic complexity theory. The majority of research in this field has focused on classes of decision problems and the logics that capture them. This talk looks at how to extend these logics to capture optimisation problems. Specifically, it shall examine the class of (deterministic) polynomial time optimisation problems (referred to as $P_{opt}$) and argue that a single logical framework, which does not discriminate between maximisation and minimisation problems, is the most appropriate way to capture this class. Such a framework, using fixed-point operators along with examples of their use, shall be presented.

**EMBEDDING A FUNCTIONAL HYBRID MODELLING LANGUAGE IN HASKELL**  
*George Giorgidze in joint work with Henrik Nilsson*

Functional Hybrid Modelling (FHM) is a new approach to the design of non-causal modelling languages. The idea is to enrich a purely functional language with a few key abstractions for supporting hybrid, non-causal modelling. Our hypothesis is that the FHM approach will result in non-causal modelling languages that are relatively simple, have clear, purely declarative semantics, and, aided by this, advance the state of the art by supporting e.g. certain forms of meta-modelling and modelling and simulation of highly structurally dynamic systems. In this talk we present the first investigation into the implementation of an FHM language for non-causal modelling and simulation of physical systems. This is realised as a domain-specific language embedded in Haskell. The language facilitates construction and composition of model fragments given by systems of implicit differential algebraic equations. The method of embedding employs quasiquoting, thus demonstrating the effectiveness of this approach for languages that are not suitable for embedding in more traditional ways. Our implementation
The Bulletin of the EATCS

is available on-line, and thus the first publicly available prototype implementation of an FHM language.

A COMPLEXITY DICHOTOMY FOR HYPERGRAPH PARTITION FUNCTIONS
Leslie Ann Goldberg in joint work with Martin Dyer and Mark Jerrum

The talk will introduce partition functions, which arise in many computational contexts, and will discuss the complexity of computing them. It will explain what a dichotomy theorem is, and why we want such theorems (essentially, we want them so that we can better understand the boundary between the class of easy-to-compute functions and the class of functions that cannot be efficiently computed).

The particular technical problem which will be discussed, to some extent, is the complexity of counting homomorphisms from an \( r \)-uniform hypergraph \( G \) to a symmetric \( r \)-ary relation \( H \). (But you don’t need to know anything about that to understand the talk!) We give a dichotomy theorem for \( r > 2 \), showing for which \( H \) this problem is in \( FP \) and for which \( H \) it is \( \#P \)-complete. Our dichotomy theorem extends to the case in which the relation \( H \) is weighted, and the partition function to be computed is the sum of the weights of the homomorphisms. This problem is motivated by statistical physics, where it arises as computing the partition function for particle models in which certain combinations of \( r \) sites interact symmetrically.

LOGICS AND BISIMULATION GAMES FOR CONCURRENCY, CAUSALITY AND CONFLICT
Julian Gutierrez

Based on a simple axiomatization of concurrent behaviour we define two ways of observing parallel computations and show that in each case they are dual to conflict and causality, respectively. We give a logical characterization to those dualities and show that natural fixpoint modal logics can be extracted from such a characterization. We also study the equivalences induced by such logics and prove that they are decidable and can be related with well-known bisimulations for interleaving and noninterleaving concurrency. Moreover, by giving a game-theoretical characterization to the equivalence induced by the main logic, which is called Separation Fixpoint Logic (SFL), we show that the equivalence SFL induces is strictly stronger than a history-preserving bisimulation (hpb) and strictly weaker than a hereditary history-preserving bisimulation (hhpb). Our study considers branching-time models of concurrency based on transition systems and petri net structures.

ATTACKING AES VIA SAT
Matthew Gwynne in joint work with Oliver Kullmann

We consider the translation of the AES (the “Advanced Encryption Standard”, the successor of DES) into SAT.

In principle, many different questions regarding AES could be solved by a
SAT solver, such as the existence of weak keys, or questions regarding various uniqueness conditions, and therefore such a translation offers a perspective on the cryptanalysis of AES through the lens of satisfiability.

It seems likely, from similar work in the past with other ciphers (for example with DES), that one cannot expect too much at this time, and therefore we emphasise a modular open-source approach (as part of the OKlibrary - http://ok-sat-library.org - based on the Maxima/Lisp part of the library) which allows for small-scale variations, other experiments (for example replacing the S-box with other random permutations) and using general constraints as a target language (including polynomial equations and term based representations).

Such an approach will also allow us to integrate our methods with the approaches taken in the area using Groebner bases.

**FLIPPING REGULAR GRAPHS AND A PEER TO PEER NETWORK**

Andrew Handley in joint work with Martin Dyer and Colin Cooper

Mahlmann and Schindelhauer [1] defined a network that relies on random flip operations to keep its topology random regular, allowing it to repair damage and to embed new peers without over-complicated joining schema. It is then important to the protocol that the flip Markov chain mixes quickly enough. Work by Cooper, Dyer and Greenhill [2] gave a bound on the mixing time of the similar switch Markov chain, and this result was extended to acquire a loose polynomial bound on that of the flip Markov chain.

We dramatically tighten the mixing time bound using a two-stage direct canonical path construction. We go on to explore the behaviour of the protocol using simulations.


**PARAMETERIZED COMPLEXITY CLASSES UNDER LOGICAL REDUCTIONS**

Yuguo He in joint work with Anuj Dawar

In the theory of parameterized complexity, the $W$-hierarchy plays a role similar to NP in classical complexity theory in that many natural parameterized problems are shown intractable by being complete for some level $W[t]$ of the hierarchy. The classes $W[t]$ were originally defined as the sets of problems reducible to certain natural complete problems by means of fixed-parameter tractable ($fpt$) reductions. We investigate whether the classes can be characterised by means of weaker reductions, just like NP can. The latter is known to admit complete problems even
under quantifier-free first-order projections.

We consider reductions defined in terms of first-order interpretations and introduce a number of parameterized versions of these. Our main result is that each class $W[t]$ has complete problems under *slicewise bounded-variable first-order* reductions. These are a natural weakening of the slicewise bounded-variable LFP reductions which, by a result of Flum and Grohe [1], are known to be equivalent to *fpt*-reductions. If we relax the restriction on having a bounded number of variables, we obtain *slicewise first-order* reductions, which are not necessarily *fpt*. Indeed, we are able to show that any problem in $W[t]$ is reducible to some problem in $W[1]$ under such reductions—a result which if it held for *fpt*-reductions would imply the collapse of the $W$-hierarchy. On the other hand, we show that if we consider *slicewise quantifier-free first-order* reductions, they are considerably weaker in that there are problems in $W[t + 1]$ that cannot reduce to any problem in $W[t]$ under such reductions—a result which if it held for *fpt*-reductions would imply the strictness of the $W$-hierarchy and therefore the separation of P from NP.


EXPRESSIVE POWER OF RANK LOGICS

Bjarki Holm in joint work with Anuj Dawar, Bastian Laubner and Martin Grohe

Descriptive complexity theory studies the relationship between logic and computational complexity. One of the main open questions in this area is whether there exists a logic that can express exactly all the polynomial-time computable properties of finite structures. The work of Cai et al. [2], and later Gurevich and Shelah [3], established that fixed-point logic with counting ($\text{FP+C}$) is not expressive enough for this purpose. Atserias et al. [1] later showed that $\text{FP+C}$ has the further limitation that it cannot determine the solvability of systems of linear equations, a natural polynomial-time problem.

We show that all the known shortcomings of the logic $\text{FP+C}$ relate to its inability to determine the solvability of systems of linear equations; or more generally, its inability to find the row rank of a matrix. This leads us to consider extensions of first-order and fixed-point logics via operators for computing the row rank of definable matrices. We show that fixed-point logic with rank ($\text{FP+R}$) is strictly more expressive than $\text{FP+C}$ and it is an open question whether $\text{FP+R}$ can express all polynomial-time properties of finite structures. We also consider the expressive power of first-order logic with rank ($\text{FO+R}$) and show that in the presence of a linear ordering this logic captures the complexity class $\oplus L$, whose descriptive complexity had been previously unknown.

A HIGHER-ORDER OBSERVATIONAL EQUIVALENCE MODEL CHECKER

David Hopkins in joint work with Luke Ong

We have developed a tool, HOMER, which can check for observational equivalence of programs from the third-order fragment of Idealized Algol augmented with iteration. Idealized Algol (IA) is a functional programming language with imperative features, such as sequencing and local variables. It is effectively a call-by-name variant of core ML. Observational equivalence is a powerful notion of program equivalence. Two programs are equivalent when one can be substituted for the other in every program context without causing any observable difference in the outcome of the computation. To our knowledge HOMER is the first model checker for third-order programs.

HOMER relies on the fully abstract game semantics of IA. In games semantics a type $T$ is represented by a two-player game $[T]$. A term in context, $\Gamma \vdash M : T$, is then interpreted as a strategy, $[M]$, for the game $[\Gamma \vdash T]$. A powerful result of the game semantics is that two terms are observationally equivalent if and only if their respective strategies contain the same set of complete plays (a play is complete when it has been played to termination).

HOMER has two main components. The first maps a term $M$ to a Visibly Pushdown Automaton (VPA) which represents the strategy denotation $[M]$. VPA are a subclass of pushdown automata in which the stack action (push, pop or neither) is uniquely determined by the input symbol. Remarkably, VPA have closure properties almost as nice as the regular languages. The language accepted by the VPA produced is a precise representation of the set of complete plays in $[M]$. A play is a (suitably constrained) sequence of moves, each equipped with a justification pointer to an earlier move. By ignoring the pointers, a set of plays naturally forms a language. Unfortunately, for third-order terms, doing so results in a loss of precision, so we have to add tags to certain moves to encode where the pointers should go.

The second main component of HOMER is a VPA toolkit which we use to check for equivalence of VPA. Using complementation, intersection and an emptiness test we can check if two VPA accept the same language. If they are inequivalent, HOMER produces as counter-examples both a game-semantic play and an operational-semantic separating context (a context which terminates given one of the terms, but diverges with the other).

THE COMPLEXITY OF WEIGHTED BOOLEAN #CSP WITH MIXED SIGNS

Markus Jalsenius in joint work with Andrei Bulatov, Martin Dyer, Leslie Ann Goldberg and David Richerby
We give a complexity dichotomy for the problem of computing the partition function of a weighted Boolean constraint satisfaction problem (denoted weighted #CSP). Such a problem is parameterized by a set of rational functions, each of which assigns a weight to any configuration. A configuration is an assignment of values from \( \{0, 1\} \) to the variables in the instance, and the partition function is the sum of weights of all configurations. Our dichotomy extends previous work in which the weight functions were restricted to being non-negative. This extension is of particular interest because functions having mixed signs can cause cancellations in the partition function, and hence may make it easier to compute. We show that a weighted #CSP problem is either in \( \text{FP} \) (easy) or is \( \text{FP}^{\#P} \)-complete (hard). Not only do we have this dichotomy, it is also decidable.

In this talk we give an introduction to weighted #CSPs and and explain how many natural problems can be expressed as weighted #CSP problems with functions of mixed signs. We also give a characterisation of weighted #CSPs in order to determine whether the problem is in \( \text{FP} \) or is \( \text{FP}^{\#P} \)-complete.

**VERIFYING TRAIN CONTROL SOFTWARE**

**Phillip James in joint work with Markus Roggenbach**

Recently, in [1] K. Kanso developed and implemented a method to verify the control software of train stations. Given a program written in so-called ladder logic [2], general safety conditions would be broken down for the concrete layout of a train station, and—using SAT solving techniques [3]—it was verified if the program respected the conditions. We develop this approach further: first we turn the software from an application dedicated to a single train station into a generic product. Secondly, in the case the verification of a safety condition fails, we produce traces of counter examples, which shall help the engineer to locate the potential fault within the program.


**A COMPLEXITY DICHOTOMY FOR PARTITION FUNCTIONS WITH MIXED SIGNS**

**Mark Jerrum in joint work with Leslie Ann Goldberg, Martin Grohe and Marc Thurley**

Partition functions, also known as homomorphism functions, form a rich family of graph invariants that contain combinatorial invariants such as the number of \( k \)-colourings or the number of independent sets of a graph and also the partition functions of certain “spin glass” models of statistical physics such as the Ising model.
Building on earlier work by Dyer and Greenhill, and Bulatov and Grohe, we completely classify the computational complexity of partition functions. Our main result is a dichotomy theorem stating that every partition function is either computable in polynomial time or #P-complete. Partition functions are described by symmetric matrices with real entries, and we prove that it is decidable in polynomial time in terms of the matrix whether a given partition function is in polynomial time or #P-complete.

While in general it is very complicated to give an explicit algebraic or combinatorial description of the tractable cases, for partition functions described by a Hadamard matrices — these turn out to be central in our proofs — we obtain a simple algebraic tractability criterion, which says that the tractable cases are those “representable” by a quadratic polynomial over the field GF(2).

PROPERTY VERIFICATION OF AN ELECTRONIC PAYMENT SYSTEM: EP2

Temesghen Kahsai

The EP2 system is an electronic payment system and it stands for `EFT/POS 2000’ short for ‘Electronic Fund Transfer/Point of Service 2000’, is a joint project established by a number of (mainly Swiss) financial institutes in order to define the infrastructure for credit, debit and electronic purse terminals in Switzerland (www.eftpos2000.ch). The system consist of seven autonomous entities and they are centered around an EP2 Terminal. These entities communicate with the Terminal and, to a certain extent, with another via XML-messages in a fixed format. Each component is a reactive system defined by a number of use cases. The EP2 specification consists of 12 documents, each of which describe the different components or some aspect common to the components.

In this talk I will show the formalization of the EP2 specification in the formal specification language Csp-Casl [3]. Csp-Casl allows to formalize computational system in a combined algebraic / process algebraic notation. In [1] we introduced refinement notions for Csp-Casl. We verify the refinement of the different level of the EP2 specification and we prove some properties such as deadlock and livelock freedom using the interactive theorem prover Csp-Casl-Prover [2].


AUTOMATED GENERATION OF VERIFIED RAILWAY INTERLOCKING SYSTEMS

Karim Kanso in joint work with Anton Setzer and Peter Mosses

In recent years the use of safety critical computer controlled industrial systems has increased. The railway domain is no exception. These large safety critical
The Bulletin of the EATCS

systems are very hard to produce requiring many man hours during development and testing. This project aims to automate the development and verification of railway interlocking systems, such that formal proofs are produced ensuring safety and liveliness properties. Briefly, this entails specifying and modelling the railway (Railway Domain Model) as done with domain engineering. Then, the next step is to create a view of this model for the interlocking system in question. Different views of this model can be taken for different interlocking systems or entirely different purposes.

AN ALGORITHM FOR FINDING k-VERTEX OUT-TREES AND ITS APPLICATION TO k-INTERNAL OUT-BRANCHING PROBLEM
EunJung Kim in joint work with Nathann Cohen, Fedor V. Fomin, Gregory Gutin, Saket Saurabh and Anders Yeo

An out-tree is an oriented tree with only one vertex of in-degree zero called the root. The k-Out-Tree problem is the problem of deciding for a given parameter k, whether an input digraph contains a given out-tree with k ≥ 2 vertices. In their seminal work on Color Coding Alon, Yuster, and Zwick provided fixed-parameter tractable (FPT) randomized and deterministic algorithms for k-Out-Tree. While Alon, Yuster, and Zwick only stated that their algorithms are of runtime $O(2^{O(k)}n)$, however, it is easy to see that their randomized and deterministic algorithms are of complexity $O^*((4e)^k)$ and $O^*((c^k)$, where $c ≥ 4e$.

The main results of Alon, Yuster, and Zwick, however, were a new algorithmic approach called Color Coding and a randomized $O^*((2e)^k)$ algorithm for deciding whether a digraph contains a path with $k$ vertices (the $k$-Path problem). Recently Chen et al. and Kneis et al. developed an approach, called Divide-and-Color, that allowed them to design a randomized $O^*((4e)^k)$-time algorithm for $k$-Path. Divide-and-Color in Kneis et al. (and essentially in Chen et al.) is ‘symmetric’, i.e., both colors play similar role and the probability of coloring each vertex in one of the colors is 0.5. We further develop Divide-and-Color by making it asymmetric, i.e., the two colors play different roles and the probability of coloring each vertex in one of the colors depends on the color. As a result, we refine the result of Alon, Yuster, and Zwick by obtaining randomized and deterministic algorithms for k-Out-Tree of runtime $O^*(5.7^k)$ and $O^*(5.7^{k+o(k)})$ respectively.

We apply the above deterministic algorithm to obtain a deterministic algorithm of runtime $O^*(c^k)$, where $c$ is a constant, for deciding whether an input digraph contains a spanning out-tree with at least $k$ internal vertices. This answers in affirmative a question of Gutin, Razgon and Kim (Proc. AAIM’08).

ON COMPUTATIONAL EXPERIMENTS WITH ARITHMETIC PROGRESSIONS
Oliver Kullmann

How large needs the natural number $n$ be, so that, however $\{1, \ldots, n\}$ is partitioned into two parts, at least one part will contain an arithmetic progression of
length 3? $n = 9$ is sufficient, while $n = 8$ won’t do (an instructive exercise for the reader).

This is an example for the smallest non-trivial van der Waerden number, and these numbers have been studied extensively since van der Waerden proved their existence in 1927. Computing precise values is a daunting task, and we give an overview on what has been achieved (and how).

Special emphasis is put on the considerations put forward by additive number theory, which in this context replaces the set $\{1, \ldots, n\}$ by the set of the first $n$ prime numbers and otherwise asks the same questions, based on the Green-Tao theorem (2004), and I will give first experimental results on these Green-Tao numbers.

The main computational methods are based on SAT solving, in the context of the OKlibrary (http://www.ok-sat-library.org/). Depending on the parameter values, very different approaches perform best, which is an interesting topic in itself.

**CONSISTENCY STATEMENTS IN THE FRAGMENTS OF EQUATIONAL THEORY PV**

Ebrahim A Larijani

Bounded Arithmetic is a subsystem of Peano Arithmetic which is strongly related to the computational complexity classes. Functions which are computable by polynomial Turing Machine are exactly the functions which are definable in the specific theories of Bounded Arithmetic. Consequently by separating theories in the hierarchy of Bounded Arithmetic we could understand the situation in the separation of Polynomial Hierarchy namely $P \neq NP$ which is the main open problem in the computational complexity theory.

In this talk I will explain one approach to the separation problem of Bounded Arithmetic based on examining consistency statements in the weak theories of arithmetic and I will discuss possible ways of improving results concerning consistency of fragments of equational theory PV.

**UML SPECIFICATION AND CORRECTION OF OBJECT-ORIENTED ANTI-PATTERNS**

Maria Teresa Llano in joint work with Rob Pooley

Nowadays, the detection and correction of software defects has become a very hard task for software engineers. Due to the constant evolution of the industry, the technology and systems that support its operation are required to fit into a constantly changing environment. Most importantly, the lack of standard specifications of these software defects alongside with the lack of tools for their detection, correction and verification enforces developers to perform manual modifications, incurring not only in mistakes, but also in costs of time and resources.

The work presented here is aimed at the study of the specification and correction of a particular type of software defects: Object-Oriented Antipatterns. More specifically, we have defined a UML specification of antipatterns and established
guidelines for their correction process through the use of rewrite rules. With this specification we expect to open up the possibility to automate the detection and correction of this kind of software defects.

**FPT ALGORITHMS FOR THE MAXIMUM INDEPENDENT SET AND MAXIMUM CLIQUE PROBLEMS**

Vadim V. Lozin

We study the **MAXIMUM INDEPENDENT SET** and **MAXIMUM CLIQUE** problems parameterized by the solution size $k$. A parameterized problem is fixed-parameter tractable (fpt for short) if it can be solved in $f(k)n^{O(1)}$ time, where $f(k)$ is a computable function depending on the value of the parameter only. In general, both problems are $W[1]$-hard, which means they are not fixed-parameter tractable unless $P = NP$. On the other hand, fpt-algorithms have been developed for the **MAXIMUM INDEPENDENT SET** problem in the classes of triangle-free graphs, graphs of bounded vertex degree, segment intersection graphs with bounded number of directions, planar graphs, and more generally, graphs excluding a single-crossing graph as a minor [2]. Therefore, the **MAXIMUM CLIQUE** problem admits fpt-algorithms in the complements of these graphs.

A common feature of all these classes is that all of them are hereditary (i.e., closed under vertex deletion) and all of them are small in the following sense. It is known [1] that for every hereditary class $X$, the number $X_n$ of $n$-vertex graphs in $X$ (also known as the speed of $X$) satisfies $\lim_{n \to \infty} \frac{\log X_n}{n^k} = 1 - \frac{1}{k}$, where $k(X)$ is a natural number called the index of the class. The triangle-free graphs have index 2 and the index of all other classes mentioned above is 1.

We focus on hereditary classes of index $k > 1$. Each class in this range can be approximated by a minimal class of the same index. Our main result is that both problems are fixed-parameter tractable in all minimal classes of index $k$ for all values of $k$.


**KEEPING PARTNERS TOGETHER: ALGORITHMIC RESULTS FOR THE HOSPITALS/RESIDENTS PROBLEM WITH COUPLES**

David Manlove in joint work with Eric McDermid

The classical Hospitals / Residents problem (HR) has many practical applications: in particular it models the assignment of junior doctors to hospitals, which is carried out by large-scale centralised matching schemes in many countries. The Hospitals / Residents problem with Couples (HRC) is a generalisation of HR that models the important case where couples submit joint preference lists over pairs of hospitals $(h_i, h_j)$ that are typically geographically close.
We consider a natural restriction of HRC in which the members of a couple wish to be placed at the same hospital, i.e., \( h_i = h_j \) for every such pair. We show that, in this context, the problem of deciding whether a stable matching exists is NP-complete, even if each resident’s preference list is of length at most 3 and each hospital has capacity at most 2. However we show that if each hospital’s preference list is of length at most 2, then a stable matching always exists and can be found in linear time (for arbitrary hospital capacities).

We also consider a more general restriction of HRC in which the members of a couple have individual preference lists over hospitals, and the joint preference list of the couple is consistent with these individual lists in a precise sense. In this context, with respect to classical (Gale-Shapley) stability, we give a linear-time algorithm to find a stable matching or report that none exists, regardless of the preference list lengths or the hospital capacities.

NEW PARAMETERIZED COMPLEXITY RESULTS FOR PROBLEMS ON DEGENERATE GRAPHS

Luke Mathieson in joint work with Stefan Szeider

We present some new parameterized results on the theme of degree constrained graphs, in particular degenerate graphs. We show that restricting inputs of problems to degenerate graphs often results in fixed-parameter tractable algorithms. Previously we had demonstrated the fixed-parameter tractability of a family of problems involving editing to achieve a regular graph, in the case of degenerate graphs we show that such editing problems are W[1]-complete, even when the desired degeneracy is a constant, and are thus unlikely to be fixed-parameter tractable. Furthermore we show that there also exist simple problems that remain hard when restricted to degenerate graphs, even when the degeneracy is a constant.

A \( \frac{2}{3} \)-APPROXIMATION ALGORITHM FOR GENERAL STABLE MARRIAGE

Eric J. McDermid

An instance of the stable marriage problem with ties and incomplete lists (SMTI) involves a set of \( n \) men and \( n \) women, each of whom provides a ranking of a subset of the members of the opposite sex in the form of a preference list. A man or a woman’s preference list may contain ties, which are sets of agents all having the same rank. A matching \( M \) of men to women is stable if there is no pair \((m, w) \notin M\) such that \( m \) and \( w \) prefer each other to their situation in \( M \).

It is known that every SMTI instance admits at least one stable matching, however stable matchings can have different cardinalities. It is APX-hard to compute a maximum cardinality stable matching, but there have recently been proposed polynomial-time approximation algorithms, with constant performance guarantees for both the general version of this problem, and for several special cases. Our contribution is to describe a \( \frac{2}{3} \)-approximation algorithm for the general version of this problem, improving upon the recent \( \frac{2}{3} \)-approximation algorithm of
Király. Interest in such algorithms arises because of the problem’s application to centralized matching schemes, the best known of which involve the assignment of graduating medical students to hospitals in various countries.

**SOLVING SIMPLE STOCHASTIC GAMES WITH INTERIOR POINT METHODS**

Julian Merschen

We introduce simple stochastic games (SSG) and give reasons why it is of interest to find the optimal strategy when considering a one-player SSG. In this setup, the optimal strategy can be found by solving a linear program (LP). We show that the constraint matrix of the corresponding linear program is an integer matrix. We introduce Vavasis’ and Ye’s layered interior point algorithm, the running time of which is polynomial and only depends on the encoding of the constraint matrix of the corresponding LP. As the encoding of the corresponding linear problem is polynomially bounded in the dimension of the SSG this known algorithm solves the LP in strongly polynomial time. Next we turn to solving SSG when both players are present. It is known that this problem can be rewritten as a linear complementarity problem with a P-matrix (LCP). Under certain conditions LCP can be solved in polynomial time using interior point methods. We analyze the feasibility of these conditions when solving SSG with two players.

**THE ITERATED PRISONER’S DILEMMA ON A CYCLE**

Velumailum Mohanaraj in joint work with Martin Dyer

Prisoner’s dilemma is a two-person game widely used as a metaphor for the evolution of cooperation among selfish agents. Many strategies have been studied in this context. A particular strategy, called Pavlov, has been shown to have some advantages. However, the Pavlov strategy leaves room to be exploited. We modify this strategy by introducing some stochasticity, thereby reducing the possibility of exploitation. We call the resulting strategy Rational Pavlov. This has a parameter \( p \) which measures the "degree of forgiveness" of the players. We study the evolution of cooperation in the Iterated Prisoner’s Dilemma game, when \( n \) players are arranged in a cycle, and all play this strategy. We examine the effect of varying \( p \) on the time taken for complete cooperation to emerge. We prove that the convergence rate is fast, \( O(n \log n) \) time, for high values of \( p \), but exponentially slow in \( n \) for low values of \( p \). Our analysis leaves a gap in the range of \( p \), but simulations suggest that there is, in fact, a sharp phase transition.

**MODULAR TYPE SYSTEMS**

Mark New

Type systems for programming languages are commonly specified using a big-step variant of the structural operational semantics (SOS) framework. There are two ways in which a type system can be specified: declaratively, or algorithmically – the latter is ‘syntax directed’, with the rules corresponding directly to a particular type checking algorithm.
We are investigating a component-based approach to language description, which involves analysing the constructs of each language in terms of abstract, language-independent constructs. For each abstract construct there should be a simple and unique typing rule, which should not require reformulation when constructs are combined. It appears that this requires declarative typing rules. However, it has been argued that certain aspects of programming languages such as Java can only be specified algorithmically.

After outlining the differences between algorithmic and declarative typing, this talk will discuss potential solutions to the problem outlined above.

A NEW APPROACH FOR AUTOMATA COORDINATION ON $Z^2$
Thomas Nickson in joint work with Igor Potapov and Russell Martin

Swarm robotics is generally categorised by simple robots, usually of unknown quantity, with access only to local information, limited resources and each with the same algorithm. The aims are to produce a collected response from robots that are in essence oblivious to the global aims which they are a part of. In this talk a method for such coordination is described. Automata are modelled on an integer grid as a cluster of unknown size and shape. Inspiration is taken from harmonic resonance phenomena which exemplifies how complex and consistent global results may be formed from only local interactions. With an initial case of two robots producing a regular pattern of waves, showing a variety of relatively complex patterns, through superposition of these waves, can be produced on a plane of very simple robots with the ability to communicate with others in their immediate neighbourhood. These patterns create points of reference and a breach of symmetry throughout the system which can be exploited for a number of organisational duties including orientation, leader election and many other situations in which a break in symmetry can be exploited. Essentially this algorithm can be seen as a form of preprocessing easing the difficulties of other operations required to manipulate the robots. Extensions to cellular automata may also be explored.

LUCIAN: DATAFLOW AND OBJECT-ORIENTATION
Dominic Orchard

There are a multitude of programming languages in existence. Why? Because no single language or paradigm can be all things to all people. Whilst one class of programs may be succinctly written, easily reasoned about, and efficiently compiled within one language another class of programs may be impenetrably intractable for the human and the compiler. Language interoperability and multi-paradigm languages are the means by which programmers can get the best of both worlds.

This talk introduces the Lucian programming language, a cross-paradigm derivative of the Lucid dataflow language, that interoperates declarative dataflow and object-orientation. Programs that are dynamic or reactive can be succinctly ex-
pressed within Lucid. However, not all parts of such a program are necessarily
easy to express in dataflow form. Lucian provides an escape to an imperative
object-oriented language where subparts of a program may be more easily written
and compiled. Conversely Lucian provides a way to write programs using objects
in Lucid’s dataflow equation style.

This talk introduces Lucid for the uninitiated viewer and proceeds to introduce
the central constructs of Lucian. A comparison of the underlying computational
models of Lucid and object-orientation is given to explain the appropriateness of
this interoperation.

**STRUCTURED THEOREM PROVING FOR Csp-Casl**

Liam O’Reilly in joint work with Markus Roggenbach

At the last WADT T. Mossakowski and M. Roggenbach [1] suggested Csp-
Casl[2] as an institution, this construction gives rise to the possibility of structure
Csp-Casl specifications.

In our talk we will discuss how to implement the structuring mechanisms for
Csp-Casland demonstrate how to use these structuring mechanisms for the com-
positional verification of systems specified in Csp-Casl.

Andrea Corradini and Fabio Gadducci, editors, WADT 2008 – Preliminary Proceed-

[2] Markus Roggenbach. CSP-CASL - a new integration of process algebra and alge-

**COUNTING INTERVAL SIZES VS. COUNTING NONDETERMINISTIC COMPUTATION PATHS**

Aris Pagourtzis in joint work with E. Bampas, A. Göbel, and A. Tentes

We investigate the complexity of hard counting problems that belong to the
class #P but have easy existence version; several well-known problems such as
#Perfect Matchings, #DNFSat share this property. We focus on classes of such
problems which emerged through two disparate approaches. In the first one,
taken by Hemaspaandra et al. [1], they define classes of functions that count the
size of intervals of ordered strings, where the underlying order is assumed to be
polynomial-time decidable. They consider both partial and total orders, and they
also investigate the case where the adjacency query on the underlying order is de-
cidable in polynomial-time. They characterize #P as the class of functions that
count the size of intervals of polynomial-time decidable total orders. They also
characterize the class of #P functions with easy existence version, as the class
of functions that count the size of intervals of polynomial-time decidable partial
orders with efficient adjacency checks.

In the second approach, by Kiayias et al. [2], they define the class TotP, con-
sisting of functions that count the total number of paths of NP computations.
Pagourtzis and Zachos [3] show that the Karp-closure of TotP coincides with the set of self-reducible #PE functions, under a natural notion of self-reducibility, thus containing many natural #P functions with easy existence version such as the ones mentioned above.

In this work, we define interval size counting classes on orders with increasingly strong feasibility constraints. We provide inclusion and separation relations between TotP and these classes. Among others, we are able to give suitable feasibility constraints that characterize TotP and FP as interval size counting classes. Our results imply that many known #P-complete problems with easy decision are contained in the classes defined in [1]—but are unlikely to be complete for these classes under certain types of reductions. We also define a new class of interval size functions which lies strictly between FP and TotP under reasonable complexity-theoretic assumptions, and we show that it contains some hard counting problems.


ON THE (SEMI)LATTICES INDUCED BY CERTAIN REDUCIBILITIES
Arno Pauly

A natural approach to computability on uncountable sets such as infinite words or real numbers involves approximation by elements of a countable set. Thus approximability, formalized as continuity, of functions is of great relevance to theoretical computer science. In particular, in computable analysis [2], continuity appears to be a straightforward generalization of computability.

In this setting, we study a continuous equivalent to bounded Turing reducibility, limited to a single oracle query. For two functions $f, g, f \leq_2 g$ holds, iff continuous functions $F, G$ exist with $f(x) = F(x, g(G(x)))$ for all $x$ in the domain of $f$. We will show that the partial ordered class of equivalence classes regarding $\leq_2$ is a complete join-semilattice, that is all suprema exist.

Another interesting reducibility corresponds to many-one reducibility, where $f \leq_0 g$ holds, iff there is a continuous function $G$ with $f = g \circ G$. As will be demonstrated, the equivalence classes for $\leq_0$ even form a complete lattice, so all suprema and infima exist.

The ability to construct suprema and infima facilitates the study of different degrees of discontinuity and incomputability of functions defined on uncountable sets.
The results presented here are a special case of those given in [1], where proofs can also be found.


GENERALIZED MATCHING
Alexandru Popa in joint work with Raphael Clifford, Aram Harrow and Benjamin Sach

Given a pattern \( p \) over an alphabet \( \Sigma_p \) and a text \( t \) over an alphabet \( \Sigma_t \), we consider the problem of determining a mapping \( f \) from \( \Sigma_p \) to \( \Sigma_t^+ \) such that \( t = f(p_1)f(p_2)\ldots f(p_m) \). This class of problems, which was first introduced by Amir and Nor in 2004, is defined by different constraints on the mapping \( f \). We give NP-Completeness results for a wide range of conditions. These include both when \( f \) is function or a bijection, when \( \Sigma_t \) is binary and when the range of \( f \) is limited to strings of constant length. We then introduce a related problem we term pattern matching under string classes which we show to be solvable efficiently. Finally, we discuss an optimization variant of generalized matching and give a polynomial time \( \sqrt{\text{opt}/k} \)-approximation algorithm for fixed \( k \).

FUNCTIONS DEFINABLE BY ARITHMETIC CIRCUITS
Ian Pratt-Hartmann in joint work with Ivo Düntsch

An arithmetic circuit is a labelled, directed graph specifying a cascade of arithmetic and logical operations to be performed on sets of non-negative integers (henceforth: numbers). Each node in this graph evaluates to a set of numbers, representing a stage of the computation performed by the circuit. Nodes without predecessors in the graph are called input nodes, and are labelled with any of the symbols \( \{1\}, \{0\}, \emptyset \) or \( \mathbb{N} \), denoting a set of numbers in the conventional way, or, alternatively, with a variable ranging over sets of numbers. Nodes with predecessors in the graph are called arithmetic gates, and are labelled with any of the symbols \( +, \cdot, \bar{\ }, \cap \) or \( \cup \), denoting an operation on sets. The symbols \( -.\), \( \cap \), \( \cup \) have the obvious Boolean interpretations (with \( \bar{\ } \) denoting complementation in \( \mathbb{N} \)), while \( + \) and \( \cdot \) denote the result of lifting addition and multiplication to the algebra of sets.

In this talk, we consider the expressive power of arithmetic circuits. In particular, we ask: which functions (from tuples of sets of numbers to sets of numbers) are definable by arithmetic circuits? We obtain two negative results: the first shows, roughly, that a function is not circuit-definable if it has an infinite range and sub-linear growth; the second shows, roughly, that a function is not circuit-definable if it has a finite range and fails to converge on certain ‘sparse’ chains under inclusion. We observe that various functions of interest fall under these descriptions. In particular, arithmetic circuits can compute remainders (on division by a constant) but not quotients; they can determine whether a set is a singleton.
but not whether it is finite; they can compute the minimum of a set, but not the maximum of a finite set, or its cardinality, or its sum.

This work is supported by the EPSRC, grant number EP/F069154/1.

**FLEXIBLE BUSINESS PROCESSES USING STPOWLA**

Stephan Reiﬀ-Marganiec

Service Oriented Computing is a paradigm for developing software systems as the composition of a number of services. Services are loosely coupled entities, that can be dynamically published, discovered and invoked over a network. The engineering of such systems presents novel challenges, mostly due to the dynam- icity and distributed nature of service-based applications. In this paper, we focus on the modelling of service orchestrations. We discuss the relationship between two languages developed under the Sensoria project: SRML as a high level modelling language for Service Oriented Architectures, and StPoWLa as a process-oriented orchestration approach that separates core business processes from system variability at the end-user’s level, where the focus is towards achieving business goals. A fundamental challenge of software engineering is to correctly align business goals with IT strategy, and as such we present an encoding of StPoWLa to SRML. This provides a formal framework for StPoWLa and also a separated view of policies representing system variability that is not present in SRML.

**AN INTRODUCTION TO THE COMPLEXITY OF CONSTRAINT SATISFACTION PROBLEMS**

David Richerby

I will give a brief survey of the computational complexity of the constraint satisfaction problem (CSP). We are given a set of variables, to which we can assign values from some finite domain, along with a set of constraints, expressed using relations over the domain. Three versions of CSP will be considered.

- **Decision CSP:** can we assign values to the variables so that all the constraints are satisfied simultaneously?
- **Counting CSP:** how many distinct variable assignments satisfy all the constraints?
- **Weighted CSP:** generalizes constraints to functions expressing the probability or desirability of variable configurations.

**ONLINE APPROXIMATE MATCHING WITH NON-LOCAL DISTANCES**

Benjamin Sach in joint work with Raphaël Clifford

A black box method was recently given that solves the problem of online approximate matching for a class of problems whose distance functions can be classified as being local. A distance function is said to be local if for a pattern $P$ of length $m$ and any substring $T[i, i + m]$ of a text $T$, the distance between $P$ and $T[i, i + m]$ is equal to $\sum \Delta(P[j], T[i + j])$, where $\Delta$ is any distance function between individual characters. We extend this line of work by showing how to tackle online approximate matching when the distance function is non-local.
In our model we assume that we are given a pattern in advance and the text to which it is to be matched arrives one character at a time. The overall task is to report matches between the pattern and text as soon as they occur and to bound the worst case time per input character. It is an important feature of both our approach and the previous work that the running time of the resulting algorithms is not amortised.

In the talk, we will present an overview of this work and briefly discuss the methods used. Our solutions are applicable to a wide variety of matching problems including function and parameterised matching, swap matching, swap-mismatch, k-difference, k-difference with transpositions, overlap matching, edit distance/LCS, flipped bit, faulty bit and $L_1$ and $L_2$ rearrangement distances. The resulting algorithms bound the worst case running time to within a log factor of their comparable offline counterpart.

**HOW TO AVOID WASTING PARALLEL PERFORMANCE**

András Z. Salamon

Parallel processing aims to reduce the amount of time (makespan) that a computation requires, by using additional processors. When a parallel computation is expressed using environments such as Khronos OpenCL or Mathematica, one is generally forced to sequence activities one after the other, or to break parts of the computation into independent activities. This imposes a series-parallel structure on the activity network that underlies the computation. How does this affect the performance that is achievable?

I consider a simple activity network model of parallel computation, consisting of a partial order with node weights, representing activity durations. Edges mean that one activity must complete before another may start, and are called precedence constraints. When expressing a computation in a parallel programming environment, one implicitly chooses which precedence constraints are added to the activity network to make it series-parallel. Adding precedence constraints cannot decrease the makespan, but can one always series-parallelise without increasing the makespan too much?

I conjecture that a $4/3$ increase is always achievable, but that achieving this bound is NP-complete. The $4/3$ bound is tight. On the other hand, a linear algorithm achieves an increase that is bounded by the ratio between the largest and smallest activity durations. The activity network model suggests that it is possible to avoid wasting one-third of the potential performance gains of parallel processing by using more expressive language constructs, by carefully choosing what is an activity, and by introducing some redundancy.

**AN ALGORITHMIC AND GRAPH THEORETIC VIEWPOINT OF SECURITY**

Paul Sant in joint work with Tim French and Nik Bessis

As we move into a world in which we are increasingly dependent upon tech-
Many models of trust have been proposed, but there is still a need to merge methods for calculating trust values with ideas from, for example, ideas from the social sciences (e.g. semiotics).

In this talk I will discuss some initial ideas and results for models of trust that approach the problem from an algorithmic perspective. There will also be a discussion about the benefits of using the framework of semiotics to add value to models of trust.

The talk will conclude with a discussion of open issues in the area.

SAFE FUNCTIONAL REACTIVE PROGRAMMING THROUGH DEPENDENT TYPES
Neil Sculthorpe in joint work with Henrik Nilsson

Functional Reactive Programming (FRP) is an approach to reactive programming where systems are structured as networks of functions operating on signals. FRP is based on the synchronous data-flow paradigm and supports both continuous-time and discrete-time signals (hybrid systems). What sets FRP apart from most other languages for similar applications is its support for systems with dynamic structure and for higher-order reactive constructs.

Statically guaranteeing correctness properties of programs is an attractive proposition. This is true in particular for typical application domains for reactive programming such as embedded systems. To that end, many existing reactive languages have type systems or other static checks that guarantee domain-specific properties, such as feedback loops always being well-formed. However, they are limited in their capabilities to support dynamism and higher-order data-flow compared with FRP. Thus, the onus of ensuring such properties of FRP programs has so far been on the programmer as established static techniques do not suffice.

Here, we show how dependent types allow this concern to be addressed. By embedding an implementation of FRP in the the dependently-typed language Agda, we can use the type system of the host language to craft a domain-specific (dependent) type system for FRP. As the implementation passes the Agda type, coverage, and termination checks, we know our type system is safe.

APPROXIMATING NODE-WEIGHTED MULTICAST TREES IN WIRELESS AD-HOC NETWORKS
Ambreen Shahnaz in joint work with Thomas Erlebach

Multicast communication in a wireless ad-hoc network can be established using a tree that spans the multicast sender and receivers as well as other intermediate nodes. If the network is modelled as a graph, the multicast tree is a Steiner tree, the multicast sender and receivers correspond to terminals, and other nodes participating in the tree are Steiner nodes. As Steiner nodes are nodes that participate in the multicast tree by forwarding packets but do not benefit from the
multicast, it is a natural objective to compute a tree that minimizes the total cost of the Steiner nodes. We therefore consider the problem of computing, for a given node-weighted graph and a set of terminals, a Steiner tree with Steiner nodes of minimum total weight. The problem is defined as follows: Given an undirected graph $G = (V, E)$ with nonnegative weights $w_v$ for $v \in V$ and a subset of nodes $K \subseteq V$ called terminals, compute a Steiner tree for $G$ and $K$, i.e., a subgraph $T$ of $G$ that is a tree and contains all the nodes in $K$. The objective is to minimize the total weight of the vertices of $T$. We can assume without loss of generality that the terminals have weight 0 (they are present in any solution and their weight increases the objective value of any solution by the same amount), so our goal is to minimize the total weight of the Steiner nodes of $T$. For graph classes that admit spanning trees of maximum degree at most $d$, we obtain a 0.775d-approximation algorithm.

We show that this result implies a 3.875-approximation algorithm for unit disk graphs, an $O(1/\alpha^2)$-approximation algorithm for $\alpha$-unit disk graphs, and an $O(\lambda)$-approximation algorithm for $(\lambda + 1)$-claw-free graphs. Unit disk graph is a simplified and idealistic model, whereas $\alpha$-unit disk graph is a more general graph model for wireless ad-hoc networks. $(\lambda + 1)$-claw-free graphs include bounded independence graphs, which also better reflect realistic wireless ad-hoc networks.

**FA-PRESENTABLE STRUCTURES**

Richard M Thomas in joint work with Alan Cain, Graham Oliver and Nik Ruškuc

We are interested in the notion of computing in structures (where a structure consists of a set together with a collection of relations). The natural approach would be to take some general model of computation (such as a Turing machine). A structure would then be said to be “computable” if its domain can represented by a set which is accepted by a Turing machine and if there are decision-making Turing machines for each of its relations. However, there have been various ideas put forward to restrict the model of computation used; whilst the range of possible structures decreases, the computation can become more efficient and certain properties of the structures may become decidable.

One interesting approach was introduced by Khoussainov and Nerode who considered structures whose domain and relations can be checked by finite automata (as opposed to Turing machines); such a structure is said to be “FA-presentable”. This was inspired, in part, by the theory of “automatic groups” introduced by Epstein et al; however, the definitions are somewhat different.

We will survey some of what is known about FA-presentable structures, contrasting it with the theory of automatic groups and posing some open questions. The talk is intended to be self-contained, in that no prior knowledge of these topics is assumed. We will be concentrating on some recent results on FA-presentable semigroups (joint work with Alan Cain, Graham Oliver and Nik Ruškuc).
PARTITIONING GRAPHS INTO CONNECTED PARTS
Pim van 't Hof in joint work with Daniël Paulusma and Gerhard J. Woeginger

The 2-DISJOINT CONNECTED SUBGRAPHS problem asks if a given graph has two vertex-disjoint connected subgraphs containing prespecified sets of vertices. We show that this problem is already NP-complete if one of the sets has cardinality 2. The LONGEST PATH CONTRACTIBILITY problem asks for the largest integer \( \ell \) for which an input graph can be contracted to the path \( P_\ell \) on \( \ell \) vertices. We show that the computational complexity of the LONGEST PATH CONTRACTIBILITY problem restricted to \( P_\ell \)-free graphs jumps from being polynomially solvable to being NP-hard at \( \ell = 6 \), while this jump occurs at \( \ell = 5 \) for the 2-DISJOINT CONNECTED SUBGRAPHS problem. We also present an exact algorithm that solves the 2-DISJOINT CONNECTED SUBGRAPHS problem faster than \( O^*(2^n) \) time for any \( P_\ell \)-free graph. For \( \ell = 6 \), its running time is \( O^*(1.5790^n) \). We modify this algorithm to solve the LONGEST PATH CONTRACTIBILITY problem for \( P_6 \)-free graphs in \( O^*(1.5790^n) \) time.

PROPERTY SPECIFICATIONS FOR WORKFLOW MODELLING
Peter Y. H. Wong in joint work with Jeremy Gibbons

Formal developments in workflow languages allow developers to describe their work flow systems precisely, and permit the application of model checking to automatically verify models of their systems against formal specifications. One of these workflow languages is the Business Process Modelling Notation (BPMN), for which we previously provided two formal semantic models in the language of Communicating Sequential Processes. Both models leverage the refinement orderings that underlie CSP’s denotational semantics, allowing BPMN to be used for specification as well as modelling of workflow processes. However, the expressiveness of BPMN is strictly less than that of CSP, and as a result some behavioural properties, against which developers might be interested to verify their workflow processes, might not be easy or even possible at all to capture in BPMN.

In this talk we will consider a pattern-based approach to expressing behavioural properties. We will describe a property specification language \( PL \) for capturing a generalisation of Dwyer et al.’s Property Specication Patterns, and present a translation from \( PL \) into a bounded, positive fragment of linear temporal logic, which can then be automatically translated into CSP for simple refinement checking. We demonstrate its application via a simple example.

This work is supported by a grant from Microsoft Research.

THE EXPRESSIVE POWER OF BINARY SUBMODULAR FUNCTIONS
Stanislav Živný in joint work with David Cohen and Peter Jeavons

It has previously been an open problem whether all Boolean submodular functions can be decomposed into a sum of binary submodular functions over a possibly larger set of variables. This problem has been considered within several
different contexts in computer science, including computer vision, artificial intelligence, and pseudo-Boolean optimisation. Using a connection between the expressive power of valued constraints and certain algebraic properties of functions, we answer this question negatively. Our results have several corollaries. First, we characterise precisely which submodular functions of arity 4 can be expressed by binary submodular functions. Next, we identify a novel class of submodular functions of arbitrary arities which can be expressed by binary submodular functions, and therefore minimised efficiently using a so-called expressibility reduction to the Min-Cut problem. More importantly, our results imply limitations on this kind of reduction and establish for the first time that it cannot be used in general to minimise arbitrary submodular functions. Finally, we refute a conjecture of Promislow and Young on the structure of the extreme rays of the cone of Boolean submodular functions.
Reports from Projects
AUTOMATA: FROM MATHEMATICS TO APPLICATIONS
REPORT ON THE AUTOMathA PROGRAMME

Abstract

AutoMathA (Automata: from Mathematics to Applications) is an international research programme of the European Science Foundation (ESF). As it lies at the crossroad of mathematics, theoretical computer science and applications, it catalyses progress in both theoretical and practical directions.

The main activities of the programme are to provide full financial support for visits/exchanges among the programme participants (usually for short periods, typically two weeks), to organize conferences, schools and workshops for programme participants and to sponsor conferences in the area of AutoMathA.

The programme was launched May 2005 and will end December 2010.

In the sequel we recall the general framework of the programme and present the events sponsored so far by the programme. All applications should be submitted via the online application forms that are available on the AutoMathA webpage (www.esf.org/automatha).

Topics

Automata theory is one of the longest established areas in computer science. Over the past few years automata theory has not only developed in many different directions but has also evolved in an exciting way at several levels: the exploration of specific new models and applications has at the same time stimulated a variety of deep mathematical theories. Standard applications of automata theory include pattern matching, syntax analysis and software verification. In recent years, novel applications of automata-theoretic concepts have emerged from biology, physics, cognitive sciences, neurosciences, control, tomography, linguistics, mathematics, etc., while developments in information technology have increased the need for formally based design and verification methods to cope with emerging technical needs such as network security, mobile intelligent devices and high performance...
computing. At the same time, the mathematical foundations of automata theory rely on more and more advanced areas of mathematics. While in the early 1960s only elementary graph theory and combinatorics were required, new tools from non-commutative algebra (semigroups, semirings and formal power series), logic, probability theory and symbolic dynamics have been successfully introduced and the latest developments borrow ideas from topology and geometry. Both trends have enhanced the role of fundamental research in automata theory and the importance of closer interaction between theoretical and applied scientists.

The programme is structured into three main parts:

- **Automata models.** Models are specified by four basic components: input, output, storage device and acceptance mode. They include several types of recently introduced models of automata.

- **Fundamental aspects of the theory of automata.** Mathematical foundations, algorithms, languages and formal grammars, symbolic dynamics and coding.

- **Applications.** System analysis and verification, natural language processing, discrete events systems, new challenges for automata theory, etc.
Activities

The programme includes two main activities: organising and supporting scientific events and providing full financial support for short-term visit/exchanges among the programme participants.

A project of Handbook covering the main topics of the programme has also been launched. It will be divided into five main sections:

1. Foundations
2. Complexity issues
3. Algebraic and topological theory of automata
4. Automata in Mathematics
5. Selected Applications

Plenary Conferences and Schools

AutoMathA Plenary Conferences

The programme organised a first plenary international conference in June 2007 in Palermo and a second one in June 2009 in Liège. The main objective of these conferences is to provide a unique forum for interdisciplinary discussions between theoretical computer science, mathematics and more applied communities. Worldwide recognised invited speakers give surveys on the main topics of AutoMathA and the programme committee selects contributed papers. A large number of PhD students attend these conferences. For instance, over 50% of the contributed talks in Palermo were given by PhD students.

In both conferences, the invited lectures gave up-to-date overviews of the topics relevant to AutoMathA. The invited speakers in Palermo were:

- Maxime Crochemore, *Computing Local Periodicities in Strings*.
- Volker Diekert, *Word problems in some inverse monoids*.
- Georg Gottlob, *Monadic Queries over Tree Structured Data*.
- Daniel Kirsten, *Distance Desert Automata and the Star Height Problem*.
- Orna Kupferman, *From formulas to systems*.
- Aldo de Luca, *Pseudopalindromes and generalization of episturmian sequences*.
The Bulletin of the EATCS

- Alexander Okhotin, *Seven families of Language Equations*.
- Dominique Perrin, *Embeddings of automata*.
- Pascal Weil, *On recognizable trace languages*.

and the invited speakers in Liège were:

- Flavio d’Alessandro, *On the Parikh functions of sparse context-free languages*.
- Jean Berstel, *Hopcroft’s automaton minimization algorithm*.
- Alberto Bertoni, *Grammatical compression: compressed equivalence and other problems*.
- Mikołaj Bojańczyk, *Algebra for tree languages*.
- Aldo de Luca, *Stefano Varricchio and his contribution to Theoretical Computer Science*.
- Shlomi Dolev, *Virtual Automata for Mobile Computing and Self Programming Control*.
- Zoltán Ésik, *Algebras of countable words and linear orders*.
- Hendrik Jan Hoogeboom, *Trees and invisible pebbles*.
- Jean-François Raskin, *Exploiting structure in automata constructions*.
- Nicole Schweikardt, *Automata in document processing*.
- Igor Walukiewicz, *Synthesis: words and traces*.

Schools

In September 2008, a two-week school on Algebraic Theory of Automata (SATA 2008) was organised in Lisbon for Ph.D. students and young researchers.

The school brought together 69 participants from 18 different countries: top researchers, both senior and young, as well as post-graduate students interested in this field of research. The participants were exposed to highly interesting research directions, through a series of eight courses and ten advanced seminars, as well as students seminar and problem discussion forum.

The invited lecturers for the eight courses were:
• Mikołaj Bojańczyk, *Tree-walking automata and tree logics*.

• Zoltán Ésik, *Equational Axioms for Fixed Points and Automata*.

• Daniel Kirsten, *Distance Desert Automata and the Star Height Problem*.

• Michal Kunc, *Structure of finite semigroups and language equations*.

• Jean-Éric Pin, *Algebraic theory of automata: an historical perspective of the algebraic approach of automata theory and new advances*.

• Pedro Silva, *Automata in Group Theory*.

• Howard Straubing, *Algebraic Approaches to the Study of Tree Automata*.

• Pascal Tesson, *Computational complexity and Algebraic automata theory*.

More recently, the programme also supported the School on Combinatorics, Automata and Number Theory (CANT’09) that was held in Liège June 2009.

**Workshops**

The programme also supports specialised workshops. Each of these workshops focused on specific topics of the programme and discussed new results as well as perspectives in the field.

In September 2007, a new tool has been launched: AutoMathA Brainstorming and Cooperation Days (ABCD). Those are mini-workshops bringing 6 to 20 researchers to work for 2–3 days on a very specific topic. Rather than just giving a forum for exchanging recent results in a given area, the Brainstorming and Cooperations Days are designed to work out promising research tasks or even projects and to launch cooperations.

Finally, more established events, in which automata play a key role, also received partial support from AutoMathA.

**AutoMathA Specialised Workshops**

So far 12 workshops were supported with 35 participants on the average. The list is given below.

• **Workshop on Semigroups and Automata** (July 2005). Organised by V. Fernandes, G. Gomes, J.-É. Pin, and M. Volkov in Lisbon (Portugal).
Figure 2: Participants of the workshop *Automata and Verification* held in Mons in August 2008.


- **Workshop on Tree Automata** (June 2006). Organised by M. Bojańczyk, C. Löding and S. Tison in Bonn (Germany).


- **Workshop on Automata and Formal Languages for DNA Computation and Bioinformatics** (October 2006). Organised by E. Csuhaj-Varjú, R. Freund, G. Maury and A. Schettini in Como (Italy).

- **Workshop on Symbolic Dynamics and Coding** (July 2007). Organised by M.P. Béal and S. Lombardy in Marne-la-Vallée (France).


- **Automata and Verification** (August 2008). Organised by V. Bruyère and J.-F. Raskin in Mons (Belgium).

Automata in Algorithmic Logic (June 2009). Organised by D. Kuske and M. Lohrey in Stuttgart (Germany).


AutoMathA Brainstorming and Cooperation Days

The list of the 5 AutoMathA Brainstorming and Cooperation Days organised so far is given below.

ABCD on Mu Calculus (May 2008). Organised by J. Duparc in Lausanne (Switzerland).

ABCD Around Cerny’s Conjecture (July 2008). Organised by A. Cherubini, A. Kisielewicz and A. Trahtman in Wroclaw (Poland).


Partially Supported Events

There have been so far 7 established events that received a partial support from AutoMathA. The list of these events is given below.

Weighted Automata Theory and Application (WATA’06) (March 2006). Organised by M. Droste and H.Vogler in Leipzig (Germany).
Figure 3: Participants of the ABCD Automata Theoretic Methods in Algorithmic Algebra held in Bratislava in November 2008.


- **Mons Days of Theoretical Computer Science 2007** (August 2006). Organised by D. Caucal in Rennes (France).

- **GAMES 2007** (September 2007). Organised by J. Duparc in Lausanne (Switzerland).

- **Workshop on Methods for Modalities** (November 2007). Organised by C. Areces and S. Demri in Cachan (France).

- **Mons Days of Theoretical Computer Science 2008** (August 2008). Organised by V. Bruyère and M. Rigo in Mons (Belgium).

- **WORDS 2009** (September 2009). Organised by A. Carpi and C. de Felice in Salerno (Italy).

**Short-term Visit/Exchanges**

A large part of the budget is devoted to short-term fellowships in order to allow research teams in Europe to develop collaborations. Long-term exchanges (typically
Figure 4: Participants of the Workshop on Two-dimensional Languages Theory held in Salerno in May 2006.

more than six weeks) are only supported in exceptional cases. The AutoMathA visit grants have been very successful and there is an increasing number of applications. Young researchers represent a large part of the applications and several new collaborations have been initiated.

So far the programme supported 80 visits among researchers from 20 different countries for a cumulated duration of 170 weeks. This means that, on the average, there was almost one AutoMathA visit per week for the past four years.

How to Apply for a Grant?

All applications should be submitted via the online application forms available on AutoMathA webpage (www.esf.org/automath).

Eligibility for short-term visit/exchange grants are:

1. Undertake work applicable to the programme, that is, related to Automata theory or applications.

2. Apply to stay in a country other than the country of origin.
3. Return to the institute of origin upon termination, so that the applicant’s institution may also benefit from the broadened knowledge of the scientist.

4. Acknowledge ESF in publications resulting from the grantee’s work in relation with the grant.

Financial rules as well as extra details concerning grants for science meetings can be found on the AutoMathA webpage.

**Funding**

ESF scientific programmes are principally financed by the Foundation’s Member Organisations on an à la carte basis. AutoMathA is supported by: Fonds zur Förderung der Wissenschaftlichen Forschung, Austria; Fonds National de la Recherche Scientifique, Belgium; Akademie věd České republiky, Czech Science Foundation, Czech Republic; Forskningsrådet for Natur og Univers, Denmark; Academy of Finland, Finland; Centre National de la Recherche Scientifique, France; Deutsche Forschungsgemeinschaft, Germany; Hungarian Academy of Sciences, Hungarian Scientific Research Fund, Hungary; The Israeli Academy of Sciences and Humanities, Israel; CNR: Ist. di Elettronica e di Ingegneria dell’Informazione e delle Telecomunicazioni, Torino, Ist. di Cibernetica "E. Caianiello", Pozzuoli; Ist. di Scienza e Tecnologia dell’Informazione “A. Faedo”, Pisa, Italy; Polska Akademia Nauk, Poland; Fundação para e Ciência e a Tecnologia, Portugal; Slovak Academy of Sciences, Slovak Republic; Universitat Rovira i Virgili, Tarragona, Spain; The Swiss National Science Foundation for the promotion of scientific research, Switzerland.

**Steering Committee**

- Jean-Éric Pin (Paris, France), chair
- Jorge Almeida (Porto, Portugal)
- Mikołaj Bojańczyk (Warsaw, Poland)
- Véronique Bruyère (Mons, Belgium)
- Stefano Crespi-Reghizzi (Milano, Italy)
- Jacques Duparc (Lausanne, Switzerland)
- Søren Eilers (Copenhagen, Denmark)
• Zoltán Ésik (Szeged, Hungary)
• Jozef Gruska (Brno, Czech Republic)
• Tatiana Jajcayova (Bratislava, Slovak Republic)
• Juhani Karhumaki (Turku, Finland)
• Werner Kuich (Wien, Austria)
• Stuart W. Margolis (Ramat Gan, Israel)
• Carlos Martín Vide (Tarragona, Spain)
• Wolfgang Thomas (Aachen, Germany)

**Getting more informations**

For more information please visit [www.esf.org/automatha](http://www.esf.org/automatha) or send an email to automatha@liafa.jussieu.fr.
European Association for Theoretical Computer Science
EATCS

HISTORY AND ORGANIZATION

EATCS is an international organization founded in 1972. Its aim is to facilitate the exchange of ideas and results among theoretical computer scientists as well as to stimulate cooperation between the theoretical and the practical community in computer science.

Its activities are coordinated by the Council of EATCS, which elects a President, Vice Presidents, and a Treasurer. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual International Colloquium on Automata, Languages and Programming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the “Bulletin of the EATCS;”
- Award of research and academic careers prizes, including the “EATCS Award,” the “Gödel Prize” (with SIGACT) and best papers awards at several top conferences;
- Active involvement in publications generally within theoretical computer science.

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: ETAPS (The European Joint Conferences on Theory and Practice of Software), STACS (Symposium on Theoretical Aspects of Computer Science), MFCS (Mathematical Foundations of Computer Science), LICS (Logic in Computer Science), ESA (European Symposium on Algorithms), Conference on Structure in Complexity Theory, SPAA (Symposium on Parallel Algorithms and Architectures), Workshop on Graph Theoretic Concepts in Computer Science, International Conference on Application and Theory of Petri Nets, International Conference on Database Theory, Workshop on Graph Grammars and their Applications in Computer Science.

Benefits offered by EATCS include:

- Subscription to the “Bulletin of the EATCS;”
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from “EATCS Monographs” and “EATCS Texts;”
- Discount (about 70%) per individual annual subscription to “Theoretical Computer Science;”
- Discount (about 70%) per individual annual subscription to “Fundamenta Informaticae.”

(1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July. Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, data security, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

226
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
- Héraclion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- Wroclaw, Poland 2007
- Reykjavik, Iceland 2008

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for rapid publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D.-Theses;
- Entertainments and pictures related to computer science.

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at http://www.eatcs.org/bulletin. Questions and proposals can be addressed to the Editor by email at bulletin@eatcs.org.

(3) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the EATCS Texts and the EATCS Monographs published by Springer-Verlag and launched during ICALP in 1984. The Springer series include monographs covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as texts intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the series are W. Brauer (Munich), J. Hromkovic (Aachen), G. Rozenberg (Leiden), and A. Salomaa (Turku). Potential authors should contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:
Prof. Dr. G. Rozenberg, LIACS, University of Leiden,  
P.O. Box 9512, 2300 RA Leiden, The Netherlands  
who acknowledges EATCS membership and forwards the order to Springer-Verlag.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

The Editors-in-Chief of the journal currently are G. Ausiello (Rome), D. Sannella (Edinburgh), G. Rozenberg (Leiden), and M.W. Mislove (Tulane).

**ADDITIONAL EATCS INFORMATION**

For further information please visit [http://www.eatcs.org](http://www.eatcs.org), or contact the President of EATCS:  
Prof. Dr. Giorgio Ausiello, Dipartimento di Informatica e Sistemistica  
Universita di Roma “La Sapienza”, Via Salaria 113, 00198 Rome, ITALY  
Email: president@eatcs.org

**EATCS MEMBERSHIP**

**DUES**

The dues are €30 for a period of one year. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for €25 per year. Additional €25 fee is required for ensuring the air mail delivery of the EATCS Bulletin outside Europe.

**HOW TO JOIN EATCS**

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website [www.eatcs.org](http://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](http://www.eatcs.org) to be filled and sent together with the annual dues (or a multiple thereof, if membership for multiple years is required) to the *Treasurer* of EATCS:  
Prof. Dr. Dirk Janssens, University of Antwerp, Dept. of Math. and Computer Science  
Middelheimlaan 1, B-2020 Antwerpen, Belgium  
Email: treasurer@eatcs.org, Tel: +32 3 2653904, Fax: +32 3 2653777

The dues can be paid (in order of preference) by VISA or EUROCARD/MASTERCARD credit card, by cheques, or convertible currency cash. Transfers of larger amounts may be made via the following bank account. Please, add €5 per transfer to cover bank charges, and send the necessary information (reason for the payment, name and address) to the treasurer.

Fortis Bank, Jules Moretuslei 229, B-2610 Wilrijk, Belgium  
Account number: 220–0596350–30–01130  
IBAN code: BE 15 2200 5963 5030, SWIFT code: GEBABE BB 18A