COUNCIL OF THE
EUROPEAN ASSOCIATION FOR
THEORETICAL COMPUTER SCIENCE

**President:** Giorgio Ausiello Italy
**Vice Presidents:**
- Mogens Nielsen Denmark
- Paul Spirakis Greece
**Treasurer:** Dirk Janssens Belgium
**Bulletin Editor:** Vladimir Sassone United Kingdom

<table>
<thead>
<tr>
<th>Name</th>
<th>Country</th>
<th>Institutional Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luca Aceto</td>
<td>Iceland</td>
<td></td>
</tr>
<tr>
<td>Wilfried Brauer</td>
<td>Germany</td>
<td></td>
</tr>
<tr>
<td>Josep Díaz</td>
<td>Spain</td>
<td></td>
</tr>
<tr>
<td>Zoltán Ésik</td>
<td>Hungary</td>
<td></td>
</tr>
<tr>
<td>Giuseppe F. Italiano</td>
<td>Italy</td>
<td></td>
</tr>
<tr>
<td>Jean-Pierre Jouannaud</td>
<td>France</td>
<td></td>
</tr>
<tr>
<td>Juhani Karhumäki</td>
<td>Finland</td>
<td></td>
</tr>
<tr>
<td>Richard E. Ladner</td>
<td>USA</td>
<td></td>
</tr>
<tr>
<td>Jan van Leeuwen</td>
<td>The Netherlands</td>
<td></td>
</tr>
<tr>
<td>Michael Mislove</td>
<td>USA</td>
<td></td>
</tr>
<tr>
<td>Eugenio Moggi</td>
<td>Italy</td>
<td></td>
</tr>
<tr>
<td>Catuscia Palamidessi</td>
<td>France</td>
<td></td>
</tr>
<tr>
<td>David Peleg</td>
<td>Israel</td>
<td></td>
</tr>
<tr>
<td>Branislav Rovan</td>
<td>Slovakia</td>
<td></td>
</tr>
<tr>
<td>Grzegorz Rozenberg</td>
<td>The Netherlands</td>
<td></td>
</tr>
<tr>
<td>Arto Salomaa</td>
<td>Finland</td>
<td></td>
</tr>
<tr>
<td>Don Sannella</td>
<td>United Kingdom</td>
<td></td>
</tr>
<tr>
<td>Jiří Sgall</td>
<td>Czech Republic</td>
<td></td>
</tr>
<tr>
<td>Andrzej Tarlecki</td>
<td>Poland</td>
<td></td>
</tr>
<tr>
<td>Wolfgang Thomas</td>
<td>Germany</td>
<td></td>
</tr>
<tr>
<td>Ingo Wegener</td>
<td>Russia</td>
<td></td>
</tr>
<tr>
<td>Emo Welzl</td>
<td>Switzerland</td>
<td></td>
</tr>
<tr>
<td>Gerhard Wöginger</td>
<td>The Netherlands</td>
<td></td>
</tr>
<tr>
<td>Uri Zwick</td>
<td>Israel</td>
<td></td>
</tr>
</tbody>
</table>

**Past Presidents:**

- Maurice Nivat (1972–1977)
- Arto Salomaa (1979–1985)
- Mike Paterson (1977–1979)
EATCS Council Members

Email Addresses

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

The bulletin is entirely typeset by pdftex and context in txfonts. The Editor is grateful to Uffe H. Engberg, Hans Hagen, Ross Horne, Marloes van der Nat, and Grzegorz Rozenberg for their support.

All contributions are to be sent electronically to bulletin@eatcs.org

and must be prepared in \texttt{\LaTeX} using the class beatcs.cls (a version of the standard \texttt{\LaTeX} \texttt{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 laid 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

EATCS NEWS
THE JAPANESE CHAPTER, by K. Makino ............................... 11
NEWS FROM INDIA, by M. Mukund ................................... 15
NEWS FROM LATIN AMERICA, by A. Viola ............................ 22
NEWS FROM NEW ZEALAND, by C.S. Calude ......................... 25

THE EATCS COLUMNS
THE ALGORITHMS COLUMN, by G. Woeginger
A BIT OF MULTIMEDIA RETRIEVAL ALGORITHMS, by R. Veltkamp .... 29

THE COMPLEXITY COLUMN, by J. Torán
POLYNOMIAL SIZE LOG DEPTH CIRCUITS: BETWEEN NC^1 AND AC^1, by
M. Mahajan ............................................................ 42

THE CONCURRENCY COLUMN, by L. Aceto
CHARACTERISTIC FORMULAE: FROM AUTOMATA TO LOGIC by L. Aceto
and A. Ingolfsdottir .................................................. 57

THE DISTRIBUTED COMPUTING COLUMN, by M. Mavronicolas
OPODIS 2006 REPORT, by B. Englert ................................ 76

THE FORMAL LANGUAGE THEORY COLUMN, by A. Salomaa
NINE OPEN PROBLEMS ON CONJUNCTIVE AND BOOLEAN GRAMMARS, by
A. Okhotin .............................................................. 96

THE FORMAL SPECIFICATION COLUMN, by H. Ehrig
FORMAL MODELING AND ANALYSIS OF FLEXIBLE PROCESSES IN MOBILE
AD-HOC NETWORKS, by J. Padberg, H. Ehrig and K. Hoffmann ........ 120

THE LOGICS IN COMPUTER SCIENCE COLUMN, by Y. Gurevich
ZERO-ONE LAWS: THESAURUS AND PARAMETRIC CONDITIONS by A. Blass
and Y. Gurevich .......................................................... 125

THE NATURAL COMPUTING COLUMN, by G. Rozenberg
SPIKING NEURAL P SYSTEMS: A TUTORIAL, by G. Păun ............... 145

THE PROGRAMMING LANGUAGES COLUMN, by I. Mackie
Web services, mobile processes and types, by K. Honda, N. Yoshida and M. Carbone .......................... 160

TECHNICAL CONTRIBUTIONS

The Freudenthal problem and its ramifications (part II), by A. Born, C. Hurkens, and G. Woeginger ....................... 189
On maximal prefix codes, by L. Staiger .................................. 205
Relabeling and the independence theorem in the double-pushout approach to graph transformations, by H.J. Schneider ............... 208

THE PUZZLE CORNER, by L. Rosaz .......................... 225

MISCELLANEOUS

Peter L. Hammer (1936–2006), by E. Boros, Y. Crama and B. Simeone .................................................. 227
Speculations on biology, information and complexity, by G.J. Chaitin .................................................. 231

REPORTS FROM CONFERENCES

CS&P 2006 ................................................................. 241
NWPT 2006 ................................................................. 243
DCM 2006 ................................................................. 245
McCBIC 2006 .............................................................. 247
ICALP 2006 ................................................................. 249

ABSTRACTS OF PHD THESES ............................... 257

EATCS LEAFLET .......................................................... 265
Letter from the President

Dear EATCS members,

Let me start this message with an important news. As you know this is the time of the year in which the scientist that will receive the EATCS Award is selected. This year the EATCS Award will go to Dana Scott for his outstanding contributions to the theory of programming. The selection Committee was led by Wolfgang Thomas and consisted of Catuscia Palamidessi and David Peleg. The Award will be presented to Dana Scott in a Ceremony that will take place during ICALP in Wroclaw.

The organization of ICALP 2007 is proceeding smoothly. The Program Committees are working hard to select the papers for the three tracks among the hundreds of papers that have been submitted. Although we registered a small decrease in the number of submissions this year the number of good papers is sufficiently high in all tracks to guarantee that the quality of the conference will be excellent as usual. The program will be known after the notification date, April 5. Also the strong names of the invited speakers will make this conference really outstanding (for more information see http://icalp07.ii.uni.wroc.pl/). Also the organization of ICALP 2008 has already started. As you know next year ICALP will be organized in Reykjavik, Iceland, 7-11 July, 2008 by Luca Aceto, Magnús M. Halldórsson and Anna Ingolfsdottir and will again host three tracks (A, B and C, respectively devoted to Algorithms and
Complexity, Logics and Semantics and Security as it has been in the most recent years). The Program Committees of the three tracks will be chaired by Leslie Goldberg, Igor Walukiewicz and Ivan Damgaard, respectively. The definition of the Program Committees is on its way. According to the new Guidelines for ICALP organizers that have been recently redefined by a Committee led by our EATCS-ICALP liaison Pino Italiano, we are doing our best to have the Call for Papers of ICALP 2008 ready at ICALP 2007.

Recently EATCS has also been working toward the aim of re-establishing a fruitful connection with the managers of the European Union research programs in Information and Communication Technology. In this context, an important event in which EATCS has been involved has been a workshop devoted to pervasive adaptive systems. The meeting has been held in Brussels on January 26 and has been jointly organized by EATCS and the Research Unit for Future and Emerging Technologies (FET) with the aim of identifying research issues interesting for the theoretical computer science community within the FET program on Pervasive Adaptation. We hope that the Workshop will be helpful to promote the role of EATCS in European research, also in view of the preparation of future work programs. Those of you who are interested may find the slides of the talks at http://ru1.cti.gr/perad07/. I wish to thank Paul Spirakis for having carefully organized the Workshop and for having contributed to set up the new section of the EATCS web site “News from Brussels” were EATCS members can find information
regarding European research. Of course all information posted in such page only reflects our view of what is going on in European research with no pretense of officiality.

We also thought it might have been useful to our members to access documents on the relevance of Theoretical Computer Science and we added a subsection of the section “Annotated Links” to our web site to provide our members with an easy access to such documents.

This issue of the Bulletin will still be received via ordinary mail by all members but soon we will set up the procedure for allowing the members to choose whether they prefer receiving the printed version by ordinary mail or downloading it from the web.

As usual I wish to thank Vladimiro Sassone for his effort in editing the Bulletin.

Giorgio Ausiello, Rome
February 2007
Letter from the Bulletin Editor

Dear Reader,

Welcome to the February 2007 issue of the Bulletin of the EATCS. We have this time a full house of regular columns (with the exception of the Security Column, which has recently been silent), spanning the customary wide variety of topics. The Natural Computing Column contains a survey on spiking neural P systems, a branch of membrane computing, while Yuri Gurevich presents in his Logic in Computer Science Column a discussion on extending the 0-1 law for first order logic. Kohei Honda et al’s paper in the Programming Languages Column presents a survey of their excellent work in bringing the pi-calculus to the W3C, and Luca Aceto’s Concurrency Column covers the notion of characteristic formula in behavioural logics. Finally, the Algorithmic and the Complexity Columns treat respectively of web search in the multimedia age, beyond text retrieval, and of computing within small complexity classes. I find all this terrifically interesting, and hope you will too.

The volume also includes three refereed contributions, respectively on algorithmics, formal languages and graph transformations, as well as Greg Chaitin’s inspiring Speculations on Biology, Information and Complexity. The pictures from ICALP 2006 that did not make issue 90 are here in the reports section.

In concluding, I sadly remind you that another distinguished member of our community passed away recently. I would like to draw your attention to the obituary that pays tribute to Peter L. Hammer.

Enjoy

Vladimiro Sassone, Southampton
February 2007
INSTITUTIONAL SPONSORS
BRICS, Basic Research in Computer Science,
Aarhus, Denmark

Elsevier Science
Amsterdam, The Netherlands

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

Microsoft Research,
Cambridge, United Kingdom

PWS, Publishing Company,
Boston, USA

TUCS, Turku Center for Computer Science,
Turku, Finland

UNU/IIST, UN University, Int. Inst. for Software Technology,
Macau, China
REPORT FROM THE JAPANESE CHAPTER

K. Makino (Tokyo Univ.)

EATCS-JP/LA Workshop on TCS
As announced in the previous report, the Fifth EATCS-JP/LA Workshop on Theoretical Computer Science will be held at Research Institute of Mathematical Sciences, Kyoto Univ., January 29 ~ 31, 2007. The workshop will be jointly organized with LA, Japanese association of theoretical computer scientists. Its purpose is to give a place for discussing topics on all aspects of theoretical computer science. Please check the program at the end of this report.

On TCS Related Activities in Japan:
TGCOMP Meetings, July ~ December, 2006
The IEICE, Institute for Electronics, Information and Communication Engineers of Japan, has a technical committee called TGCOMP, Technical Group on foundation of COMPUTing. During July ~ December of 2006, TGCOMP organized 3 meetings and about 28 papers (including 3 tutorials) were presented there. Topics presented are, very roughly, classified as follows.

Algorithm: On Graphs (7)  Computational Learning / Knowledge
Algorithm: On Strings (1)  Discovery (1)
Algorithm: On Other Objects (3)  Distributed Computing (4)
Combinatorics / Probabilistic Analysis (5)  Formal Languages and Automata (1)
Computational Complexity and
Semantics and Term Rewriting System (1)

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

EATCS-JP/LA Workshop on TCS Program
NOTICE: EATCS-JP/LA Workshop is an unrefereed meeting; that is, all submissions are accepted for the presentation. Thus there should be no problem of presenting these papers in refereed conferences and/or journals.

Mon, 29th Jan.
2. Learning intersections of regular pattern languages from positive data and its time complexity J. Uemura (Aletheia Univ.)
3. On extensions of alternating context-free grammars E. Moriya (Waseda Univ.), F. Otto, Hartmut Messerschmidt (Univ. Kassel)

4. Neighborhood function for CA H. Nishio (ex. Kyoto Univ.), T. Worsch (The Univ. of Karlsruhe)

5. Variations on neighborhoods in CA T. Worsch (The Univ. of Karlsruhe), H. Nishio (ex. Kyoto Univ.)

6. Tricodes and modified RSA cryptosystems - F. Ding, S. Jimbo, K. Hashiguchi (Okayama Univ.)

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

8. Private Approximation of the Set Cover Problem - M. Yashiro, Keisuke Tanaka (Tokyo Inst. of Tech.)


10. A Lagrangean heuristic algorithm for the R-rooted cycle cover problem - A. Hada (Tohoku Univ.)

11. Approximation Algorithms for Maximum Triangle Packing and Metric Maximum Clustering - Zhi-Zhong Chen, Ruka Takahashi, Tatsuyuki Sekiyama (Tokyo Denki University)


13. Threshold circuits with small energy complexity - K. Uchizawa, E. Takimoto (Tohoku Univ.)

**Tue, 30th Jan.**

14. Orthogonal drawings for plane graphs with specified face areas - A. Kawaguchi, H. Nagamochi (Kyoto Univ.)

15. An efficient algorithm for the MPQ-tree from an interval representation - T. Saitoh, M. Kiyomi, R. Uehara (JAIST)

16. Optimality and algorithms for balancing edge cover problem - Y. Harada, H. Ono, K. Sadakane, M. Yamashita (Kyushu Univ.)

17. On computing longest paths and related problems in ptolemaic graphs - Y. Takahara, S. Teramoto, R. Uehara (JAIST)

18. Quantum asymmetric-key cryptosystems secure against computationally unbounded adversaries - A. Kawachi, C. Portmann (Tokyo Inst. of Tech.y)

19. On the relationships between quantum circuit complexity and the number of controled-NOT gates - S. Okubo, T. Aoki, Y. Kakishita, T. Nishino (The Univ. of Elec.-Comm.)

20. Negation-limited complexity of parity and inverters - H. Morizumi (Kyoto Univ.), J. Tarui (The Univ. of Elec.-Comm.), K. Iwama (Kyoto Univ.)
21. A SUDOKU coloring of a fractal 2-dimensional solid - H. Tsuiki (Kyoto Univ.)
22. Searching a polygon by two boundary 1-searcher - H. Fukami, H. Ono, K. Sadakane, M. Yamashita (Kyushu Univ.)
23. Exact algorithms for the traveling salesman problem on cubic graphs - T. Nakashima, K. Iwama (Kyoto Univ.)
24. A fast algorithm for computing Jones polynomials of montesinos links - M. Murakami (Nihon Univ.), M. Hara (Tokai Univ.), Makoto Yamamoto (Chuo Univ.), Seiichi Tani (Nihon Univ.)
25. Multitoning by the direct binary search - Y. Hirano, K. Nakano (Hiroshima Univ.)
27. Confluence of length preserving string rewriting systems is undecidable - Y. Wang, M. Sakai, N. Nishida, T. Sakabe, K. Kusakari (Nagoya Univ.)
28. On theorem-proving for higher-order TRSs using proof-by-consistency - H. Okamura, M. Oyamaguchi, T. Yamada (Mie Univ.)
29. Synthesis of scheduler of soft real-time systems based on value function - S. Kano, S. Yamane (Kanazawa Univ.)
30. Automatic verification of real-time programs based on predicate abstraction and its refinement - R. Komagata, S. Yamane (Kanazawa Univ.)
31. A consideration for approximation estimation for minimum spanning tree cost on graphs with randomly distributed edge weights - E. Ando, H. Ono, M. Yamashita (Kyushu Univ.)
32. An approximation algorithm for task scheduling of DAGs of height not exceeding four - K. Shimizu, M. Oyamaguchi, T. Yamada, T. Yanagimoto, K. Komatsu (Mie Univ.)
33. Online learning of linear threshold functions with large biases - K. Ishibashi, K. Hatano, M. Takeda (Kyushu Univ.)
34. A lower bound for the path distance width of the complete binary tree - K. Ukegawa, K. Aoki, K. Kozawa, Y. Otachi, K. Yamazaki (Gunma Univ.)
35. Branchlength and treelength - K. Umezawa, N. Nakahiro, K. Yamazaki (Gunma Univ.)

**Wed, 31st Feb.**

36. Approximation algorithm for constructing phylogenetic tree using bottom-up method - K. Maemura, H. Ono, K. Sadakane, M. Yamashita (Kyushu Univ.)

38. Simple compressed data structure for fast decoding substrings - T. Goto, H. Ono, K. Sadakane, M. Yamashita (Kyushu Univ.)

39. Local search based algorithm for DNA sequence design - S. Kawasimo, H. Ono, K. Sadakane, M. Yamashita (Kyushu Univ.)

40. Confluence of left-linear term rewriting systems - M. Iwami (Shimane Univ.)

41. Data power-saving and reliable data transmission on sensor network - K. Sanagi, M. Yamashita, K. Sadakane, H. Ono (Kyushu Univ.)

42. Applying TrustRank on P2P networks - K. Yonezawa, Y. Tanaka (Hokkaido Univ.)

43. A failure detector to solve k-set agreement - A. Sakata, H. Ono, K. Sadakane, M. Yamashita (Kyushu Univ.)

44. An efficient algorithm for trust negotiation strategy - Y. Yamamoto, Y. Takata, H. Seki (Nara Inst. of Sci. and Tech.)

---

THE JAPANESE CHAPTER

Chair: Kazuo Iwama
V. Chair: Osamu Watanabe
Secretary: Kazuhisa Makino
Email: eatcs-jp@is.titech.ac.jp
URL: http://www.is.titech.ac.jp/~watanabe/eatcs-jp
In this edition of News from India, we carry reports on various meetings between September, 2006 and January, 2007.

**SEFM 2006.** The 4th IEEE International Conference on Software Engineering and Formal Methods (SEFM) was organized in Pune, India from 11th–15th September 2006. This is the first time that this conference was held in India. The conference chair was Mathai Joseph from TRDDC, Pune, while the program committee chairs were Paritosh Pandya (TIFR, Mumbai, India) and Dang Van Hung (UNU-IIST, Macao, China). The conference website is at [http://www.iist.unu.edu/SEFM06](http://www.iist.unu.edu/SEFM06).

SEFM 2006 featured four excellent invited talks. Joseph Sifakis (Verimag, Grenoble) talked about modeling heterogeneous real-time components in the BIP language as super-positions of behaviour, interactions and priorities. In his talk titled “The Context of Object Computation”, Bertrand Meyer (ETH, Zurich) spoke about how contract specifications and proof obligations can be used in object-oriented programming to formalize software development. Sriram Rajamani (Microsoft Research, Bangalore) gave an overview of the state of the art and a glimpse of the future of software validation that combines techniques from the historically disparate fields of software testing and property validation based on static analysis. John Rushby (SRI, Menlo Park, CA) presented some approaches to improve the state of the art in theorem proving by harnessing innovations and progress in other fields such as SAT solving, predicate abstraction and model checking.

SEFM 2006 also featured two days of tutorials prior to the main conference dealing with topics such as retrenchment, static analysis of programs, software verification and formal methods.
Over 100 papers were submitted to the conference, out of which 24 papers were selected for presentation at the conference. The papers covered diverse areas of software engineering and formal methods such as formalizing Java, formal modeling, requirements modeling, program analysis, and software testing. The conference was attended by 85 participants from 12 countries—a record for participation in SEFM!

The event was sponsored by Computer Society of India, Honeywell, Indian Association for Research in Computing Science (IARCS), Microsoft Research, Persistent Software Private Ltd, Tata Consultancy Services and UNU-IIST.

**Indocrypt 2006.** The 7th International Conference on Cryptology in India, Indocrypt2006, was held in Kolkata during December 11–13, 2006. It was organized by the Cryptology Research Society of India. The general chair was Bimal Roy (ISI Kolkata) and the program co-chairs were Rana Barua (ISI, Kolkata) and Tanja Lange (TU of Denmark and ETU). Two invited lectures were presented at Indocrypt; one by James L. Massey entitled “Whither Cryptography” and the other by Alfred Menezes entitled “Another Look at ‘Provable Security’” (joint work with Neil Koblitz).

In the pre-conference tutorials held on December 10, D.J. Bernstein of the University of Chicago at Illinois spoke on “High Speed Diffie-Hellman” and Palash Sarkar of ISI, Kolkata spoke on “Generic Attacks on Symmetric Ciphers”.

186 papers were submitted of which 29 were accepted after peer reviews. There were about 140 participants of which more than 40 were from outside India. The proceedings of the conference are published by Springer as LNCS 4329. The tutorials and the invited talks are available on the conference website http://www.isical.ac.in/~indocrypt.

**FSTTCS 2006.** The 26th edition of FSTTCS, the annual conference of IARCS, took place in Kolkata during December 13–15, 2006. The conference was organized by the Indian Statistical Institute, Kolkata. The Program Committee was co-chaired by S. Arun-Kumar and Naveen Garg from IIT, Delhi. A total of 34 papers were accepted from over 150 submissions. The conference included two satellite workshops, one on Timed Systems before FSTTCS and the other on Approximation Algorithms on December 16 after the main conference.

There were five invited speakers for FSTTCS 2006. The first invited talk, by Gordon Plotkin (Edinburgh, UK), was entitled “Hennessy-Plotkin-Brookes Revisited”. On his first visit to India, Plotkin gave an engaging talk on constructing a fully abstract semantics for concurrent programs. Emo Welzl (ETH Zurich, Switzerland) gave a very entertaining talk on “The Number of Crossing Free Configurations on Finite Point Sets”. He developed the intricate combinatorics...
involved in this problem in a very systematic way. The talk was pitched at just
the right level of detail so that he could carry the audience with him all the way.
The third invited talk was by Gérard Boudol (INRIA, Sophia Antipolis, France)
on “Shared-Variable Concurrency: A Proposal”. Boudol put forward a persuasive
case for a new model based on cooperative preemption to describe concurrent
threads. In a talk intriguingly entitled “All That Noise!”, Eugene Asarin (LIAFA,
Paris 7, France) set out with the plausible premise that many undecidable prob-
lems in timed and hybrid systems may become tractable if measurements were
made imprecise to account for ambient noise. Unfortunately, he finally demon-
strated that this hope was false and that, even in the presence of noise, these prob-
lems remain undecidable. The final invited talk was by David Shmoys (Cornell,
USA). In a wonderfully clear presentation on “Approximate Algorithms for 2-
stage Stochastic Optimization”, he illustrated why he is such a popular lecturer
with students.

A unique feature of FSTTCS 2006 was the general strike (or bandh) called by
trade unions on the second day of the conference. With the Left Front ruling West
Bengal, the state of which Kolkata is the capital, the bandh was, in local parlance,
“totally successful and peaceful”. This meant that all shops were closed and there
was no public transport available on December 14. The conference programme
was reorganized so that all talks were squeezed into two hectic days, with a “rest
day” in between. Fortunately, pedestrians were not harassed and many visitors got
a rare chance to explore Kolkata on foot, unhindered by the usual chaotic traffic.
A few participants were even heard suggesting that a rest day be built in as an
integral part of future FSTTCS conferences!

FSTTCS 2006 received sponsorship from the host institution, ISI Kolkata,
as well as from IBM India Research Laboratory, Delhi and Microsoft Research,
Bangalore.

As usual, the annual business meeting of IARCS was held during FSTTCS.
It was announced that FSTTCS 2007 would be held in New Delhi, with the pro-
gramme committee jointly chaired by V. Arvind (IMSc, Chennai) and Sanjiva
Prasad (IIT Delhi). The venue for FSTTCS 2008 was fixed to be IISc, Bangalore.

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

Workshop on Timed Systems. A pre-FSTTCS workshop on recent advances
and issues in timed systems was organized on December 11 and 12 at Kolkata.
The speakers at the workshop were mostly drawn from those with papers at the
conference, including a strong French contingent from Paris and Cachan. Eugene
Asarin and Paul Gastin spoke about language-theoretic characterisations for timed
languages, in particular using signal-event models. Patricia Bouyer spoke about
decidability issues in finding optimal strategies in weighted timed games. Among
the speakers from India, Paritosh Pandya gave an interesting talk on discretizing real-time logics, while Madhavan Mukund introduced a notion of timed Message Sequence Charts and a textual notation for describing their hierarchical version. Martin Leucker enlightened us on using Angluin’s learning model to learn Event Recording Automata. Finally Ashish Tiwari from SRI gave an interesting talk on using abstractions for analysing hybrid systems.

The workshop was preceded by a day of preparatory lectures, the highlight of which was an excellent presentation by Fabrice Chevalier on decidability issues for different versions of Metric Temporal Logic.

**Workshop on Approximation Algorithms.** A workshop on “Recent Advances in Approximation Algorithms” was held in Kolkata after FSTTCS on December 16, 2006. It was organized by Anupam Gupta (CMU) and Amit Kumar (IIT Delhi).

David Shmoys (Cornell University) talked about algorithms developed for solving stochastic inventory control models. Sudipto Guha (University of Pennsylvania) presented new models of optimization under partial information. Venkat Guruswami (University of Washington Seattle) presented new methods for proving lower bounds for learning with noise. Nisheeth Vishnoi (IBM India Research Lab) talked about recent techniques developed for proving upper and lower bounds on the approximation ratio of graph partitioning problems. Yuval Rabani (Technion) talked about new techniques for the k-means clustering problem. Alberto Marchetti Spaccamela (Universita di Roma “La Sapienza”) gave a talk on new algorithmic problems arising out of data aggregation in sensor networks. Kunal Talwar (Microsoft Research) talked about new results for the approximate degree bounded spanning tree problem. Fabrizio Grandoni (Universita di Roma “La Sapienza”) gave a talk on new results on the approximation ratio of the connected facility location problem.

**ISAAC 2006.** The Seventeenth International Symposium on Algorithms and Computation (ISAAC 2006) took place at Hotel Taj Bengal, Kolkata during December 18–20, 2006. This international conference on Algorithms and Computation takes place annually in the Asia-Pacific region. This is the second ISAAC meeting organized in India; the first one was organized in 1999 at Chennai.

The program committee was chaired by Tetsuo Asano (Japan Advanced Institute of Science and Technology, Japan). Out of 225 submissions, 73 papers were selected for presentation. The proceedings of the conference are published by Springer as LNCS 4288. Two special issues, one of Algorithmica and one of the International Journal of Computational Geometry and Applications, will contain selected papers from the proceedings of ISAAC 2006.
The Bulletin of the EATCS

The symposium consisted of 2 invited lecture sessions, one best paper presentation session, a special session from IBM and 18 technical paper presentation sessions. The invited lectures were delivered by Kazuo Iwama (Kyoto University, Japan) and Tamal K. Dey (Ohio State University, USA). The topics were Stable Matching Problem and Delaunay Meshing of Surfaces, respectively. The best paper award was given for the paper titled “Algorithmic graph minor theory: improved grid minor bounds and Wagner’s contraction” by Erik Demaine, Mohammad Taghi Hajiaghayi and Ken-ichi Kawarabayashi. The best student paper award was given for the paper titled “Branching and treewidth based exact algorithms” by Serge Gaspers, Fedor Fomin and Saket Saurabh. The total number of participants of ISAAC 2006 was approximately 150; among them 77 were from outside India.

The symposium was organized by the Indian Statistical Institute, Kolkata in cooperation with the Dept. of Computer Science and Engineering, IIT Kharagpur and IARCS. ISAAC received financial support from several agencies under the Government of India and the Government of West Bengal. The symposium also received financial support from some non-government institutions, namely Capgemini Consulting India Private Limited, Tata Consultancy Services, IBM India Private Limited, Cognizant Technology Solutions and Anshin Software for sponsoring various events.

The advisory committee of ISAAC met on the evening of 18 December. It was decided that ISAAC 2007 will be held at Sendai, Japan and ISAAC 2008 will be held at Sydney, Australia.

Workshop on Algorithms for Data Streams. This workshop was held in the Department of Computer Science at the Indian Institute of Technology Kanpur from December 18–20, 2006. The scope of the workshop was to explore cutting-edge research in data streaming and other closely related models of computation. In these models, data arrives at a rapid and continuous rate, and computation has to be performed typically in a single-pass over the data using significantly sub-linear resources (time and space). The objective is often to maintain a small amount of useful statistics from the voluminous data. In related emerging models, multiple-passes over the data are traded for more resource efficient computation. These models find large scale utility in network monitoring, financial applications, database applications, etc.

The workshop was very fortunate to have lectures from leading researchers in the world in this area, including, Pankaj Agarwal (Duke), Amit Chakrabarti (Dartmouth), Graham Cormode and Divesh Srivastava (AT&T Research), Sudipto Guha, Sampath Kannan and Andrew McGregor (U. Pennsylvania), Piotr Indyk (MIT), Ravi Kannan (Yale), Ravi Kumar and Michael Mahoney (Yahoo), Yossi
Matias (Google and Tel Aviv U.), S. Muthukrishnan and D. Sivakumar (Google), Nicole Schweikardt (U. Humboldt-Berlin), Rajeev Raman (U. Leicester), Christian Sohler (U. of Paderborn), Martin Strauss (U. Michigan), Subhash Suri (UC Santa Barbara) and Srikanta Tirthapura (U. Iowa). Besides the speakers, there were 54 delegates (not counting the students and faculty at IIT Kanpur), comprising both faculty and students from different colleges in India and a few international delegates from around the world.

The talks at the workshop were divided into the following themes. The roots: algorithms for classical data streams, Lower bounds, Linear Algebra Computations in small space, Algorithms for geometry problems, Algorithms for streaming graphs, Compressed Sensing, Data Streaming Systems, Machine learning and data streams. The workshop was very stimulating, with plenty of questions, discussions, open problems, hypotheses, etc. As classical problems, such as finding frequent items and estimating Lp norms, entropy, etc. seem better understood, the focus appears to shift towards more fundamental questions and towards newer areas (linear algebra, combinatorial optimization, compressed sensing).

The slides and the talks (in mpeg format) can be downloaded from the workshop site www.cse.iitk.ac.in/users/sganguly/workshop.html.

TECS Week 2007. The 5th TCS Excellence in Computer Science Week (TECS Week 2007) was held at TRDDC, Pune from 3–7 January 2007. It was conducted by Tata Research Development and Design Centre (TRDDC) jointly with United Nations University (UNU/IIST) and IARCS. Each TECS Week is an advanced workshop on a relevant topic related to computer science and software engineering, and aims at providing high-quality computer science education to students, teachers and practitioners from developing countries. TECS Week 2007 was on Data-intensive Computing.

The speakers at TECS Week ’07 were Jim Kurose (Univ. of Massachusetts, Amherst, USA), S. Muthukrishnan (Rutgers University, USA and Google), Raghu Ramakrishnan (Univ. of Wisconsin, Madison, USA and Yahoo!), and Alex Smola (National ICT, Australia).

Jim Kurose talked about CASA (Collaborative Adaptive Sensing of the Atmosphere), a case-study of a data-driven sense-and-response system. He elaborated on problems and solutions in collecting, analyzing and presenting data in large sensor networks with a specific application to meteorological sensing.

S. Muthukrishnan talked about algorithms for analyzing massive streams of data. He provided insight into data stream models, sublinear space/time algorithms, compressed sensing and massive distributed systems. He illustrated the algorithms with several interesting examples in IP network traffic analysis, web traffic analysis and signal processing.
Raghu Ramakrishnan talked about the recent trends in databases with a special emphasis on data mining and exploratory analysis. He covered a variety of topics such as DBMS support for complex data analysis (OLAP, warehousing, view materialization), database support for exploratory data mining, and web data management.

Alex Smola gave a detailed picture of active topics in machine learning such as Support Vector Machines (SVM) and other kernel-based learning methods. He provided an overview of support vector classification and regression, novelty detection, and quantile regression. He introduced several kernel-based methods and illustrated them with applications to named entity recognition and ranking of web pages for search engines.

TECS Week 2007 received an enthusiastic response with over 200 applications from all over the world. 60 selected candidates from India and neighboring countries attended the event.

For more details, please visit http://www.tcs-trddc.com/tecs/.

Acknowledgments. Ashok Sreenivas <ashok.sreenivas@gmail.com> contributed the writeup on SEFM 2006. The report on Indocrypt2006 was from Rana Barua <rana@isical.ac.in>. Subhas Nandy <nandysc@isical.ac.in> provided the report on ISAAC 2006. Deepak D’Souza <deepakd@csa.iisc.ernet.in> wrote about the Timed Systems workshop while Amit Kumar <amitk@cse.iitd.ac.in> contributed the report on the Approximation Algorithms workshop. Sumit Ganguly <sganguly@iitk.ac.in> sent in a report on the Data Streaming workshop. The report on TECS Week 2007 was from Niranjan Pedanekar <niranjan.pedanekar@tcs.com>.

Madhavan Mukund, Chennai Mathematical Institute
Secretary, IARCS (Indian Association for Research in Computing Science)
http://www.cmi.ac.in/~madhavan
In this issue I present the Call for Papers of ADHOCNOW 2007, the report of CLAIO 2006 by J. M. Martinez and the 2nd Call for Proposals for STIC-AmSud. At the end I present a list of the main events in Theoretical Computer Science to be held in Latin America in the following months.

**Call for Papers of ADHOCNOW 2007.**

Ad-Hoc networks are wireless, self-organizing systems formed by co-operating nodes within communication range of each other that form temporary networks. Their topology is dynamic, decentralized, ever changing and the nodes may move around arbitrarily. The last few years have witnessed a wealth of research ideas on ad hoc networking that are moving rapidly into implemented standards. Following previous Ad-Hoc Networks and Wireless conferences in Ottawa, Canada (2006), Cancun, Mexico (2005), Vancouver, Canada (2004), Montreal, Canada (2003), and Toronto, Canada (2002), the sixth International Conference on Ad-Hoc Networks and Wireless will take place in Morelia, Mexico, from September 24-26, 2007.

We request research papers in any of the following areas: Access Control, Ad Hoc Networks of Autonomous Intelligent Systems, Algorithmic Issues, Analytic Methods and Modeling for Performance Evaluation, Application for Ad
The Bulletin of the EATCS


The conference proceedings will be published by Springer-Verlag as Lecture Notes in Computer Science (LNCS). Papers should have a length of up to 14 pages following the Springer LNCS format; for more information contact the Springer website at http://www.springeronline.com/lncs/. Abstract submissions are due in March 26, 2007 and paper submissions are due in April 2, 2007. The Web page of the event is http://www.scs.carleton.ca/~adhocnow/2007/.


The XIII Latin-Ibero-American Congress on Operations Research (CLAIO 2006) has taken place in Montevideo, Uruguay, on November 27-30 2006. Héctor Cancela chaired the Program Committee and Maria Urquhart chaired the Organizing Committee. Together with CLAIO, the first ALIO/INFORMS Workshop on Operations Research dedicated to Education was held. CLAIO is the most traditional Operations Research meeting in the region and takes place every two years since 1982. Montevideo, the Capital of Uruguay, a charming 250 years old city, is one of the only continental capitals with a ring of beaches incorporated into the urban landscape. The city maintains historic testimonies both in its old and new sectors. The meeting included ten plenary talks by Martin Groetschel, Carlos Coello, Suresh Sethi, James Cochran, Michel Gendreau, Pierre L’Ecuyer, José Mario Martinez, Monique Guignard, Julian Araoz and Gerardo Rubino and 377 contributed presentations in all modern areas of Operations Research. Contributions came from 23 different countries, 9 of them out from the Latin-Ibero-American area. Communications were organized in seven parallel sessions in the NH-Columbia Hotel. The physical environment was quite adequate, with time enough for moving from one presentation to another and a confortable common area for informal discussions. Plenary talks were given in Club Uruguay, three blocks away from NH-Columbia. The participation of graduate students was quite impressive, both in terms of their presentations as in terms of their contribution to discussion. The Organizing Commitee was very successful in their choice of the social program. This included an instructive and relaxing visit to wineyards followed by an excellent barbecue and dancing. Moreover, Héctor, Marita and their collaborators did an outstanding work in making our visit to Uruguay quite enjoyable.
STIC-AmSud, 2nd Call for Proposals.

STIC-AmSud is a program of the French - South America cooperation, intending to promote cooperation and networking in research in Science, Information Technology and Communications in South America. The 2nd call for projects proposals is now open, and the deadline for the submission of proposals is March 31, 2007. Research and innovation projects in Information and Communications Sciences and Technologies are sought. Proposals should include at least two regional partners (Argentina, Brasil - Sao Paulo, Chile, Peru and Uruguay) and one French partner. All relevant information is available in http://www.sticamsud.org.

Regional Events

1 Scientific and Community News

The New Zealand Chapter of ACM (ACM_NZ) operates exclusively for educational and scientific purposes within New Zealand. It “aims to cater the needs of its membership by not only providing a common platform for exchange of ideas but also as a resource base fulfilling regional needs.” ACM_NZ webpage is http://is-alt.massey.ac.nz/acmnz/. ACM_NZ produces the Association for Computing Machinery New Zealand Bulletin, ISSN 1176-9998, http://is-alt.massey.ac.nz/acmnz/bulletin.php; it contains a wealth of information about various IT/CS-related activities in NZ.

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


284. C.S. Calude and M.A. Stay. Most Short Programs Halt Quickly or Never Halt. 08/2006
286. U. Speidel. IT-Complexity and T-Information Theory—an Executive Sum-
mary, 2nd revised version. 10/2006
287. G. Pritchard and M.C. Wilson. Probability Calculations Under the IAC
Hypothesis. 10/2006
Diameter of Random Cayley Digraphs of Given Degree. 10/2006
289. G. Firror, T. Mansour and M.C. Wilson. Longest Alternating Subsequences
in Pattern-Restricted Permutations. 10/2006
11/2006
293. C.S. Calude and M.J. Dinneen. Exact Approximations of Omega Numbers.
12/2006
01/2007
THE EATCS COLUMNS
A Bit of Multimedia Retrieval Algorithmics

Remco C. Veltkamp
Department of Information and Computing Sciences
Utrecht University
Remco.Veltkamp@cs.uu.nl

Abstract
After text retrieval, the next waves in web searching and multimedia retrieval are the search for and delivery of images, music, video, and 3D scenes. Not only the perceptual and cognitive aspects, but also many of the algorithmic and performance aspects are still badly understood. One relevant issue is the design of dissimilarity measures (distance functions) that have desired properties. Another aspect is the development of algorithms that can compute or approximate these distances efficiently. Indexing data structures and search algorithms are necessary to make the search more efficient than sequential browsing through large collections.

1 Introduction
Multimedia research has been going on since the nineteen-sixties, even if it was not called like that. A key aspect of multimedia research is interfacing: establish-
ing a seamless interaction and communication between the user and the computer. In that respect it represents an important ingredient of current developments which are denoted by buzz phrases such as ubiquitous computing, ambient intelligence, context awareness, the disappearing computer, video at your fingertips, anything, anyone, anywhere, anytime. Multimedia retrieval is essential for coping with the problems of information overload, in production and content management, and in personalized usage. Indeed, the reason that email and web search engines have become so immensely popular are precisely that they cope with these issues with respect to text. However, if perceptually relevant multimedia methods, that guarantee performance, are not invented soon, there is no hope that similar problems are effectively solved with respect to images, music, video, and 3D models.

Since the first pictorial information systems in the early nineteen-eighties, research has come a long way in developing various methods to handle visual information by its content, as opposed to processing by keywords [8]. However, these content descriptions consist of low level color, texture, and shape features [12], and they often miss perceptual relevance. The methods for extracting and comparing these features are primarily heuristic, which, although they are clever themselves, miss guaranteed properties. In contrast, an algorithmic approach is focused on provable properties, see section 1.1.

Looking at a particular multimedia framework as in figure 1, we see that those processes that are of an algorithmic nature are the extraction of features from the multimedia documents, the matching of the query features with the database features, the construction of the indexing data structure to speed up the searching, and the visualization of the resulting retrieved multimedia documents.

The big challenge in multimedia for the next years is the processing of information in a way that is perceptually and semantically relevant. Because of the need for personalized information access and searching, processing should be done in a manner that is guaranteed effective. Because the searching and filtering is performed on very large databases of multimedia information, it must be done with guaranteed efficiency also. The holy grail is not yet within reach.

What makes this difficult is the gap between the high level semantic information and the low level features of current multimedia systems. For example, if one is looking for an image of the holy grail on the basis of image content features, one may query for a chalice shape and a star shape. However, simple edge detection yields a set of unconnected lines, not a star. Therefore, low level features will fail miserably for this purpose.

A concrete listing of research issues in multimedia is the following. Firstly, in order to arrive at semantic access, a necessary step is the identification of what is perceptually and cognitively important in the multimedia documents.

Secondly, in order to cope with the data and information overload, it is becoming essential that effective and efficient searching techniques are developed.
Indeed, not only company archives contain huge amounts of media. The success of mobile phone with sms (short message service) shows that as soon as consumer groups adopt devices like mobile phones with built-in digital cameras and mms (media message service) via broad band communication like GPRS or UMTS, massive amounts of images and video are produced and stored. Digital music and movies is already causing a very large amount of Internet traffic. The so-called fourth wave in multimedia (after images, video and music), consisting of 3D models and scenes, is showing more and more on the web. Together these media form an enormous amount of data, and it is essential to provide tools to match and filter, and to retrieve personalized information from it.

Thirdly, to make retrieval feasible from such large quantities, efficient searching methods must be invented. In particular, indexing data structures and algorithms must be designed that avoid the need to scan whole collections from front to back, but instead refer the user in a few steps to the right place in the collection.

Fourthly, any successful fully-fledged system needs to provide a combination of image, video, music, and 3D model handling with text capabilities. The integrated system engineering is far from trivial, and challenging in itself.

The algorithmic aspects of these items form an area of research that I have coined multimedia algorithmics [13].
1.1 Multimedia Algorithmics

Like all computer systems, all multimedia systems are built on algorithms. Any system in modern information and communication technology is an algorithmic system. When it comes to the design of algorithms for multimedia, this involves for example algorithms for extracting and grouping perceptually relevant patterns, computing the similarity, indexing and searching in large collections, and visualizing retrieval results in a way that is meaningful for relevance feedback. Research issues are the invention of new algorithms that solve problems in an efficient way, guaranteeing provable properties in a rigorous way, taking an axiomatic approach, basing derivations on first principles.

The above-stated aspects are combined into a line of research that is rooted in the discipline of fundamental algorithm design, and applied to the domain of multimedia: multimedia algorithmics. The gap between the high level semantic information and the low level features of current multimedia systems makes it difficult to make a significant step forward. A challenging research agenda for the next years is to invent algorithms for multimedia along the following orthogonal axes:

1. The tasks in a typical multimedia framework that are of an algorithmic nature: perceptual feature extraction, pattern matching, indexing, and visualization.
2. The different media (images, music, video, 3D models and scenes) to which these tasks are applied.
3. The desired properties of algorithms that must be invented: robustness, invariance, and efficiency, etc.

Together, these aspects span a whole research space, as illustrated in figure 2.

This paper discusses some aspects from the field of multimedia algorithmics, and is largely based on [14]. In the next sections we will discuss an algorithm for a particular type of pattern matching, and an algorithm for indexing over large collections to speed up the retrieval.

2 Shape Matching

This section is about one of the aspects mentioned above, matching of patterns. We look at shape matching, in particular matching of multiple polylines to a single polygon.

There is evidence that, for the task of object recognition, the human visual system uses a part-based representation. Biederman [1], for example, suggested that objects are segmented at regions of deep concavity into an arrangement of simple geometric components. For the retrieval of polygonal shapes, we have
therefore developed an algorithm to search for the best matching polygon, given one or more query parts. This dissimilarity measure models partial matching, is translation and rotation invariant, and deformation robust.

Let $P_1$ be a polyline, and let $P_1(s)$ be the point on $P_1$ at distance $s$ along the polyline from its beginning. The turning-angle function $\Theta_1$ of a polyline $P_1$ measures the angle of the counterclockwise tangent at $P_1(s)$ with respect to a reference orientation as a function of $s$. It is a piecewise constant function, with jumps corresponding to the vertices of $P_1$. The domain of the function is $[0, \ell_1]$, where $\ell_1$ is the length of $P_1$. Rotating $P_1$ by an angle $\theta$ corresponds to shifting $\Theta_1$ over a distance $\theta$ in the vertical direction.

The turning-angle function $\Theta_P$ of a polygon $P$ is defined in the same way, except that the distance $s$ is measured by going counterclockwise around the polygon from an arbitrarily chosen reference point. Since $P$ is a closed polyline, we can keep going around the polygon, and the domain of $\Theta_P$ can thus be extended to the entire real line, where $\Theta_P(s + \ell_P) = \Theta_P(s) + 2\pi$. Moving the location of the reference point over a distance $s$ along the boundary of $P$ corresponds to shifting $\Theta_P$ horizontally over a distance $s$.

To measure the mismatch between $P_1$ and the part of $P$ starting at $P(t)$, we align $P_1(0)$ with $P(t)$ by shifting the turning-angle function of $P$ over a distance $t$ and computing the $L_2$-distance between the two turning-angle functions, minimized over all possible rotations $\theta$ (that is: vertical shiftings of the turning functions). The squared mismatch between $P_1$ and $P$, as a function of $t$, is thus given by:

$$d_1(t) := \min_{\theta \in \mathbb{R}} \int_0^{\ell_1} (\Theta_P(s + t) - \Theta_1(s) + \theta)^2 ds.$$  \hspace{1cm} (1)
An ordered set of $k$ polylines $\{P_1, P_2, \ldots, P_k\}$ can be represented by concatenating the turning-angle functions of the individual polylines. Thus we get a function $\Theta_{PL} : [0, \ell_1] \rightarrow \mathbb{R}$, where $\ell_j$ is the cumulative length of polylines $P_j$ through $P_j$. For $1 \leq j \leq k$ and $\ell_{j-1} \leq s \leq \ell_j$ we have $\Theta_{PL}(s) := \Theta_j(s - \ell_{j-1})$, so that each polyline $P_j$ is represented by the section of $\Theta_{PL}$ on the domain $[\ell_{j-1}, \ell_j]$. The squared mismatch between $P_j$ and $P$ (shifted by $t$) is now given by:

$$d_j(t) := \min_{0 \leq \theta < 2\pi} \int_{\ell_{j-1}}^{\ell_j} (\Theta_p(s + t) - \Theta_{PL}(s) + \theta)^2 ds.$$  

(2)

We now express the mismatch between the set of polylines $\{P_1, P_2, \ldots, P_k\}$ and $P$ as the square root of the sum of squared mismatches between each polyline and $P$, minimized over all valid shiftings:

$$d(P_1, \ldots, P_k; P) := \min_{\text{valid shiftings } t_1 \ldots t_k} \left( \sum_{j=1}^{k} d_j(t_j) \right)^{1/2}. \quad (3)$$

It remains to define what the valid shiftings are. To keep the polylines disjoint (except possibly at their endpoints) and in counterclockwise order around the polygon, each polyline has to be shifted at least as far as the previous one, that is: $t_{j-1} \leq t_j$ for all $1 < j \leq k$. Furthermore, to make sure that $P_2$ does not wrap around the polygon beyond the starting point of $P_1$, we have to require that $t_k + t_k \leq t_1 + \ell_p$ (see figure 3).

2.1 Algorithm

For $j \in \{2, \ldots, k\}$, $1 \leq a \leq |X_j|$, and $b \in \{a, \ldots, |X_j|\}$, with $x_b \leq x_a + \ell_p - \ell_j$, we define:

$$D[j, b](a) := \min_{t_2, \ldots, t_j \in \{x_b, \ldots, x_a\}} d_j(x_a) + \sum_{b=2}^{j} d_b(t_b). \quad (4)$$

It can be shown that there is an optimal solution to the optimization problem in equation (3) such that $t_j \in X$ for $j \in \{1, \ldots, k\}$ and $t_1 \in X_1$. Thus, we get $d(P_1, \ldots, P_k, P) = \min_{a \in \{1, \ldots, |X_1|\}} D[k, b](a)$, where $b(a)$ is the maximum $b$ such that $x_b \leq x_a + \ell_p - \ell_j$. We now show that an optimal solution of this problem can be constructed recursively. Let $(t_2, \ldots, t_j)$ be an optimal solution for $D[j, b](a)$. Regarding the value of $t_j$ we distinguish two cases:

- $t_j = x_b$, in which case $(t_1, \ldots, t_{j-1})$ must be an optimal solution for $D[j - 1, b](a)$, otherwise $(t_1, \ldots, t_j)$ would not give a minimum for $D[j, b](a)$; thus in this case, $D[j, b](a) = D[j - 1, b](a) + d_j(x_b)$;
Figure 3: To match polylines $P_1, \ldots, P_3$ to polygon $P$, we shift the turning functions of the polylines over the turning function of the polygon. To maintain the order of the polylines around the polygon, we need to guarantee $t_1 \leq t_2 \leq t_3$ and $\ell_3 + t_3 \leq t_1 + \ell_P$.

- $t_j < x_b$, in which case $(t_1, \ldots, t_j)$ must be an optimal solution for $D[j, b - 1](a)$, otherwise $(t_1, \ldots, t_j)$ would not give a minimum for $D[j, b](a)$; thus in this case, $D[j, b](a) = D[j, b - 1](a)$.

We now conclude for $j \in \{2, ..., k\}$, $1 \leq a \leq |X_1|$, and $b \in \{a, ..., |X|\}$, with $x_b \leq x_a + \ell_P - \ell_j$:

$$D[j, b](a) = \min \left( D[j - 1, b](a) + d_j(x_b), D[j, b - 1](a) \right),$$

(5)

where the boundary cases are $D[1, b](a)$, which is $d_1(x_a)$ for all $b$ and all $1 \leq a \leq b$. Note that $D[j, b](a)$ has no solution when $b < a$. We represent that by defining $D[j, b](a) = \infty$ in that case.

Equation 5 leads to a straightforward dynamic programming algorithm SimpleCompute for computing the similarity measure $d(P_1, \ldots, P_k; P)$, see figure 4. In [9] it is shown that this algorithm takes $O(km^2n^2)$ time and $O(kmn)$ storage. Furthermore, it gives a novel algorithm that runs in $O(kmn \log(mn))$ time and space.

3 Indexing

Proximity searching in multimedia databases has gained more and more interest over the years. In particular searching in dissimilarity spaces (rather than extract-
Algorithm SimpleCompute d(P₁, . . . , Pₖ; P)
1: Compute the sorted set of canonical shiftings \( X = \{x₁, \ldots, x_{|X|}\} \)
2: For all \( j \in \{1, \ldots, k\} \) and all \( b \in \{1, \ldots, |X|\} \), evaluate and store \( d_j(x_b) \)
3: \( \text{minimumMismatch} \leftarrow \infty \)
4: for \( \text{startingPoint} \leftarrow 1 \) to \( |X| \) do
5: \( b \leftarrow \text{startingPoint} \)
6: for \( j \leftarrow 2 \) to \( k \) do
7: \( D[j, b - 1] \leftarrow \infty \)
8: while \( x_b + \ell_k \leq x_{\text{startingPoint}} + \ell_P \) do
9: \( D[1, b] \leftarrow d_1(x_{\text{startingPoint}}) \)
10: for \( j \leftarrow 2 \) to \( k \) do
11: \( D[j, b] \leftarrow \min \{D[j - 1, b] + d_j(x_b), D[j, b - 1]\} \)
12: \( b \leftarrow b + 1 \)
13: \( \text{minimumMismatch} \leftarrow \min(D[k, b - 1], \text{minimumMismatch}) \)
14: return \( \sqrt{\text{minimumMismatch}} \)

Figure 4: Dynamic programming algorithm for optimal placement.

ing a feature vector for each database object) is an increasing area of research. With growing multimedia databases indexing has become a necessity.

Vantage indexing works as follows: given a multimedia database \( A \) and a distance measure \( d : A \times A \rightarrow \mathbb{R} \), select from the database a set of \( m \) objects \( A^* = \{A^*_1, \ldots, A^*_m\} \), the so called vantage objects. Compute the distance from each database object \( A_i \) to each vantage object, thus creating a point \( p_i = (x_1, \ldots, x_m) \), such that \( x_j = d(A_i, A^*_j) \). Each database object corresponds to a point in the \( m \)-dimensional vantage space.

A query on the database now translates to a range-search or a nearest-neighbor search in this \( m \)-dimensional vantage space: compute the distance from the query object \( q \) to each vantage object (i.e. position \( q \) in the vantage space) and retrieve all objects within a certain range around \( q \) (in the case of a range query), or retrieve the \( k \) nearest neighbors to \( q \) (in case of a nearest neighbor query). The distance measure used on the points in vantage space is \( L_\infty \).

Vleugels and VelTkamp show [15] that as long as the triangle inequality holds for the distance measure \( d \) defined on the database objects, recall (ratio of number of relevant retrieved objects to the total number of relevant objects in the whole database) is 100%, meaning that there are no false negatives. However, false positives are not excluded from the querying results, so precision (ratio of number of relevant retrieved objects to the total number of retrieved objects) is not necessarily 100%. We claim that by choosing the right vantage objects, precision can increase significantly.

The retrieval performance of a vantage index can improve significantly with a
The Bulletin of the EATCS

proper choice of vantage objects. This improvement is measured in terms of false positives, as defined below. Let $\delta$ be the distance measure in vantage space.

**Definition 1. Return set** Given $\epsilon > 0$ and query $A_q$, object $A_i$ is included in the return set of $A_q$ if and only if $\delta(A_q, A_i) \leq \epsilon$.

**Definition 2. False positive** $A_p$ is a false positive for query $A_q$ if $\delta(A_q, A_p) \leq \epsilon$ and $d(A_q, A_p) > \epsilon$.

We present a new technique for selecting vantage objects that is based on two criteria which address the number of false positives in the retrieval results directly. The first criterion (spacing) concerns the relevance of a single vantage object, the second criterion (correlation) deals with the redundancy of a vantage object with respect to the other vantage objects. We call this method Spacing-based Selection.

The main idea is to keep the number of objects that are returned for a query $A_q$ and range $\epsilon$ low. Since false negatives are not possible under the condition that the triangle inequality holds for $d$, minimization of the number of false positives is achieved by spreading out the database along the vantage space as much as possible. False positives are, intuitively speaking, pushed out of the returned sets.

**Spacing** In this section we will define a criterion for the relevance of a single vantage object $V_j$. A priori the query object $A_q$ is unknown, so the distance $d(A_q, V_j)$ between a certain query $A_q$ and vantage object $V_j$ is unknown. The size of the range query ($\epsilon$) is unknown beforehand as well. Optimal performance (achieved by small return sets given a query $A_q$ and range $\epsilon$) should therefore be scored over all possible queries and all possible ranges $\epsilon$.

This is achieved by avoiding clusters on the vantage axis belonging to $V_j$. Our first criterion therefore concerns the spacing between objects on a single vantage axis, which is defined as follows:

**Definition 3.** The spacing between two consecutive objects $A_i$ and $A_{i+1}$ on the vantage axis of $V_j$ is $d(A_{i+1}, V_j) - d(A_i, V_j)$.

Let $\mu$ be the average spacing. Then the variance of spacing is given by formula $\frac{1}{n-1} \sum_{i=1}^{n-1} ((d(A_{i+1}, V_j) - d(A_i, V_j)) - \mu)^2$. To ensure that the database objects are evenly spread in vantage space, the variance of spacing has to be as small as possible. A vantage object with a small variance of spacing has a high discriminative power over the database, and is said to be a relevant vantage object.

**Correlation** It is not sufficient to just select relevant vantage objects, they also should be non-redundant. A low variance of spacing does not guarantee that the database is well spread out in vantage space, since the vantage axes might be strongly correlated.
Algorithm SpacingBasedSelection

Input: Database A with objects $A_1, ..., A_n$, $d(A, A) \rightarrow \mathbb{R}$, thresholds $\epsilon_{corr}$ and $\epsilon_{spac}$

Output: Vantage Index with Vantage objects $V_1, V_2, ..., V_m$

1: select initial $V_1, V_2, ..., V_m$ randomly
2: for All objects $A_i$ do in random order
3: for All objects $V_j$ do
4: compute $d(A_i, V_j)$
5: add $A_i$ to index
6: if var(Spacing)$(V_j) > \epsilon_{spac}$ then
7: remove $V_j$
8: select new vantage object randomly
9: if for any pair $p(V_k, V_l)$, Corr$(V_k, V_l) > \epsilon_{corr}$ then
10: remove $p$’s worst spaced object
11: select new vantage object randomly

Figure 5: Spacing-based Selection.

Therefore, we compute all linear correlation coefficients for all pairs of vantage objects and make sure these coefficients do not exceed a certain threshold. Experiments show that on the MPEG-7 shape images set pairwise correlation is sufficient and that higher order correlations are not an issue.

3.1 Algorithm

Spacing-based Selection selects a set of vantage objects according to the criteria defined above with a randomized incremental algorithm. The key idea is to add the database objects one by one to the index while inspecting the variance of spacing and correlation properties of the vantage objects after each object has been added. As soon as either the variance of spacing of one object or the correlation of a pair of objects exceeds a certain threshold, a vantage object is replaced by a randomly chosen new vantage object. These repair steps are typically necessary only at early stages of execution of the algorithm, thus keeping the amount of work that has to be redone small. For details, see the algorithm in figure 5.

The complexity of our algorithm is expressed in terms of distance calculations, since these are by far the most expensive part of the process. The running time complexity is then $O(\sum_{i=0}^{n} P_i \times i + (1 - P_i) \times k)$ where $k$ is the (in our case constant) number of vantage objects and $P_i$ is the chance that, at iteration $i$, a vantage object has to be replaced by a new one. This chance depends on the choice for $\epsilon_{spac}$ and $\epsilon_{corr}$. There is a clear trade-off here: the stricter these threshold values are, the better the selected vantage objects will perform but also the higher the chance a
vantage object has to be replaced, resulting in a longer running time. If we only look at spacing and set \( \epsilon_{\text{spac}} \) such that, for instance, \( P_i \) is \((\log n)/i\), the running time would be \( O(n \log n) \) since \( k \) is a small constant (8 in our experiments).

3.2 Experimental Evaluation

We implemented our algorithm and tested it on the MPEG-7 test set CE-Shape-1 part B, and the distance measure used to calculate the distance between two of these shape images is the Curvature Scale Space (CSS), [5]. To justify our criteria, we manually selected four sets of eight vantage objects that either satisfy both criteria (weakest correlation and lowest variance of spacing: weak-low), none (strongest correlation and highest variance of spacing: strong-high) or a strong-low or weak-high combination.

The performance of these four sets of vantage objects was evaluated by querying with all 1400 objects. The number of nearest neighbors that was retrieved for each query object varied from 1 to 200. The distance of the furthest nearest neighbor functioned as \( \epsilon \), which was used to calculate the number of false positives among these nearest neighbors, see Definition 2. For each vantage index, and all \( k \)-NN queries, \( k = 1, \ldots, 200 \), an average ratio of false positives in result was calculated over all 1400 queries. The results are displayed in figure 6, together with some typical runs of our algorithm, the “MaxMin” approach [15] and the “loss-based” approach [4].

These results show that both criteria need to be satisfied in order to achieve good performance (only the set called weak-low scores less than 50% false positives for all sizes of nearest neighbor query). Furthermore, it shows that our algorithm can actually select a set of vantage objects in which these criteria are satisfied, since false positive ratios are low for these sets. For more details, see [11].

4 Concluding Remarks

This paper primarily shows multimedia algorithms in the domain of image retrieval, but we have taken a similar approach to music retrieval. As a dissimilarity measure we have designed the Proportional Transportation Distance [3], a normalized version of the Earth Mover’s Distance [7]. It satisfies the triangle inequality, which makes it suitable for indexing with the vantage method. Indeed, we have used it in combination with the vantage indexing method in our music retrieval systems Muugle (http://give-lab.cs.uu.nl/muugle) [2] and Orpheus (http://give-lab.cs.uu.nl/orpheus/). The vantage indexing made it possible to identify anonymous incipits (beginnings of pieces, for example
Figure 6: False positive ratios of different algorithms.

twenty notes long) from the RISM A/II collection [6] consisting of about 480,000
incipits [10]. All 80,000 anonymous incipits were compared to the remaining
400,000 ones, giving a total of 32,000,000,000 comparisons. Should a single
comparison take 1 ms, this would have taken about 370 days. The vantage index-
ing made it possible to do this within a day on a 1 GHz PC. A total of 17,895
incipits were identified.

Acknowledgment

I want to thank all persons I have worked with on multimedia retrieval, and with whom
the results reported here are obtained. In particular I thank Martijn Bosma, Panos Gi-
nnopoulos, Herman Haverkort, Reinier van Leuken, Mirela Tanase, Rainer Typke, and
Frans Wiering. This research was supported by the FP6 IST projects 511572-2 PROFI
and 506766 AIM@SHAPE.

References


experimentation framework. In *Proceedings of the 9th International Conferencen on

The Bulletin of the EATCS


Meena Mahajan gives in these pages a personal view to the landscape of “small” complexity classes, reviewing from a modern perspective the fundamental results related to the problems that can be computed using logarithmic depth circuits.

**Polynomial size log depth circuits:**
**Between NC\(^1\) and AC\(^1\)**

Meena Mahajan

---

**1 Introduction**

When a theoretical computer scientist asks me my area of research, I usually say complexity theory. This is often followed by the question “what kind of complexity theory” to which I inevitably reply “inside \( P \)”. And usually the questioning stops there. In this brief survey, I would like to go further, and describe some of my favourite complexity classes. They all lie in the range between \( \text{NC}^1 \) and \( \text{AC}^1 \); hence this title. I cannot even begin to attempt being exhaustive, and I apologize.

---

*The Institute of Mathematical Sciences, CIT Campus, Chennai 600041, India.*
in advance to those whose favourite results I have omitted. Much of this material (and much more!) can be found in the text [42] and the surveys [1, 23].

2 Principal classes between NC$^1$ and AC$^1$

Consider (uniform) families of polynomial size log depth circuits with internal AND and OR gates, and literals / constants at the leaves. (No internal negations, without loss of generality). Restricting the gates to have constant fanin gives the complexity class NC$^1$; leaving it unrestricted (limited, of course, by the circuit size itself) gives AC$^1$.

Without loss of generality, we can assume that our circuits are layered: gates appear in layers, and wires connect adjacent layers in one direction. The maximum number of gates at any one layer of the circuit is called the width of the circuit.

Several well-known classes are sandwiched in between; let’s take a look at each of these.

![Figure 1: The landscape between NC$^1$ and AC$^1$](image)

2.1 NC$^1$

At the lower end, we begin with the class NC$^1$. NC$^1$ has many equivalent characterizations. It equals the class of languages accepted by

- alternating Turing machines in logarithmic time ALOGTIME (under appropriate uniformity conditions). ([34])
- poly size programs over finite monoids BP-M.

A program over a monoid $M = (S, ◦)$ is a list of instructions of the form $(i, a, b)$ where $i \in [n]$, $a, b \in S$. The instruction $(i, a, b)$, on input $x \in \{0, 1\}^n$, evaluates to $a$ if $x_i = 1$, to $b$ otherwise. The entire program, on input $x \in$
[0, 1]^n, thus constructs a word \( w \in S^* \). The program accepts \( x \) if \( w \) evaluates to a designated value in \( M \).

A simple divide-and-conquer approach establishes that programs over finite monoids can be evaluated in \( \text{NC}^1 \). Barrington [4] established the converse by showing that over any non-solvable group, words can be constructed to code logical AND and negation. In particular, he used the permutation group \( S_5 \); thus programs over this group are complete for \( \text{NC}^1 \).

- **bounded-width poly size branching programs BWBP.**
  These are constant-width layered graphs (one for each input length) with designated start and finish vertices \( s, t \), and edges labelled by literals or constants. An input is accepted if the corresponding graph has an \( st \) path where all edges are labelled by 1 or true literals.

  It is folklore (and easy to see) that programs over monoids can be described in this form, and vice versa.

  Such programs are also equivalent to skew circuits, where OR gates are unrestricted but AND gates can have at most one input that is not a literal or a constant.

- **bounded-width poly size circuits \( \text{SC}^0 \).**
  \( \text{SC} \) is the class of polynomial size poly logarithmic width circuits (width \( O(\log^i n) \) for \( \text{SCI}^i \)). (Again, wlog the circuits can be assumed to have negations only at the leaves.) In simulating a circuit by a Turing machine, width roughly translates to space and size to time; thus \( \text{SC} \) corresponds in the uniform setting to a simultaneous time-space bound. (\( \text{SC} \) stands for Steve’s Classes, named after Stephen Cook who proved the first non-trivial result about polynomial time log-squared space \( \text{PLoSS} \), i.e. \( \text{SC}^2 \), in [11]. See for instance [22]). As described above, Barrington’s result places \( \text{NC}^1 \) inside skew \( \text{SC}^0 \). But even non-skew \( \text{SC}^0 \) is easily seen to be inside \( \text{NC}^1 \), since only a constant amount of memory is needed to evaluate the gates of the circuit layer by layer. Thus \( \text{SC}^0 \) equals \( \text{NC}^1 \).

- **poly size formulae \( F \), even when restricted to log-width formula \( \text{LWF} \).**
  A formula is a circuit where each gate has fanout at most 1. \( \text{NC}^1 \) circuits can be converted to formulae by duplication; the blow-up in size is still within a polynomial. Conversely, any formula can be restructured into an equivalent one with polynomial blow-up in size and logarithmic depth; a non-uniform way to do this was first described in [7, 35], while it can be done uniformly as in [28]. Further, any log depth formula can be restructured to log width (at the expense of depth, of course), as observed in [20]; thus \( \text{LWF} = F = \text{NC}^1 \).
• predicates expressed in first-order logic, augmented with a group quantifier $Q_G$ over any non-solvable group $G$, $\text{FO}[Q_G]$. [28].

2.2 AC$^1$

At the higher end, we have AC$^1$. Less is known about AC$^1$; it equals the class of languages accepted by

• alternating Turing machines, using logarithmic space and making at most logarithmic alternations between universal and existential states, on any computation path $\text{ASP,ALT}(\log,\log)$ (under appropriate uniformity conditions);

• concurrent read, concurrent write PRAMs working in logarithmic time with polynomially many processors.

Now consider the intermediate classes.

2.3 DLOG

DLOG is the class of languages accepted by deterministic logspace machines. It also equals the class of languages accepted by log width poly size circuits $\text{SC}^1$. DLOG equals sentences expressible in FO augmented with deterministic transitive closure $\text{FO}[\text{DTC}]$, [18], and it follows from [32] that DLOG also equals FO + symmetric transitive closure $\text{FO}[\text{STC}]$.

2.4 NLOG

NLOG is the class of languages accepted by nondeterministic logspace machines. The inductive counting technique of Immerman and Szelepcsényi [19, 37] shows that NLOG is closed under complementation. An equivalent formulations of NLOG is the class of languages accepted by uniform poly size skew circuits (or branching programs); see [39]. In descriptive complexity, NLOG is characterized by sentences in first-order logic with positive transitive closure $\text{FO}[$pos TC$]$, see [18].

2.5 LogCFL

LogCFL, by definition, is the class of languages reducible via logspace many-one reductions to some context-free language. It follows that each such language can be accepted by a machine which has logspace to perform the reduction, and a nondeterministic finite control and a stack to then parse the CFL in polynomial
time. Such machines are called AuxPDA(poly), and Sudborough showed that they accept exactly LogCFL ([36]). That is, allowing arbitrary interleaving of the two types of computation involved – (1) deterministic logspace reduction, and (2) nondeterministic PDA – is no more powerful than performing these two phases sequentially. (An aside: the polynomial time restriction is necessary, since Cook [10] showed that in unbounded time, and even in exponential time, deterministic PDA augmented with logspace worktape capture all of P.)

Using the notion of realizable pairs of surface configurations, Ruzzo showed [33] that AuxPDA(poly) can be simulated by alternating TMs using logspace and having poly-sized proof trees. What is a proof-tree? Consider the computation graph of a logspace-bounded ATM, where nodes are time-stamped configurations. (The logspace bound ensures a poly-sized graph; the time-stamping ensures that the graph is acyclic.) To prove that it accepts its input, it suffices to show a subgraph that contains (1) the initial configuration, (2) both children of each universal node included, (3) at least one child of each existential node included, and (4) only accepting configurations as leaves. Such a subgraph, unfolded or expanded out by duplicating nodes if necessary so that it is a tree, is what we call a proof-tree. It is easy to see that poly-sized graphs can have exponential-sized proof-trees. Ruzzo’s proof shows that to describe the computations of AuxPDA(poly), poly-sized proof trees suffice. Conversely, if a logspace-bounded ATM has, for each accepted input, a proof-tree of size at most $t(n)$, then an AuxPDA can accept the same language in time $t(n)$. Thus we have a characterization of LogCFL via ATMs: LogCFL = ASP;TRSZ(log,poly). Note that the above proof-tree definition can be applied to circuits as well. Using a very nice tree-cutting argument, Venkateswaran showed [38] that a poly-sized circuit of any depth, but with a poly-size bound on its proof trees, can be flattened to log depth, at the cost of increasing the fanin of OR gates. This is the circuit class SAC$^1$, semi-unbounded alternating circuits. The converse simulation is direct, giving ASP;TRSZ(log,poly) = SAC$^1$.

An interesting offshoot of Venkateswaran’s construction is that each AuxPDA(poly) can be simulated by an AuxPDA(poly) whose stack height never grows beyond $O(\log^2 n)$. (Only $O(\log n)$ pairs of surface configurations, each needing $O(\log n)$ bits, need to be stacked.)

More recently, in [27], McKenzie, Rienhardt and Vinay gave a direct proof that ASP;TRSZ(log,poly) is in LogCFL, thus eliminating the need for the elaborate construction of Sudborough.

The class of all CFLs is not closed under complementation. Nonetheless, one could expect that a logspace reduction closure captures complements as well, and indeed this is the case. Interestingly, none of the above forms directly show that LogCFL is closed under complementation. The SAC$^1$ formulation was used by Borodin et al [6] to apply inductive counting and thus establish this closure. This closure captures a certain symmetry between the OR and AND operators: as long
as one of them has bounded arity, we are within LogCFL.

Bedard, Lemieux and McKenzie gave yet another characterization of LogCFL in [5]. Generalising the programs-over-monoids framework of Barrington, they show that LogCFL equals languages accepted by programs over groupoids. These are algebraic structures where a non-associative binary operator $*$ on a set $A$ is defined. Given a word $w \in A^*$, consider all possible ways of parenthesising it to apply $*$. These different ways yield a set of possible values $S(w)$. Acceptance is defined in terms of $S(w)$ containing some designated element, or equalling some designated set. By imposing syntactic conditions on programs over groupoids, NC$^1$, DLOG and NLOG can also be captured in this framework [5, 25].

The framework of [5] directly leads to a logical characterization as well: LogCFL is exactly those languages whose membership is expressible in first-order logic augmented by groupoidal quantifiers. A more detailed treatment of this characterization can be found in [24].

2.6 LogDCFL

LogDCFL is the class of languages reducible via logspace many-one reductions to some deterministic context-free language. As in the case of LogCFL, the two computation phases in deciding membership in a LogDCFL language can be interleaved [36]; thus LogDCFL equals DAuxPDA(poly). It is also characterized in the PRAM model: it is the restriction of AC$^1$ to concurrent read owner-write (CROW) PRAMs, see [15, 16]. One of the most non-trivial properties about LogDCFL is that it is contained in SC$^2$; this was shown by Cook in [11]. No subclass of NC containing LogDCFL is known to be inside SC, though a possibly incomparable chunk of NC consisting of randomized (bounded two-sided error) logspace is also known to be in SC [30]. Surprisingly, we do not yet know how to combine these two constructions to place randomized poly time AuxPDA inside SC.

2.7 A formal language view

For many reasons, AC$^1$ is not as interesting formally as the classes within it. The main reason is to do with proof-tree size: AC$^1$ circuits can have exponentially large proof trees. This crucially impacts arithmetic versions of these circuits; we will come to that shortly. Another is that there is no neat characterization of AC$^1$ via formal language classes. From the formal-language-theoretic point of view, we have the following containment diagram:

All the containments are proper, and DCFL and Lin are incomparable. Applying very weak closures to these classes – uniform FO projections – gives exactly the complexity classes of Figure 1. Notice that AC$^1$ is not covered here. The jump
from CFLs to context-sensitive languages is too big; closure of CSLs gives all of PSPACE. We need something much smaller to capture exactly AC$^1$.

2.8 Completeness

Here is a partial list of problems complete for each of these classes:

**LogCFL** BlockChoice(Dyck-2), the hardest CFL: Given a sequence of blocks, each containing a list of strings, can we pick exactly one string from each block so that their concatenation, in that order, is in Dyck-2 (the language of balanced parentheses with two types of parentheses)?

- Non-zero Tame Tensor Formula [13]: Given a tensor formula satisfying a certain “tameness” property, determine whether it is non-zero.
- Semi-extended regular expression membership [31]: Given an expression $r$ over some alphabet $\Sigma$ where $r$ is like a regular expression but is also allowed to use $\cap$, and given a string $x \in \Sigma^*$, determine whether $x \in L(r)$.

**NLOG** Reachability in a directed acyclic graph.

- 2-CNF-SAT.
- Regular expression membership [21]: Given a regular expression $r$ over some alphabet $\Sigma$ and a string $x \in \Sigma^*$, determine whether $x \in L(r)$.

**LogDCFL** BlockChoice(Dyck-2), the hardest DCFL: let Dyck-2 be over $\{a, b, c, d\}$ with $a$ and $c$ opening, and matched by $b$ and $d$ respectively. Given a string $x_0 \in (a+c)^*$, and a sequence of blocks $B_1, \ldots, B_k$ each consisting of one string in $b(a+c)^*$ and one in $d(a+c)^*$, can we pick exactly one string from each block so that their concatenation, with $x_0$, is in Dyck-2?

**DLOG** Reachability in an undirected graph, presented by its adjacency lists. [32]

- Remains hard even if the graph is a two-tree forest. [12]
- Bipartiteness: given an undirected graph, determine if it is bipartite.

**NC$^1$** Reachability in a bounded-width layered graph.

- The Boolean Formula Value problem.
The Bulletin of the EATCS

The word problem over the group $S_5$ (for that matter, over any finite non-solvable monoid).

Fixed Regular expression membership: For a fixed regular expression $r$ over some alphabet $\Sigma$, given a string $x \in \Sigma^*$, determine whether $x \in L(r)$.

3 Lesser-known classes

By varying parameters appropriately between NC$^1$ and LogCFL, we get some lesser-known classes in this range:

3.1 Syntactic restrictions

- BP-width: Within polysize, constant-width BPs gives NC$^1$ and unbounded width BPs give NLOG. One could thus consider width $w(n)$ BPs, for $w$ a function of $n$. Vinay showed [41] that for each polylog $w \in O(\log n')$, the corresponding class is closed under complement. But nothing much more is known. For instance, even the smallest class here, log-width BPs, lying between NC$^1$ and DLOG, is not known to capture any natural problem in this range.

- OR fanin: Within poly size log depth circuits with constant AND fanin, varying OR fanin from constant to polynomial takes us from NC$^1$ to SAC$^1$ (i.e. LogCFL). What about OR fanin $f(n)$ where $f$ is, say, polylog? Again, Vinay showed closure under complement, [41]. Also, just as SAC$^1$ contains NLOG (at $f = \text{poly}$), each of these classes contains the corresponding BP-width-constrained class described above. But do they capture any natural problems?

- Circuit-width: Constraining circuit width alone to polylog gives the SC hierarchy, and very little of the NC hierarchy is known to lie within it. However, the defining property separating LogCFL from P is poly size proof trees (also referred to as poly degree). One could combine a width restriction with a degree restriction to obtain a sub-hierarchy of SC within LogCFL. Limaye et al [26] define what they call small SC denoted sSC: its $i$th level has poly size poly degree $O(\log^i)$ width circuits. Again, each level here contains the corresponding width-constrained BPs, though no relationship with the constrained-OR-fanin circuits is known. Though these classes are not yet known to be closed under complement, [26] shows that co-sSC$^0$ is in sSC$^0$. At the smallest level, sSC$^0$ equals SC$^0$, but it is not known whether sSC$^1$ is as powerful as SC$^1$. 


3.2 Language/Automata-theoretic constructs

- Let us take a closer look at Figures 1, 2. $NC^1$ equals the closure of regular languages. Yet some non-trivial non-regular CFL families are included in it. These include parenthesis languages [8], visibly pushdown languages VPLs [3, 14], linear CFLs with an LL[1] condition [17]. (Imposing an LR[1] condition is what corresponds to determinism. Thus CFLs with an LR[1] condition equal DCFLs, linear CFLs with an LR[1] condition equal languages accepted by 1-turn DPDA, usually referred to as deterministic linear languages.)

Let me highlight the membership in $NC^1$ of VPLs. Firstly, what are VPLs? These are languages accepted by visibly pushdown automata VPAs. So what are VPAs? These are PDAs with no $\varepsilon$ moves, where the stack movement (push / no change / pop) is dictated solely by the input letter being read. They are clearly stronger than NFAs (they can accept $a^n b^n$: push on $a$, pop on $b$), but also weaker than PDAs (they cannot accept $a^n b a^n$: is $a$ a push letter or a pop letter?). In [3], it was shown that VPAs can be determinized; thus VPLs are in DCFLs. But well before this was known, these languages had been studied under the name input-driven languages. Dymond gave a nice construction [14] showing that they are in fact in $NC^1$. His approach is generic and works not just for VPAs but for any PDA satisfying the following:

1. no $\varepsilon$ moves,
2. an accepting run should end with an empty stack,
3. the height of the pushdown, after processing $i$ letters of the input, should be computable in $NC^1$. If there is more than one run (non-deterministic PDA), and if the height profiles across different runs are different, then the heights computed should be consistent with a single run. Furthermore, if there is an accepting run, then the heights computed should be consistent with some accepting run.

For such PDA, Dymond transforms the problem of recognition to an instance of formula value problem, and then invokes Buss’s ALogTime algorithm [8] for it.

VPAs satisfy these conditions (with appropriate padding to satisfy condition (2)). But much more can be achieved via condition (3). The height profiles of all runs in a VPA are the same, and can be computed in TC0. Understanding exactly what can be placed inside $NC^1$ by carefully using Dymond’s proof is a nice question.
An interesting proper generalization of VPAs are what Caucal introduced in [9] and calls synchronized PDA. Languages accepted by these are contained in DPDA but incomparable with DLin. Does their closure create a new class between NC\(^1\) and LogDCFL, or does it collapse to one of these or even to DLOG?

- The fact that the logspace closure of Lin is NLOG is interesting. The machine model for Linear CFLs is PDA which, on each run, make at most 1 turn on the stack. That is, no stack symbols are pushed after the first pop move. Thus the machine model for the logspace closure of Lin is AuxPDA(poly) making 1-turn in stack movement. This suggests a fine gradation between NLOG and LogCFL parameterized by the number of turns the AuxPDA is allowed to make. A similar gradation arises between DLOG and LogDCFL by considering the deterministic counterpart.

3.3 Counting constructs

- Unambiguity: Between DLOG and NLOG lies, quite naturally, unambiguous logspace ULOG. Similarly, LogUCFL lies between LogDCFL and LogCFL. Interestingly, the correspondence between the formal language class and the complexity class is not known to hold here: the logspace closure of unambiguous CFLs viz. LogUCFL, is contained in unambiguous logspace machines UAuxPDA(poly), but the converse is not known, and similarly for unambiguous Linear CFLs. There are also close relationships in the PRAM model: while LogDCFL is characterized by log time CROW PRAMs, LogUCFL is contained in by log time CREW PRAMs, which correspond to a strong form of unambiguity in AC\(^1\) circuits. There are several subtleties in the definition of unambiguous machines/circuits: is there at most one accepting path, or at most one path from the initial to any configuration, or at most one path between any pair of configurations? For more about these nuances, see [23, 29].

- Randomization: Between ULOG and NLOG lies one-sided-error randomized logspace RLOG: either none, or overwhelmingly many, accepting runs. Nisan showed [30] that RLOG (and even its two-sided-error version) is contained in SC\(^2\). No class above RLOG is known to be in SC. A natural containment to expect, since LogDCFL is also in SC\(^2\), is that randomized LogCFL, RLogCFL, is also in SC\(^2\). So far this has not been shown to hold. But another interesting set of questions here is to do with the appropriate definition of RLogCFL itself. LogCFL has multiple characterizations, each of which can be randomized to give competing definitions for RLogCFL:
1. a randomized logspace reduction to some CFL
2. a randomized AuxPDA(poly) with bounded error
3. a randomized AuxPDA(poly) with stack-height bounded by $O(\log^2 n)$ and bounded error
4. an SAC\(^1\) circuit with polynomially many supplementary random input bits, and bounded error

Which of these truly reflects RLogCFL?

4 Arithmetization

In this section, I will briefly discuss arithmetizations of these classes over integers. There are two standard ways to arithmetize a circuit class: (1) assuming there are negations only at the leaves, replace AND and OR gates by $\times$ and $+$ gates respectively, or (2) count the number of proof-trees. Both give the same class of functions. Equivalent models can be appropriately arithmetized: For programs over monoids, consider an NFA corresponding to the monoid, view the instructions as projections transforming an input, and count the number of accepting paths of the NFA on the transformed version. For branching programs, count the number of $st$ paths. For LogCFL, count the number of parse trees in the target CFL. For NLOG and AuxPDA(poly), count the number of accepting paths. And so on.

The arithmetization of AC\(^1\) is not very interesting. Within log depth, a circuit can, starting with 0s and 1s, compute numbers that need exponentially many bits in their binary representation. This is because they can have exponentially large proof trees. For feasible computation, we may be justified in restricting attention to poly size arithmetic circuits that compute numbers with feasible representation. This corresponds to poly size circuits with poly degree, and over integers, is essentially the same as Valiant’s class VP. It also corresponds directly to an arithmetization of one characterization of LogCFL, namely, ASPTRSZ(log,poly).

Interestingly, all arithmetizations of LogCFL coincide: poly size poly degree arith circuits, poly size log depth #SAC\(^1\), number of accepting paths in AuxPDA(poly) machines, number of parse trees in a CFL, number of good parenthesizations of a word over a groupoid. Venkateswaran’s tree-cutting construction [38] placing LogCFL in SAC\(^1\) is not parsimonious; it does not give a one-to-one correspondence between accepting paths of the AuxPDA(poly) machine and proof trees of the SAC\(^1\) circuit. To establish the equivalence of #AuxPDA(poly) and #SAC\(^1\), two independent and different constructions were described by [40] and [29] (see also [2]). These can also be thought of as tree-cutting, but the cuts are applied more carefully to uniquely halve the degree in a constant number of
stages. Since these techniques apply to the Boolean case as well, we have three different proofs that LogCFL is in SAC$^1$. Since unbounded addition and bounded multiplication are both in NC$^1$, it is easy to see that #SAC$^1$ is in Boolean NC$^2$.

Over NLOG too, the two arithmetizations coincide: number of accepting paths in an NLOG machine, and number of proof trees in a poly size skew circuit, both give the function class #L.

For LogDCFL and DLOG, it is not clear how to define an arithmetic version. One possibility is to consider functional versions FLOG and FLogDCFL. But this is not entirely satisfactory because in this kind of framework, we expect arithmetization to yield more power. Another possibility for DLOG is to consider #SC$^1$, since DLOG equals SC$^1$. But #SC$^1$ can compute infeasible values, so this is an unreasonable class. Yet another possibility is to consider poly degree SC$^1$ circuits, #sSC$^1$. But it is not even known whether sSC$^1$ is as powerful as SC$^1$, so we may be restricting ourselves too much this way. An interesting spin on #sSC$^1$ is that it is contained in both #SAC$^1$ and Boolean SC$^2$. Thus, inverting the question of “How much of NC is in SC?”, it gives a piece of SC inside NC.

At NC$^1$, the picture is considerably murkier. Recall that Boolean NC$^1$ has multiple characterizations. #NC$^1$ as log depth arithmetic circuits has been studied quite a bit since first formally defined in CMTV. There it is shown that #BWBP equals #BP-M (or #BP-NFA, as referred to in [26]), contains functional NC$^1$, and is contained in #NC$^1$. But the reverse containments are still intriguingly open; we know that #NC$^1$ can be computed by Boolean poly size circuits of bounded fanin and depth $O(\log^2 n \log^* n)$, but the log$^* n$ factor remains. However, if the constant -1 is allowed, we get the classes GapNC$^1$ and GapBWBP, and these are known to coincide, and they are contained in FLOG.

It turns out that #LWF and #F both equal #NC$^1$. On the other hand, #BP-VPA equals #BWBP: adding a visibly pushdown stack to an NFA not only does not increase the complexity of the language class, it does not increase the complexity of the counting function class as well. Another arithmetization of a class equivalent to NC$^1$ is #sSC$^0$; this contains #BWBP and is contained in FLOG, but no relationship to #NC$^1$ is known. These results are described in [26].

Language classes can be defined based on these arithmetizations; the predicates typically applied to NC$^1$ and NLOG are:

- Is the # function greater than 0? yielding NC$^1$, NLOG
- Is the Gap function greater than 0? yielding PNC$^1$, PL
- Are two # (or Gap) functions equal? yielding C=NC$^1$, C=L

With these predicates, the multitude of arithmetic classes around NC$^1$ gives rise to a host of language classes between NC$^1$ and DLOG. I hope that the true picture is considerably simpler.
Acknowledgements

The idea of writing this survey arose after a talk I gave at a Dagstuhl seminar on Circuits, Logic and Games in November 2006, and I thank Heribert Vollmer for the invitation to participate in the seminar. I gratefully acknowledge many helpful and insightful discussions over several occasions with Eric Allender, Nutan Limaye, Kamal Lodaya, Pierre McKenzie, Antoine Meyer, Raghavendra Rao, Jayalal Sarma, H Venkateswaran and V Vinay.

References


The Bulletin of the EATCS


This installment of the concurrency column is devoted to a piece by Anna Ingolfsdottir and myself on a classic notion in concurrency theory that I feel deserves to be better known, viz. the notion of characteristic formula. To my mind, characteristic formulae offer one of the most satisfying connections between automata-based formalisms for the description of reactive computation, and specification logics. Moreover, as argued by several researchers—for instance, by Cleaveland and Steffen in an ICALP 1991 paper—, characteristic formulae may be used to compute behavioural relations efficiently via model checking.

By the time this issue of the Bulletin is available, the submission deadline for CONCUR 2007 will be approaching. I encourage the readers of this column to support the flagship conference in concurrency theory by submitting their best work to it.

Last, but not least, I wish all of the readers of this column a happy and productive 2007. I am always glad to receive proposals for columns. Do contact me if you would like to contribute to the concurrency column.
CHARACTERISTIC FORMULAE:
FROM AUTOMATA TO LOGIC

Luca Aceto  Anna Ingolfsdottir
Department of Computer Science, Reykjavík University
Kringlan 1, 103 Reykjavík, Iceland
and
BRICS (Basic Research in Computer Science)
Centre of the Danish National Research Foundation
Department of Computer Science, Aalborg University
Fr. Bajersvej 7B, 9220 Aalborg Ø, Denmark
Email: luca@ru.is, annai@ru.is

Abstract

This paper discusses the classic notion of characteristic formulae for processes using variations on Hennessy-Milner logic as the underlying logical specification language. It is shown how to characterize logically (states of) finite labelled transition systems modulo bisimilarity using a single formula in Hennessy-Milner logic with recursion. Moreover, characteristic formulae for timed automata with respect to timed bisimilarity and the faster-than preorder of Moller and Tofts are offered in terms of the logic $L_{\nu}$ of Laroussinie, Larsen and Weise.

1 Motivation

The aim of this paper is to introduce the basic ideas and results on a piece of classic concurrency theory that is, perhaps, not as well known as it deserves to be, namely the notion of characteristic formulae for processes. Characteristic formulae are neither a new nor a particularly hot topic in concurrency theory at the time of writing. However, we believe that the notion of characteristic formula sheds such a natural connection between automata-based formalisms as a way of describing the actual behaviour of processes and logics—for instance, modal or temporal logics—as formalisms for writing down specifications of the expected behaviour of processes that it is worth surveying some of the developments on
this branch of concurrency theory here. We state at the outset that our aim is not to give a comprehensive account of all of the work that has been carried out on characteristic formulae and their use in concurrency theory. Rather, we shall present the basic ideas underlying the notion of characteristic formulae for two specific models of concurrent computation—namely, (finite) labelled transition systems [24] and timed automata [4]—, and we shall refer the reader to the literature for further developments on, and applications of, this notion. Apart from the research papers we shall refer to in this survey, an introduction to characteristic formulae suitable for classroom use in courses on concurrency theory may be found in the forthcoming book [1].

What are Characteristic Formulae? Various types of automata are fundamental formalisms for the description of the behaviour of computing systems. For instance, a widely used model of computation is that of labelled transition systems (LTSs) [24]. LTSs underlie Plotkin’s Structural Operational Semantics [36, 37] and, following Milner’s pioneering work on CCS [32], are by now the formalism of choice for describing the semantics of various process description languages.

Since automata like LTSs can be used for describing specifications of process behaviours as well as their implementations, an important ingredient in their theory is a notion of behavioural equivalence or preorder between (states of) LTSs. A behavioural equivalence describes formally when (states of) LTSs afford the same ‘observations’, in some appropriate technical sense. On the other hand, a behavioural preorder is a possible formal embodiment of the idea that (a state in) an LTS affords at least as many ‘observations’ as another one. Taking the classic point of view that an implementation correctly implements a specification when each of its observations is allowed by the specification, behavioural preorders may therefore be used to establish the correctness of implementations with respect to their specifications, and to support the stepwise refinement of specifications into implementations.

The lack of consensus on what constitutes an appropriate notion of observable behaviour for reactive systems has led to a large number of proposals for behavioural equivalences and preorders for concurrent processes. In his by now classic papers [16, 17, 18], van Glabbeek presented a taxonomy of extant behavioural preorders and equivalences for processes.

The approach to the specification and verification of reactive systems in which automata like LTSs are used to describe both implementations and specifications of reactive systems is often referred to as implementation verification or equivalence checking.

An alternative approach to the specification and verification of reactive systems is that of model checking [8, 10, 39]. In this approach, automata are still the
formalism of choice for the description of the actual behaviour of a concurrent system. However, specifications of the expected behaviour of a system are now expressed using a suitable logic, for instance, a modal or temporal logic [14, 38]. Verifying whether a concurrent process conforms to its specification expressed as a formula in the logic amounts to checking whether the automaton describing the behaviour of the process is a model of the formula.

It is natural to wonder what the connection between these two approaches to the specification and verification of concurrent computation is. A classic, and most satisfying, result in the theory of concurrency is the characterization theorem of bisimulation equivalence [32, 35] in terms of Hennessy-Milner logic (HML) due to Hennessy and Milner [21]. This theorem states that two bisimilar processes satisfy the same formulae in Hennessy-Milner logic, and if the processes satisfy a technical finiteness condition, then they are also bisimilar when they satisfy the same formulae in the logic. This means that, for bisimilarity and HML, logical equivalence coincides with behavioural equivalence, and that, whenever two processes are not equivalent, we can always find a formula in HML that witnesses a reason why they are not. This distinguishing formula is useful for debugging purposes, and can be algorithmically constructed for finite processes—see, e.g., [26, 31]. (Algorithms for computing such distinguishing formulae for strong and weak bisimilarity are implemented in tools like the Edinburgh Concurrency Workbench.)

The characterization theorem of Hennessy and Milner is, however, less useful if we are interested in using it directly to establish when two processes are behaviourally equivalent using model checking. Indeed, that theorem seems to indicate that to show that two processes are equivalent we need to check that they satisfy the same formulae expressible in the logic, and there are countably many such formulae, even modulo logical equivalence. Is it possible to find a single formula that characterizes the bisimulation equivalence class of a process $p$—in the sense that any process is bisimilar to $p$ if, and only if, it affords that property? Such a formula, if it exists, is called a characteristic formula. When a characteristic formula for a process modulo a given notion of behavioural equivalence or preorder can be algorithmically constructed, implementation verification can be reduced to model checking, and, as the sub-title of our paper indicates, we can translate automata to logic. (An investigation of the model checking problems that can be reduced to implementation verification may, for instance, be found in the paper [6].)

To sum up the above discussion, characteristic formulae provide a very elegant connection between automata and logic, and between implementation verification and model checking. But, can they be constructed for natural, and suitably expressive, automata-based models and known logics of computation? To the best of our knowledge, this natural question was first addressed in the literature on concur-
The Bulletin of the EATCS

Can one characterize the equivalence class of an arbitrary finite process—for instance one described in the regular fragment of CCS—up to bisimilarity using HML? The answer is negative because each formula in that logic can only describe a finite fragment of the initial behaviour of a process—see, for instance, [1] for a textbook presentation. However, in the rest of this survey, we shall show that adding a facility for the recursive definition of formulae to (variants of) HML yields a logic that is powerful enough to support the construction of characteristic formulae for various types of finite processes modulo notions of behavioural equivalence or preorder. We shall focus on bisimilarity as a notion of behavioural equivalence between processes, but the formalism that we consider is powerful enough to handle a wealth of other semantics from van Glabbeek’s spectrum.

Roadmap of the Paper The rest of the paper is organized as follows. Section 2 discusses the construction of characteristic formulae for finite LTSs modulo bisimilarity using HML with a facility for the recursive definition of formulae as the logical specification language. We then proceed to show that both the logic and the approach used in that section in the setting of finite LTSs apply equally well to the formalism of timed automata. We conclude the paper with suggestions for further reading and further research on the topic of characteristic formulae (Section 4).

2 Characteristic Formulae for Finite LTSs Modulo Bisimilarity

As our first concrete example of characteristic formula construction, we now proceed to show how to build characteristic formulae for finite labelled transition systems modulo bisimilarity, using an extension of HML with a facility for the recursive definition of formulae.

Definition 2.1. [Labelled transition system] A labelled transition system (LTS) is a triple (Proc, Act, \( \rightarrow \), \( a \rightarrow a \in \text{Act} \)), where:

- Proc is a set of states (or processes);
- Act is a set of actions (or labels);
• \( \rightarrow \subseteq \text{Proc} \times \text{Proc} \) is a transition relation, for every \( a \in \text{Act} \). As usual, we shall use the more suggestive notation \( s \xrightarrow{a} s' \) in lieu of \( (s, s') \in \rightarrow \), and write \( s \xrightarrow{a} \) (read ‘s refuses \( a \)’) iff \( s \xrightarrow{a} s' \) for no state \( s' \).

A labelled transition system is finite if its sets of states and actions are both finite.

In this section, LTSs and their states will be considered modulo the classic notion of bisimulation equivalence [32, 35].

**Definition 2.2.** [Bisimulation and bisimilarity] A binary relation \( \mathcal{R} \) over the set of states of an LTS is a bisimulation iff whenever \( s_1 \mathcal{R} s_2 \) and \( a \) is an action:

- if \( s_1 \xrightarrow{a} s'_1 \), then there is a transition \( s_2 \xrightarrow{a} s'_2 \) such that \( s'_1 \mathcal{R} s'_2 \);
- if \( s_2 \xrightarrow{a} s'_2 \), then there is a transition \( s_1 \xrightarrow{a} s'_1 \) such that \( s'_1 \mathcal{R} s'_2 \).

Two states \( s \) and \( s' \) are bisimilar, written \( s \sim s' \), iff there is a bisimulation that relates them. Henceforth the relation \( \sim \) will be referred to as bisimulation equivalence or bisimilarity.

Consider a finite LTS \((\text{Proc}, \text{Act}, \{\xrightarrow{a} | a \in \text{Act}\})\). Our order of business will now be to show how to associate with each process \( p \in \text{Proc} \) a formula \( F_p \) in a suitable logic such that, for each process \( q \in \text{Proc} \),

\[ p \text{ is bisimilar to } q \text{ if, and only if, } q \text{ ‘affords the property } F_p \text{’}. \]

Such a formula \( F_p \) will be called the characteristic formula for process \( p \).

The logic that we shall use to define such characteristic formulae is an extension of HML with recursively defined formulae.

Let \( \mathcal{X} \) be a countably infinite collection of formula variables. The collection of formulae in Hennessy-Milner logic with recursion, denoted by \( \mathcal{M}_{\mathcal{X}} \), is given by the following grammar:

\[
F ::= X \mid \bot \mid F_1 \land F_2 \mid F_1 \lor F_2 \mid \langle a \rangle F \mid [a] F,
\]

where \( X \) ranges over \( \mathcal{X} \) and \( a \in \text{Act} \).

The meaning of formula variables is specified by means of a declaration. A declaration is a function \( D : \mathcal{X} \rightarrow \mathcal{M}_{\mathcal{X}} \) that associates a formula \( D(X) \) with each variable \( X \). Intuitively, if \( D(X) = F \), then \( X \) stands for the largest solution of the equation \( X = F \). In general, we shall only be interested in the restriction of a declaration to a finite collection of formula variables. In that case, we write

\[
\begin{align*}
X_1 & = F_{X_1} \\
& \vdots \\
X_n & = F_{X_n} 
\end{align*}
\]
where \(\{X_1, \ldots, X_n\}\) is a set of variables in \(X\), and, for \(1 \leq i \leq n\), the formula 
\[D(X_i) = F_{X_i}\] 
can only contain variables from \(\{X_1, \ldots, X_n\}\).

**Definition 2.3.** [Satisfaction relation] The binary relation \(\models\) relating processes in \(\text{Proc}\) to formulae in \(M_X\) is the largest relation satisfying the following clauses:

- \(p \models \top\), for each \(p\),
- \(p \models \bot\), for no \(p\),
- \(p \models F \land G\) implies \(p \models F\) and \(p \models G\),
- \(p \models F \lor G\) implies \(p \models F\) or \(p \models G\),
- \(p \models \langle a \rangle F\) implies \(p \xrightarrow{a} p'\) for some \(p'\) such that \(p' \models F\),
- \(p \models [a] F\) implies \(p' \models F\), for each \(p'\) such that \(p \xrightarrow{a} p'\), and
- \(p \models X\) implies \(p \models D(X)\).

The existence of the relation \(\models\) is guaranteed by classic fixed point theory [25, 41].

Semantically, a formula \(F\) that may contain occurrences of a finite subset \(\{X_1, \ldots, X_n\}\) of variables in \(X\) is interpreted as a function \(O_F\) that, given a vector of sets of processes \((S_1, \ldots, S_n)\) that are assumed to satisfy the formulae \(X_1, \ldots, X_n\), returns the set of processes that satisfy \(F\). Similarly, a mutually recursive system of equations of the form

\[
\begin{align*}
X_1 &= F_{X_1} \\
\vdots \\
X_n &= F_{X_n}
\end{align*}
\]

where \(\{X_1, \ldots, X_n\}\) is a set of variables in \(X\), and the formula \(F_{X_i}\) \((1 \leq i \leq n)\) can only contain variables from \(\{X_1, \ldots, X_n\}\), is interpreted as the largest vector of sets of processes \((S_1, \ldots, S_n)\) such that

\[
\begin{align*}
S_1 &= O_{F_{X_1}}(S_1, \ldots, S_n) \\
\vdots \\
S_n &= O_{F_{X_n}}(S_1, \ldots, S_n)
\end{align*}
\]

This means that the logic we have just introduced enriches classic HML with greatest fixed points, and is thus a fragment of Kozen’s \(\mu\)-calculus [27]. We refer
the reader to, for instance, [1] for more details on the semantics of Hennessy-Milner logic with recursion.

We are now ready to define the characteristic formula for each process \( p \in \text{Proc} \). A characteristic formula for a process has to describe both which actions the process can perform, which actions it cannot perform and what happens to it after it has performed each action. Let

\[
\text{Der}(a, p) = \{ p' \mid p \overset{a}{\rightarrow} p' \}
\]

be the set of states that can be reached from \( p \) by performing action \( a \). If \( p' \in \text{Der}(a, p) \) and \( p' \) has a characteristic property \( X_{p'} \), then \( p \) has the property \( \langle a \rangle X_{p'} \).

We therefore have that

\[
p \models \bigwedge_{a, p' \in \text{Der}(a, p)} \langle a \rangle X_{p'}.
\]

Furthermore, if \( p \overset{a}{\rightarrow} p' \) then \( p' \in \text{Der}(a, p) \). Therefore \( p \) has the property

\[
[a] \left( \bigvee_{p, p' \overset{a}{\rightarrow}} X_{p'} \right),
\]

for each action \( a \). The above property states that, by performing action \( a \), process \( p \) (and any other process that is bisimilar to it) must become a process satisfying the characteristic property of a state in \( \text{Der}(a, p) \). (Note that if \( p \not\overset{a}{\rightarrow} \), then \( \text{Der}(a, p) \) is empty. In that case, since an empty disjunction is just the formula \( ff \), the above formula becomes simply \( [a]ff \)—which is what we would expect.)

Since action \( a \) is arbitrary, we have that

\[
p \models \bigwedge a [a] \left( \bigvee_{p, p' \overset{a}{\rightarrow}} X_{p'} \right).
\]

If we summarize the above requirements, we may conclude that

\[
p \models \bigwedge_{a, p', p' \overset{a}{\rightarrow}} \langle a \rangle X_{p'} \land \bigwedge a [a] \left( \bigvee_{q, q' \overset{a}{\rightarrow}} X_{q'} \right).
\]

As this property is apparently a complete description of the behaviour of process \( p \), this is our candidate for its characteristic property. \( X_p \) is therefore defined as a solution to the equational system obtained by giving the following equation for each \( q \in \text{Proc} \):

\[
X_q = \bigwedge_{a, q' \overset{a}{\rightarrow} q} \langle a \rangle X_{q'} \land \bigwedge a [a] \left( \bigvee_{q', q'' \overset{a}{\rightarrow}} X_{q''} \right).
\]
Theorem 2.1. Let \((\text{Proc}, \text{Act}, [\alpha \mapsto a | a \in \text{Act}])\) be a finite transition system and, for each \(p \in \text{Proc}\), let \(X_p\) be defined by

\[
X_p = \bigwedge_{a, p', p' \xrightarrow{a} p} \langle a \rangle X_{p'} \land \bigvee_a \left[ a \left( \bigwedge_{p', p' \xrightarrow{a} p'} X_{p'} \right) \right].
\]

Then \(X_p\) is the characteristic property for \(p\)—that is, \(q \models X_p\) iff \(p \sim q\), for each \(q \in \text{Proc}\).

To the best of our knowledge, the above theorem was first proved in the MSc thesis [23]. The results in that study were later generalized to a family of bisimulation-like preorders and equivalences by Steffen and the second author in [40], which is by now the standard reference for characteristic formulae for finite labelled transition systems.

Remark 2.1. The above construction and theorem apply equally well to finitely branching LTSs with countably many states. However, in that case, the resulting characteristic formulae will use infinite systems of recursive equations.

3 Characteristic Formulae for Timed Automata

The approach to (automated) verification where the problem of checking behavioural relations between finite LTSs is reduced to model checking is advocated by Cleaveland and Steffen in [11, 12]. In their approach, the language being model checked is a logic equivalent in expressive power to the alternation-free fragment of the modal \(\mu\)-calculus [27]. The efficiency of this approach hinges on the following two facts:

1. the characteristic formula associated with a finite labelled transition system has size that is linear in that of the original LTS, and
2. the time complexity of determining whether a process satisfies a formula is proportional to the product of the sizes of the process and the formula.

The resulting algorithm offered in [12] is still considered to be one of the most efficient for checking behavioural preorders.

In the setting of modelling and verification for real-time systems, a characteristic formula construction for timed bisimulation equivalence over timed automata [4] has been offered in [29]. In op. cit., Laroussinie, Larsen and Weise have proposed the logic \(L_\nu\), a real-time version of Hennessy-Milner Logic [21] with greatest fixed points. Moreover, they have shown that its associated model checking problem is decidable, and that this logic is sufficiently expressive for
representing any timed automaton as a single characteristic $L_\nu$ formula. Such a formula uniquely characterizes the timed automaton up to timed bisimilarity.

The characteristic formula construction offered in [29], together with a model checking algorithm for the logic $L_\nu$, yields an algorithm for checking whether two timed automata are timed bisimilar, which may be seen as an implementation of the approach advocated in [12] in a real-time setting. Unfortunately, however, the characteristic formula construction for timed automata proposed in [29] produces formulae whose size is exponential in that of the original automaton, and this makes its use in checking timed bisimilarity for timed automata infeasible. The exponential blow-up involved in the characteristic formula construction from op. cit. is due to the fact that the formula is essentially constructed by applying the standard, untimed construction presented in Section 2 to the region graph associated with the timed automaton [4]. As shown by Alur and Dill in [4], the size of the region graph is exponential in that of the original timed automaton.

Our order of business in this section will be to present characteristic formula constructions for timed automata using the logic $L_\nu$ that, like those in the untimed setting and unlike that offered in [29], yield formulae whose size is linear with respect to that of the timed automaton they characterize.

We limit ourselves to presenting characteristic formula constructions for timed bisimilarity [42], and for the faster-than preorder proposed by Moller and Tofts in [33]. Constructions of characteristic formulae for some other behavioural relations over timed automata, as well as proofs of the results we mention in what follows, may be found in [2].

**Timed Labelled Transition Systems** Let $\operatorname{Act}$ be a finite set of actions, ranged over by $a$, and let $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ denote the sets of natural and non-negative real numbers, respectively. We use $\mathcal{L}$ to stand for the union of $\operatorname{Act}$ and $\mathbb{R}_{\geq 0}$. The meta-variable $\alpha$ will range over $\mathcal{L}$.

**Definition 3.1.** A timed labelled transition system (TLTS) is a structure $\mathcal{T} = (\mathcal{S}, \mathcal{L}, s^0, \rightarrow)$ where $\mathcal{S}$ is a set of states, $s^0 \in \mathcal{S}$ is the initial state, and $\rightarrow \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ is a transition relation satisfying the following properties:

- **(Time Determinism)** for every $s, s', s'' \in \mathcal{S}$ and $d \in \mathbb{R}_{\geq 0}$, if $s \xrightarrow{d} s'$ and $s \rightarrow s''$, then $s' = s''$;
- **(Time Additivity)** for every $s, s'' \in \mathcal{S}$ and $d_1, d_2 \in \mathbb{R}_{\geq 0}$, $s \xrightarrow{d_1 + d_2} s''$ iff $s \xrightarrow{d_1} s' \xrightarrow{d_2} s''$, for some $s' \in \mathcal{S}$;
- **(0-Delay)** for every $s, s' \in \mathcal{S}$, $s \xrightarrow{0} s'$ iff $s = s'$.
The Bulletin of the EATCS

The axioms of time determinism, time additivity and 0-delay are standard in the literature on Yi’s TCCS (see, for instance, [42]).

Timed Automata  Let $C$ be a set of clocks. We use $B(C)$ to denote the set of boolean expressions over atomic formulae of the form $x \approx m$ and $x - y \approx m$, with $x, y \in C$, $m \in \mathbb{N}$, and $\approx \in \{<, >, =\}$. Expressions in $B(C)$ are interpreted over the collection of time assignments. A time assignment, or valuation, $v$ for $C$ is a function from $C$ to $\mathbb{R}_{\geq 0}$. Given an expression $g \in B(C)$ and a time assignment $v$, we write $v \models g$ if $v$ satisfies $g$. Note that $B(C)$ is closed under negation. For every time assignment $v$ and $d \in \mathbb{R}_{\geq 0}$, we use $v + d$ to denote the time assignment which maps each clock $x \in C$ to the value $v(x) + d$. Two assignments $a$ and $v$ are said to agree on the set of clocks $C'$ iff they assign the same real number to every clock in $C'$. For every subset $C'$ of clocks, $v[C' \mapsto 0]$ denotes the assignment for $C$ which maps each clock in $C'$ to the value 0 and agrees with $v$ over $C \setminus C'$.

Definition 3.2. A timed automaton is a quintuple $A = (\text{Act}, N, n_0, C, E)$ where $N$ is a finite set of nodes, $n_0$ is the initial node, $C$ is a finite set of clocks, and $E \subseteq N \times N \times \text{Act} \times 2^C \times B(C)$ is a set of edges. The quintuple $e = (n, n', a, r_e, g_e) \in E$ stands for an edge from node $n$ to node $n'$ (the target of $e$) with action $a$, where $r_e$ denotes the set of clocks to be reset to 0 and $g_e$ is the enabling condition (or guard) over the clocks of $A$.

A state of a timed automaton $A$ is a pair $(n, v)$ where $n$ is a node of $A$ and $v$ is a time assignment for $C$. The initial state of $A$ is $(n_0, [C \mapsto 0])$ where $n_0$ is the initial node of $A$, and $[C \mapsto 0]$ is the time assignment mapping all clocks in $C$ to 0.

The operational semantics of a timed automaton $A$ is given by the timed labelled transition system $T_A = (S_A, \mathcal{L}, s_A^0, \rightarrow)$, where $S_A$ is the set of states of $A$, $s_A^0$ is the initial state of $A$, and $\rightarrow$ is the transition relation defined as follows:

$$(n, v) \xrightarrow{a} (n', v') \text{ iff } \exists e = (n, n', a, r_e, g_e) \in E. v \models g_e \land v' = v[r_e \mapsto 0]$$

$$(n, v) \xrightarrow{d} (n', v') \text{ iff } n = n' \text{ and } v' = v + d,$$

where $a \in \text{Act}$ and $d \in \mathbb{R}_{\geq 0}$.

The Logic $L_\nu$ The logic $L_\nu$ is a real-time version of Hennessy-Milner Logic with greatest fixed points that stems from [29]. We now briefly review its syntax and semantics for the sake of completeness.

Definition 3.3. [Syntax of $L_\nu$] Let $K$ be a finite set of formula clocks, $\text{Id}$ a finite set of identifiers and $k$ a non-negative integer. The set $L_\nu$ of formulae over $K$,
\[ \varphi ::= t \mid \text{ff} \mid \varphi \land \psi \mid \varphi \lor \psi \mid \exists D \varphi \mid \forall D \varphi \mid \langle a \rangle \varphi \mid \lbrack a \rbrack \varphi \mid x \in D \varphi | x \ni m | x + m \ni y + \ell | Z \].

where \( a \in \text{Act} \), \( x, y \in K \), \( \ell, m \in \{0, \ldots, k\} \), \( \ni \in \{=, <, \leq, >, \geq\} \) and \( Z \in \text{Id} \).

The logic \( L_\nu \) allows for the recursive definition of formulae by including a finite set \( \text{Id} \) of identifiers. The formula associated with each of the identifiers is specified by a declaration \( D \), i.e., \( D \) assigns a formula of \( L_\nu \) to each identifier. For an identifier \( Z \) we let \( Z = \varphi \) denote \( D(Z) = \varphi \). Intuitively \( Z \) will stand for the largest solution of the equation \( Z = \varphi \). We refer the interested reader to [1, 29] for more information on \( L_\nu \).

Given a timed automaton \( A \), whose set of clocks \( C \) is disjoint from \( K \), we interpret the formulae in \( L_\nu \) over extended states. An extended state of \( A \) is a pair \((n, vu)\), where \((n, v)\) is a state of \( A \), \( u \) is a time assignment for \( K \), and we use \( vu \) for the assignment over \( C \cup K \) that agrees with \( v \) over \( C \) and with \( u \) over \( K \).

**Definition 3.4.** [Semantics of \( L_\nu \)] Let \( A \) be a timed automaton and \( D \) a declaration. The satisfaction relation \( \models_D \) is the largest relation satisfying the following implications:

\[
\begin{align*}
(n, vu) &\models_D t \quad \Rightarrow \quad \text{true} \\
(n, vu) &\models_D \text{ff} \quad \Rightarrow \quad \text{false} \\
(n, vu) &\models_D \varphi \land \psi \quad \Rightarrow \quad (n, vu) \models_D \varphi \text{ and } (n, vu) \models_D \psi \\
(n, vu) &\models_D \varphi \lor \psi \quad \Rightarrow \quad (n, vu) \models_D \varphi \text{ or } (n, vu) \models_D \psi \\
(n, vu) &\models_D \exists D \varphi \quad \Rightarrow \quad \exists d \in \mathbb{R}_{\geq 0}. (n, (v + d)(u + d)) \models_D \varphi \\
(n, vu) &\models_D \forall D \varphi \quad \Rightarrow \quad \forall d \in \mathbb{R}_{\geq 0}. (n, (v + d)(u + d)) \models_D \varphi \\
(n, vu) &\models_D \langle a \rangle \varphi \quad \Rightarrow \quad \exists (n', v'). (n, v) \xrightarrow{a} (n', v') \text{ and } (n', v'u) \models_D \varphi \\
(n, vu) &\models_D \lbrack a \rbrack \varphi \quad \Rightarrow \quad \forall (n', v'). (n, v) \xrightarrow{a} (n', v') \text{ implies } (n', v'u) \models_D \varphi \\
(n, vu) &\models_D x \ni m \quad \Rightarrow \quad u(x) \ni m \\
(n, vu) &\models_D x + m \ni y + \ell \quad \Rightarrow \quad u(x) + m \ni u(y) + \ell \\
(n, vu) &\models_D x \in D \varphi \quad \Rightarrow \quad (n, vu') \models_D \varphi \text{ where } u' = u[x] \mapsto 0 \\
(n, vu) &\models_D D(Z) \quad \Rightarrow \quad (n, vu) \models_D D(Z).
\end{align*}
\]

Again, the existence of \( \models_D \) follows from standard fixed point theory [25, 41].

In the remainder of this section, we shall use the logic \( L_\nu \) to construct characteristic formulae for timed automata with respect to timed bisimilarity [42], and for the faster-than preorder [33].
The notion of timed bisimulation stems from [42]. It is the obvious adaptation to the timed setting of the classic definition presented in Definition 2.2.

**Definition 3.5.** [Timed bisimulation] Let $T = (S, L, s^0, \rightarrow)$ be a TLTS. A timed bisimulation is a relation $R \subseteq S \times S$ such that whenever $s_1 R s_2$ and $\alpha \in L$, then:

- If $s_1 \xrightarrow{\alpha} s'_1$ then $s_2 \xrightarrow{\alpha} s'_2$ for some $s'_2$ such that $s'_1 R s'_2$.

- If $s_2 \xrightarrow{\alpha} s'_2$ then $s_1 \xrightarrow{\alpha} s'_1$ for some $s'_1$ such that $s'_1 R s'_2$.

For states $s_1, s_2$, we write $s_1 \sim_T s_2$ iff there exists a timed bisimulation $R$ with $s_1 R s_2$.

Moller and Tofts [33] have proposed a preorder on processes that distinguishes functionally behaviourally equivalent processes which operate at different speed. Their original proposal applied to their version of timed CCS, but it is simple enough to adapt it to the setting of TLTSs.

**Definition 3.6.** [Faster-than bisimulation] Let $T = (S, L, s^0, \rightarrow)$ be a TLTS. A faster-than bisimulation is a relation $R \subseteq S \times S$ such that whenever $s_1 R s_2$, $a \in \text{Act}$ and $d \in \mathbb{R}_{\geq 0}$ then:

1. if $s_1 \xrightarrow{a} s'_1$ then there are $d \in \mathbb{R}_{\geq 0}, s''_1, s'_2$ and $s''_2$ such that $s'_1 \xrightarrow{d} s''_1, s_2 \xrightarrow{d} s''_2, s'_2 \xrightarrow{a} s''_2$, and $s'_1 R s''_2$;

2. if $s_2 \xrightarrow{a} s'_2$ then $s_1 \xrightarrow{a} s'_1$ for some $s'_1$ such that $s'_1 R s''_2$;

3. if $s_1 \xrightarrow{d} s'_1$ then $s_2 \xrightarrow{d} s'_2$ for some $s'_2$ such that $s'_1 R s''_2$;

4. if $s_2 \xrightarrow{d} s'_2$ then $s_1 \xrightarrow{d} s'_1$ for some $s'_1$ such that $s'_1 R s''_2$.

For states $s_1, s_2$, we write $s_1 \not\subseteq_{FT} s_2$ iff there exists a faster-than bisimulation $R$ with $s_1 R s_2$.

It is well-known that $\not\subseteq_{FT}$ is a preorder, and is the largest faster-than bisimulation. Similarly, $\sim_T$ is an equivalence relation, and is the largest timed bisimulation. Both of the previously defined behavioural relations can be lifted to the setting of timed automata as follows.

**Definition 3.7.** Let $A, B$ be two timed automata. We write $A \sim_T B$ iff $s^0_A \sim_T s^0_B$ in the TLTS that results by taking the disjoint union of $T_A$ and $T_B$. Similarly, we write $A \not\subseteq_{FT} B$ iff $s^0_A \not\subseteq_{FT} s^0_B$ in the TLTS that results by taking the disjoint union of $T_A$ and $T_B$. 
Characteristic Formula Constructions

To increase the readability of the characteristic formulae we make use of some derived constructs in the logic $L$. These we now present for the sake of clarity.

For a reset set $r = \{x_1, \ldots, x_k\}$, we use the abbreviation $r \in \varphi$ to stand for the formula inductively defined thus:

$$
\emptyset \in \varphi = \varphi
$$

$$
\{x_1, \ldots, x_k\} \in \varphi = x_1 \in (\{x_2, \ldots, x_k\} \in \varphi) \quad (k \geq 1).
$$

Note that the order of the clocks is arbitrary because $x \in (y \in \varphi)$ is logically equivalent to $y \in (x \in \varphi)$.

The expression $g \Rightarrow \varphi$ will stand for $\neg g \lor \varphi$, where $\neg g$ is the negation of the guard $g$. This is a formula in $L$, because the collection of guards is closed under negation.

In the remainder of this section, we shall implicitly assume a given timed automaton $A$, for which the characteristic formulae will be defined. Given a node $n$ in $A$, and action $a$, we use $E(n, a)$ to stand for the set of $a$-labelled edges stemming from node $n$.

We first consider timed bisimilarity. A formula characterizing a node of a timed automaton up to timed bisimilarity should offer a description of:

1. all the actions that are enabled in the node,
2. which node is entered by taking a given transition, together with the resets associated with it, and
3. the fact that arbitrary delays are allowed in the node.

The resulting characteristic formula is presented below, where we consider each $\Phi(n)^\sim_T$ to be an identifier. The formula consists of three sets of conjuncts, each associated to one of the above properties, for each node $n$ of a timed automaton $A$:

$$
\Phi^\sim_T(n) = \bigwedge_{a \in \text{Act}} \bigwedge_{e \in E(n, a)} g_e \Rightarrow ((a) r_e \in \Phi^\sim_T(n_e)) \land
$$

$$
\bigwedge_{a \in \text{Act}} [a] (\bigvee_{e \in E(n, a)} g_e \land (r_e \in \Phi^\sim_T(n_e))) \land
$$

$$
\forall \Phi^\sim_T(n),
$$

where $n$ is a node of $A$, $e = (n, n_e, a, r_e, g_e)$, and we recall that $E(n, a)$ denotes the set of $a$-labelled edges from node $n$. We shall use $D^\sim_T$ to denote the declaration that consists of the equations above, one for each node of $A$. 

70
The Bulletin of the EATCS

**Theorem 3.1.** Let \( A, B \) be timed automata with disjoint sets of clocks. Let \( n \) be a node of \( A \) and \( m \) be a node of \( B \). Assume that \( u \) and \( v \) are valuations for the clocks of \( A \) and \( B \), respectively. Then

\[
(n, u) \sim_T (m, v) \iff (m, vu) \models \Phi^\sim_T(n)
\]

where \((m, vu) \models \Phi^\sim_T(n)\) holds with respect to the declaration \( D_A^\sim_T \).

In the characteristic formula construction for timed bisimilarity, no use was made of the existential modality \( \exists \exists \) over delay transitions. The use of the \( \exists \exists \) modality will instead play a crucial role in the definition of the characteristic property for the faster-than bisimulation preorder. This we now proceed to present.

For every node \( n \) in a timed automaton \( A \), we define:

\[
\Phi^{\leq FT}(n) = (\bigwedge_{a \in \text{Act}} \bigwedge_{e \in E(n, a)} \forall g_e \rightarrow (r_e \in \exists \exists (a) \Phi^{\leq FT}(n_e))) \land \\
(\bigwedge_{a \in \text{Act}} [a] \bigvee_{e \in E(n, a)} g_e \land (r_e \in \exists \exists \Phi^{\leq FT}(n_e))) \land \\
\forall \Phi^{\leq FT}(n),
\]

where \( e = (n, n_e, a, r_e, g_e) \) and \( E(n, a) \) denotes the set of \( a \) labelled edges from node \( n \). We shall use \( D_A^{\leq FT} \) to denote the declaration that consists of the equations above, one for each node of \( A \).

**Theorem 3.2.** Let \( A, B \) be timed automata with disjoint sets of clocks. Let \( n \) be a node of \( A \) and \( m \) be a node of \( B \). Assume that \( u \) and \( v \) are valuations for the clocks of \( A \) and \( B \), respectively. Then

\[
(n, u) \leq_{FT} (m, v) \iff (m, vu) \models \Phi^{\leq FT}(n)
\]

where \((m, vu) \models \Phi^{\leq FT}(n)\) holds with respect to the declaration \( D_A^{\leq FT} \).

It is interesting to remark that no other characteristic formula construction presented in [2] uses the the existential modality \( \exists \exists \) over delay transitions.

### 4 Suggestions for Further Reading

It is an instructive exercise to construct characteristic formulae for (states of) variations on finite LTSs in variants of HML with greatest fixed points for other (bi)simulation based behavioural relations in van Glabbeek’s spectrum. This bears
witness to the naturalness of this logic for the specification of behavioural properties of reactive systems modelled as LTSs. Examples of such constructions may be found in, for instance, [15, 34]. The former reference offers a characteristic formula in terms of the μ-calculus for each finite underspecified transition system—essentially a transition system where transitions may have sets of states as their target. The latter shows how to derive characteristic formulae in the μ-calculus for finite LTSs up to strong or weak bisimilarity directly from the characterization of those relations in terms of greatest fixed points.

All of the results we have surveyed in this paper show that, in light of its beautiful connection with bisimilarity, HML and its variations are prime candidates for logics in which to express characteristic properties for bisimulation-like relations. However, there are other options.

A classic, early result on characteristic formulae was obtained in the paper [7]. That paper shows that each finite Kripke structure can be characterized by a formula in Computation Tree Logic (CTL) [9] up to the natural variation on bisimilarity over Kripke structures.

Another characteristic formula result is presented in that paper for an equivalence between states that takes ‘stuttering’ into account. (This equivalence is closely related to van Glabbeek’s and Weijland’s branching bisimilarity [19], for which logical characterizations have been offered by De Nicola and Vaandrager in the paper [13].) Browne, Clarke and Grümberg show that equivalence classes of states in a finite Kripke structure modulo stuttering equivalence are completely characterized by next-time-free CTL formulae. (The absence of the next-time operator is expected in light of the inability of stuttering equivalence to ‘count’ the number of steps in a stuttering sequence.) Kučera and Schnoebelen have presented a refinement of the above classic theorem by Browne, Clarke and Grümberg in the paper [28]. To the best of our knowledge, it is not known whether the timed version of CTL presented in [3] is sufficiently expressive to characterize timed bisimilarity.

Larsen and Skou present a characteristic formula construction in a probabilistic variation on HML for a recursion-free calculus of probabilistic processes in [30].

Recently, Berger, Honda and Yoshida have been investigating the notion of descriptive completeness for logics of higher-order functions. For instance, in their paper [22], they show that, given a program in call-by-value PCF, one can construct a Hoare triple representing the program’s behaviour up to observational equivalence. This notion is the counterpart of characteristic formulae in the setting of program logics. Our readers will find further information on the, by now very substantial, body of work on this topic by Berger, Honda and Yoshida at the URL http://www.dcs.qmul.ac.uk/~kohei/logics/index.html.
Despite all of the aforementioned, classic studies on the notion of characteristic formula, we feel that there is still scope for further investigation, both from the point of view of theory and from that of applications. We hope that this small paper will contribute to a renewed interest in this topic.

References


THE DISTRIBUTED COMPUTING COLUMN

by

MARIO MAVRONICOLAS

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

OPODIS 2006 REPORT

Burkhard Englert

Department of Computer Engineering and Computer Science
California State University Long Beach
1250 Bellflower Blvd.
Long Beach CA 90840
USA
benglert@csulb.edu

Abstract

The International Conference On Principles Of Distributed Systems, OPODIS, is an annually held open forum for the exchange of state-of-the-art knowledge on principles of distributed computing and systems among researchers from around the world. The 10th anniversary conference was held from December 12 to 15 in Bordeaux France. It featured 4 invited talks, 28 accepted papers out of over 230 submissions and 3 brief announcements.

1 Scientific content

OPODIS, the International Conference On Principles Of Distributed Systems, is an annual forum for the exchange of state-of-the-art knowledge on principles of distributed computing and systems among researchers from around the world. The 10th anniversary edition of OPODIS was held from December 12-15, 2006, in Bordeaux, France.
Bordeaux, France. This year an unprecedented number of 230 papers were submitted, out of which 28 papers were accepted as regular papers and 3 as brief announcements. The decisions were made by the Program Committee during an electronic meeting held during the week of September 10th, 2006, following a review period. The symposium also featured keynote addresses by Amir Pnueli (Weizmann Institute, Israel), Butler Lampson (Microsoft, USA), Michel Raynal (IRISA, France), and Gerard Roucairol (Bull, France). Based on the attention and the quality of the submissions it received it is clear that OPODIS has found its place among the conferences related to principles of distributed computing, networks, and systems.

The proceedings of OPODIS 2006 were published in the series Springer Lecture Notes in Computer Science (LNCS) Volume 4305.

The general chair of the OPODIS conference is Ivan Lavallée, Université de Paris 8. The conference was organized by Thibault Bernard from the Université de Reims, France, Celine Butelle from the Université de Paris 8, France, Nicole Lavallée, Devan Sohier from EPHE France. The excellent local organization and the duties of Publicity Chair were handled by Antoine Rollet from ENSEIRB in Bordeaux. The Steering Committee of the conference consists of Alain Bui from the Université de Reims, France, Marc Bui from EPHE, France, Hacène Fouchal from the Université de Antilles-Guyane, France, Roberto Gomez from ITESM-CEM, Mexico, Nicola Santoro from Carleton University, Canada and Philippas Tsigas from Chalmers University, Sweden. The Program Committee was chaired by Alex Shvartsman from the University of Connecticut and MIT, USA. The members of the program committee were James Anderson - University of North Carolina, Anish Arora - Ohio State University, Hagit Attiya - The Technion Israel, Joffroy Beauquier - Université de Paris 11, Riccardo Bettati - Texas A&M University, Jianmong Cao - Polytechnic University Hong Kong, Richard Castanet - ENSEIRB Bordeaux, Ajoy Datta - University of Nevada Las Vegas, Michael Fischer - Yale University, Paola Flocchini - Ottawa University, Chryssis Georgiou - University of Cyprus, Mohamed Gouda - University of Texas, Teruo Higashino - Osaka University, Friedhelm Meyer auf der Heide - University of Paderborn, Mikhail Nesterenko - Kent State University, Marina Papatriantafilou - Chalmers University Sweden, Boaz Patt-Shamir - Tel Aviv University, Andrzej Pelc - University of Quebec, Giuseppe Prencipe - University of Pisa, Sergio Rajarbaum - UNAM Mexico, Michel Raynal - IRISA France, André Schiper - EPF Lausanne, Ulrich Schmid - Technical University of Vienna, Marc Shapiro - INRIA France, Nir Shavit - Sun Microsystems, David Simplot-Ryl - Université de Lille, Paul Spirakis - University of Patras, Mark Tuttle - Intel, Vincent Villain - Université de Picardie France, Roger Wattenhofer - ETH Zurich, Carlos Becker Westphall - University of St. Catarina, Brazil and Peter Widmayer - ETH Zurich.

The conference was sponsored by AUF - Agence Universitaire de la Franco-
Scientific program - Invited Talks The exciting scientific program included four very interesting invited presentations.

- Butler Lampson from Microsoft USA spoke on *Lazy and Speculative Execution in Computer Systems*.

  The distinction between lazy and eager (or strict) evaluation has been studied in programming languages since Algol 60's call by name, as a way to avoid unnecessary work and to deal gracefully with infinite structures such as streams. It is deeply integrated in some languages, notably Haskell, and can be simulated in many languages by wrapping a lazy expression in a lambda. Less well studied is the role of *laziness*, and its opposite, *speculation*, in computer systems, both hardware and software. A wide range of techniques can be understood as applications of these two ideas. In both cases it is usual to insist that laziness or speculation is strictly a matter of scheduling that doesn’t affect the result of a computation but only improves the performance. Sometimes, however, the spec is weakened, for example in eventual consistency.

  In his presentation Butler Lampson examined many of the examples of the role of laziness and speculation in computer systems in detail to investigate what they have in common, how they differ, and what factors govern the effectiveness of laziness and speculation in computer systems.

- The second invited talk was given by Michel Raynal based on joint work with Corentin Travers both from IRISA France. The title of his presentation was *In search of the holy grail: Looking for the weakest failure detector for wait-free set agreement*.

  Asynchronous failure detector-based set agreement algorithms proposed so far assume that all the processes participate in the algorithm. This means that (at least) the processes that do not crash propose a value and consequently execute the algorithm. It follows that these algorithms can block forever (preventing the correct processes from terminating) when there are correct processes that do not participate in the algorithm. This paper investigates the wait-free set agreement problem, i.e., the case where the correct
participating processes have to decide a value whatever the behavior of the other processes (i.e., the processes that crash and the processes that are correct but do not participate in the algorithm). The paper presents a wait-free set agreement algorithm. This algorithm is based on a leader failure detector class that takes into account the notion of participating processes. Interestingly, this algorithm enjoys a first class property, namely, design simplicity.

• Amir Pnueli from the Weizmann Institute in Israel spoke about *Automatic Synthesis of Programs from Property Specifications*. The current state of the art uses specification of properties in a behavioral form, such as linear temporal logic, mainly for static and dynamic verification of implementations of reactive systems, AFTER they have been constructed. In the talk we consider the more radical approach by which a specification in terms of its desired properties is formed at the beginning of a system development, before any implementation is attempted. After some analysis and validation of this specification, one may apply a synthesis algorithm in order to automatically derive a correct-by-construction implementation. No further verification or testing is required. While this has been considered an unachievable dream for a long time, recent progress in the algorithms for synthesis has made program synthesis from property specification applicable for medium size systems. In the talk, we provide more details about the problem of synthesizing reactive programs from their LTL (and similar) specifications. In spite of the theoretical double exponential lower bound, we show that for many expressive specifications of reactive systems the problem can be solved in time $N^3$. We describe the context of the problem, as part of the Prosyd European Project which aims to provide a property-based development flow for hardware designs. Within this project, synthesis plays an important role, first in order to check whether a given specification is realizable, and then for synthesizing part of the developed system.

• The fourth invited talk was given by Gerard Roucairol from Bull, France with the title *The future of massive parallel computers*.

2 Scientific Program - Accepted Papers

Many different areas of distributed Computing were very well represented in the remaining scientific program of the conference.

2.1 Distributed Algorithms I

This year for the first time a best paper award was given at OPODIS. Rachid Guerraoui from EPF Lausanne Switzerland received it for a paper coauthored
The title of their paper is *A Topological Treatment of Early-deciding et-agreement*.

Their paper considers the \( k \)-set-agreement problem in a synchronous message passing distributed system where up to \( t \) processes can fail by crashing. They determine the number of communication rounds needed for all correct processes to reach a decision in a given run, as a function of \( k \), the degree of coordination, and \( f \leq t \) the number of processes that actually fail in the run. They prove a lower bound of \( \min(\lceil f/k + 2 \rceil, \lceil t/k + 1 \rceil) \) rounds. Their proof uses simple topological tools to reason about runs of a full information set-agreement protocol. In particular, they introduce a topological operator, which they call the *early deciding* operator, to capture rounds where \( k \) processes fail but correct processes see only \( k \) failures.

**Renaming with \( k \)-Set-Consensus: An Optimal Algorithm into \( n + k - 1 \) Slots** by Eli Gafni.

Recently Mostefaoui, Raynal, and Travers showed that with the use of \( k \)-set-consensus they can strongly rename \( n \) processors into the range of \( 2^n - \lceil n/k \rceil \). That is the overhead is \( n - \lceil n/k \rceil \). In this paper the author reduces the range to \( n + k - 1 \), i.e. he reduces the overhead to \( k - 1 \). For \( k = c \) the improvement in the overhead is \( O(n) \). The author first argues that such an algorithm exists using a topological embedding. The author then presents a novel explicit algorithm that does not require explicit embedding. The latter technique employed is of independent importance and interest.

**When Consensus Meets Self-Stabilization** by Shlomi Dolev, Ronen I. Kat and Elad M. Schiller.

This paper presents a self-stabilizing failure detector, asynchronous consensus and replicated state-machine algorithm suite, the components of which can be started in an *arbitrary state* and converge to act as a virtual state-machine.

Self-stabilizing algorithms can cope with transient faults. Transient faults can alter the system state to an arbitrary state and hence, cause a temporary violation of the safety property of the consensus. New requirements for consensus that fit the on-going nature of self-stabilizing algorithms are presented. The wait-free consensus (and the replicated state-machine) algorithm presented is a classic combination of a failure detector and a (memory bounded) rotating coordinator consensus that satisfy both *eventual safety* and *eventual liveness*. 
Several new techniques and paradigms are introduced. The bounded memory failure detector abstracts away synchronization assumptions using bounded heartbeat counters combined with a balance-unbalance mechanism. The practically infinite paradigm is introduced in the scope of self-stabilization, where an execution of, say, $2^{64}$ sequential steps is regarded as (practically) infinite. Finally, they present the first self-stabilizing wait-free reset mechanism that ensures eventual safety and can be used in other scopes.

- **On the Cost of Uniform Protocols whose Memory Consumption is Adaptive to Interval Contention** by Burkhard Englert.

A distributed shared memory protocol is called memory-adaptive, if all writes to MWMR registers are "close to the beginning of shared memory", that is the indices of all MWMR registers processes write to when executing the protocol are functions of the contention. The notion of memory-adaptiveness captures what it means for a distributed protocol to most efficiently make use of its shared memory. The author et al. previously considered a store/release protocol where processes are required to store a value in shared MWMR memory so that it cannot be overwritten until it has been released by the process. They showed that there do not exist uniformly wait-free store/release protocols using only the basic operations read and write that are memory-adaptive to point contention. They further showed that there exists a uniformly wait-free store/release protocol using only the basic operations read, write, and read-modify-write that is memory-adaptive to interval contention and time-adaptive to total contention. This left a significant gap which the author closes in this paper. He shows that no uniform store/release protocol can exist that is memory adaptive to interval contention and only uses read/write (no read-modify-write) registers. The author furthermore illustrates the validity and practicality of the concept of memory adaptiveness by providing a uniform, memory-adaptive to interval contention store/release protocol for Network Attached Disks.

- **Optimistic Algorithms for Partial Database Replication** by Nicolas Schiper, Rodrigo Schmidt and Fernando Pedone.

In this paper, the authors study the problem of partial database replication. Numerous previous works have investigated database replication, however, most of them focus on full replication. In this paper the authors are interested in genuine partial replication protocols, which require replicas to permanently store only information about data items they replicate. They define two properties to characterize partial replication. The first one, Quasi-Genuine Partial Replication, captures the above idea; the second one, Non-
Trivial Certification, rules out solutions that would abort transactions unnecessarily in an attempt to ensure the first property. They also present two algorithms that extend the Database State Machine to partial replication and guarantee the two aforementioned properties. Their algorithms compare favorably to existing solutions both in terms of number of messages and communication steps.

2.2 Real Time Systems

The next group of papers dealt with real time system issues.


  This paper introduces a simple real-time distributed computing model for message-passing systems, which reconciles the distributed computing and the real-time systems perspective: By just replacing instantaneous computing steps with computing steps of non-zero duration, they obtain a model that both facilitates real-time scheduling analysis and retains compatibility with classic distributed computing analysis techniques and results. As a by-product, it also allows us to investigate whether/which properties of real systems are inaccurately or even wrongly captured when resorting to zero step-time models. They revisit the well-studied problem of deterministic internal clock synchronization for this purpose, and show that, contrary to the classic model, no clock synchronization algorithm with constant running time can achieve optimal precision in their real-time model. They prove that optimal precision is only achievable with algorithms that take $\Omega(n)$ time in their model, and establish several additional lower bounds and algorithms.

- **Distributed Priority Inheritance for Real-Time and Embedded Systems** by César Sánchez, Henny B. Sipma, Christopher D. Gill and Zohar Manna.

  In this paper they study the problem of priority inversion in distributed real-time and embedded systems and propose a solution based on a distributed version of the priority inheritance protocol (PIP). Previous approaches to priority inversions in distributed systems use variations of the priority ceiling protocol (PCP), originally designed for centralized systems as a modification of PIP that also prevents deadlock. PCP, however, requires maintaining a global view of the acquired resources, which in distributed systems leads to high communication overhead.

  This paper presents a distributed PIP built on top of a deadlock avoidance schema that requires much less communication than PCP. Since the system
is already deadlock free and priority inversions can be detected locally, they obtain an efficient dynamic resource allocation system that prevents deadlocks and handles priority inversions.

- **Safe Termination Detection in an Asynchronous Distributed System when Processes may Crash and Recover** by Neeraj Mittal, Kuppahalli L. Phaneesh and Felix C. Freiling.

  The termination detection problem involves detecting whether an ongoing distributed computation has ceased all its activities. The authors investigate the termination detection problem in an asynchronous distributed system under crash-recovery model. It has been shown that the problem is impossible to solve under crash-recovery model in general. They identify two conditions under which the termination detection problem can be solved in a safe manner. They also propose algorithms to detect termination under the conditions identified.

- **Lock-free Dynamically Resizable Arrays** by Damian Dechev, Peter Pirkelbauer and Bjarne Stroustrup.

  The authors present a first lock-free design and implementation of a dynamically resizable array (vector). The most extensively used container in the C++ Standard Template Library (STL) is vector, offering a combination of dynamic memory management and constant-time random access. Their approach is based on a single 32-bit word atomic compare-and-swap (CAS) instruction. It provides a linearizable and highly parallelizable STL-like interface, lock-free memory allocation and management, and fast execution. Their current implementation is designed to be most efficient on multi-core architectures. Experiments on a dual-core Intel processor with shared L2 cache indicate that their lock-free vector outperforms its lock-based STL counterpart and the latest concurrent vector implementation provided by Intel by a large factor. The performance evaluation on a quad dual-core AMD system with non-shared L2 cache demonstrated timing results comparable to the best available lock-based techniques. The presented design implements the most common STL vector’s interfaces, namely random access read and write, tail insertion and deletion, pre-allocation of memory, and query of the container’s size. Using the current implementation, a user has to avoid one particular ABA problem.

### 2.3 Distributed Algorithms II

- **Distributed Spanner Construction in Doubling Metric Spaces** by Mirela Damian, Saurav Pandit, and Sriram Pemmaraju.
This paper presents a distributed algorithm that runs on an \( n \)-node unit ball graph (UBG) \( G \) residing in a metric space of constant doubling dimension, and constructs, for any \( \epsilon > 0 \), a \((1+\epsilon)\)-spanner \( H \) of \( G \) with maximum degree bounded above by a constant. In addition, they show that \( H \) is "lightweight", in the following sense. Let \( \Delta \) denote the aspect ratio of \( G \), that is, the ratio of the length of a longest edge in \( G \) to the length of a shortest edge in \( G \). The total weight of \( H \) is bounded above by \( O(\log \Delta \cdot \text{wt}(\text{MST})) \), where \( \text{MST} \) denotes a minimum spanning tree of the metric space. Finally, they show that \( H \) satisfies the so called leapfrog property, an immediate implication being that, for the special case of Euclidean metric spaces with fixed dimension, the weight of \( H \) is bounded above by \( O(\text{wt}(\text{MST})) \). Thus, the current result subsumes the results of the authors in PODC 2006 that apply to Euclidean metric spaces, and extends these results to metric spaces with constant doubling dimension.

- **Verification Techniques for Distributed Algorithms** by Anna Philippou and George Michael.

A value-passing, asynchronous process calculus and its associated theory of confluence are considered as a basis for establishing the correctness of distributed algorithms. In particular, the authors present an asynchronous version of value-passing CCS and they develop its theory of confluence. They show techniques for demonstrating confluence of complex processes in a compositional manner and they study properties of confluent systems that can prove useful for their verification. These results give rise to a methodology for system verification which they illustrate by proving the correctness of two distributed leader-election algorithms.


In this paper, the authors are interested in the computational power of a mobile agent system and, more particularly, in the comparison with a message passing system. First they give formal definitions. Then they explain how a mobile agent algorithm can be simulated by a message passing algorithm. They also prove that any message passing algorithm can be implemented by a mobile agent algorithm. As a consequence of this result, known characterizations of solvable tasks by message passing algorithms can be translated into characterisations of solvable tasks by mobile agent algorithms. They illustrate this result with the election problem.
The Bulletin of the EATCS

2.4 Sensor Networks

The following papers deal with issues in Sensor Networks.

- **Incremental Construction of k-Dominating Sets in Wireless Sensor Networks** by Mathieu Couture, Michel Barbeau, Prosenjit Bose and Evangelos Kranakis.

  Given a graph $G$, a $k$-dominating set of $G$ is a subset $S$ of its vertices with the property that every vertex of $G$ is either in $S$ or has at least $k$ neighbors in $S$. The authors present a new incremental distributed algorithm to construct a $k$-dominating set. The algorithm constructs a monotone family of dominating sets $D_1 \subseteq D_2 \subseteq \ldots \subseteq D_i \subseteq \ldots \subseteq D_k$ such that each $D_i$ is an $i$-dominating set. For unit disk graphs, the size of each of the resulting $i$-dominating sets is at most six times the optimal.

- **Of Malicious Motes and Suspicious Sensors - On the efficiency of malicious interference in wireless networks** by Seth Gilbert, Rachid Guerraoui and Calvin Newport.

  In this paper the authors ask how efficiently a malicious device can disrupt communication in a wireless network. They imagine a basic game involving two honest players, Alice and Bob, who want to exchange information, and an adversary, Collin, who can disrupt communication using a limited budget of $adv$ broadcasts. They investigate how long Collin can delay Alice and Bob from communicating. The authors note that the trials and tribulations of Alice and Bob capture the fundamental difficulty shared by several $n$-player problems, including reliable broadcast, leader election, static $k$-selection, and $t$-resilient consensus. They provide round complexity lower bounds—and (nearly) tight upper bounds—for each of those problems. These results imply bounds on adversarial efficiency, which they analyze in terms of jamming gain and disruption-free complexity.

- **Empire of Colonies - Self-Stabilizing and Self-Organizing Distributed Algorithms** by Shlomi Dolev and Nir Tzachar.

  Self-stabilization ensures automatic recovery from an arbitrary state; the authors define *self-organization* as a property of algorithms which display local attributes. More precisely, they say that an algorithm is self-organizing if (1) it converges in sublinear time and (2) reacts “fast” to topology changes. If $s(n)$ is an upper bound on the convergence time and $d(n)$ is an upper bound on the convergence time following a topology change, then $s(n) \in o(n)$ and $d(n) \in o(s(n))$. The self-organization property can then be used for gaining, in sub-linear time, global properties and reaction to changes. They
present self-stabilizing and self-organizing algorithms for many distributed algorithms, including distributed snapshot and leader election.

The authors present a new randomized self-stabilizing distributed algorithm for cluster definition in communication graphs of bounded degree processors. These graphs reflect sensor networks deployment. The algorithm converges in $O(\log n)$ expected number of rounds, handles dynamic changes locally and is, therefore, self-organizing. Applying the clustering algorithm to specific classes of communication graphs, in $O(\log n)$ levels, using an overlay network abstraction, results in a self-stabilizing and self-organizing distributed algorithm for hierarchy definition.

Given the obtained hierarchy definition, they present an algorithm for hierarchical distributed snapshot. The algorithms are based on a new basic snap-stabilizing snapshot algorithm, designed for message passing systems in which a distributed spanning tree is defined and in which processors communicate using bounded links capacity. The combination of the self-stabilizing and self-organizing distributed hierarchy construction and the snapshot algorithm form an efficient self-stabilizer transformer. Given a distributed algorithm for a specific task, the authors are able to convert the algorithm into a self-stabilizing algorithm for the same task with an expected convergence time of $O(\log^2 n)$ rounds.


Distance-sensitivity guarantee in querying is a highly desirable property in wireless sensor networks as it limits the cost of executing a “query” operation to be within a constant factor of the distance to the nearest node that contains an answer. However, such a tight guarantee may require building an infrastructure for efficient resolution of queries, and the cost of maintaining this infrastructure may be prohibitive. Here the authors show that it is possible to implement distance-sensitive querying in an efficient way by exploiting the geometry of the network. Their querying service Glance ensures that a “query” operation invoked within $d$ distance of an event intercepts the event’s “advertise” operation within $d \ast s$ distance, where $s$ is a “stretch-factor” tunable by the user.

• **O**n Many-to-Many Communication in Packet Radio Networks by Bogdan S. Chlebus, Dariusz R. Kowalski and Tomasz Radzik.

Radio networks model wireless data communication when bandwidth is limited to one wave frequency. The key restriction of such networks is mutual interference of packets arriving simultaneously to a node. The many-
to-many (m2m) communication primitive involves \( p \) participant nodes of a distance at most \( d \) between any pair of them, from among \( n \) nodes in the network, and the task is to have all participants get to know all input messages. The authors consider three cases of the m2m communication problem. In the ad-hoc case, each participant knows only its name and the values of \( n \), \( p \) and \( d \). In the partially centralized case, each participant knows the topology of the network and the values of \( p \) and \( d \), but does not know the names of other participants. In the centralized case each participant knows the topology of the network and the names of all the participants. For the centralized m2m problem, they give deterministic protocols, for both undirected and directed networks, working in \( O(d + p) \) time, which is provably optimal. For the partially centralized m2m problem, they give a randomized protocol for undirected networks working in \( O((d + p + \log^2 n) \log p) \) time with high probability (whp), and they show that any deterministic protocol requires \( \Omega(p \log_{\log p} n + d) \) time. For the ad-hoc m2m problem, they develop a randomized protocol for undirected networks that works in \( O((d + \log p) \log^2 n + p \log p) \) time with high probability. They show two lower bounds for the ad-hoc m2m problem. One states that any m2m deterministic protocol requires \( \Omega(n \log_{\log n/d+1} n) \) time when \( n = p \Omega(n) \) and \( d > 1 \); \( \Omega(n) \) time when \( n - p = o(n) \); and \( \Omega(p \log_{\log p} n) \) time when \( d = 1 \). The other lower bound states that any m2m randomized protocol requires \( \Omega(p + d \log(n/d + 1) + \log^2 n) \) expected time.

### 2.5 Peer-to-Peer Systems

The following results were presented in this part of the conference:

- **Robust Random Number Generation for Peer-to-Peer Systems** by Baruch Awerbuch and Christian Scheideler.

  It considers the problem of designing an efficient and robust distributed random number generator for peer-to-peer systems that is easy to implement and works even if all communication channels are public. A robust random number generator is crucial for avoiding adversarial join-leave attacks on peer-to-peer overlay networks. The authors show that their new generator together with a light-weight rule recently proposed by the same authors for keeping peers well-distributed can keep various structured overlay networks in a robust state even under a constant fraction of adversarial peers.

- **About the Lifespan of Peer to Peer Networks** by Rudi Cilibrasi, Zvi Lotker, Alfredo Navarra, Stephane Perennes and Paul Vitanyi.
In this paper the authors analyze the ability of peer to peer networks to deliver a complete file among the peers. Early on they motivate a broad generalization of network behavior organizing it into one of two successive phases. According to this view the network has two main states: first centralized - few sources (roots) hold the complete file, and next distributed - peers hold some parts (chunks) of the file such that the entire network has the whole file, but no individual has it. In the distributed state they study two scenarios, first, when the peers are “patient”, i.e., do not leave the system until they obtain the complete file; second, peers are “impatient” and almost always leave the network before obtaining the complete file.

They first analyze the transition from a centralized system to a distributed one. They describe the necessary and sufficient conditions that allow this vital transition. The second scenario occurs when the network is already in the distributed state. They provide an estimate for the survival time of the network in this state, i.e., the time in which the network is able to provide all the chunks composing the file. They first assume that peers are patient and they show that if the number of chunks is much less than $e^n$, where $n$ is the number of peers in the network, then the expected survival time of the network is exponential in the number of peers. Moreover they show that if the number of chunks is greater than $\frac{\log n}{\log(e n + 1)}$, the network’s survival time is constant. This surprisingly suggests that peer to peer networks are able to sustain only a limited amount of information. They also analyze the scenario where peers are impatient and almost always leave the network before obtaining the complete file. They calculate the steady state of the network under this condition. Finally a simple model for evaluating peer to peer networks is presented.

- **Incentive-based Robust Reputation Mechanism for P2P Services by Emmanuelle Anceaume and Aina Ravoaja.**

In this paper, the authors address the problem of designing a robust reputation mechanism for peer-to-peer services. The mechanism they propose achieves high robustness against malicious peers (from individual or collusive ones) and provides incentive for participation. They show that the quality of the reputation value of trustworthy and participating peers is always better than the one of cheating and non participating ones. Finally they formally prove that, even when a high fraction of peers of the system exhibits a collusive behavior, a correct peer can still compute an accurate reputation mechanism towards a server, at the expense of a reasonable convergence time.
2.6 Mobile Agents and Robots

The following three papers deal with mobile agents (paper 1) and mobile robots (paper 2 and 3).

- **Searching for black holes with multiple agents** by Colin Cooper, Ralf Klasing, and Tomasz Radzik.

  In this paper the authors consider a fixed communication network where (software) agents can move freely from node to node along the edges. A *black hole* is a faulty or malicious node in the network such that if an agent enters this node, then it immediately “dies.” The authors are interested in designing an efficient communication algorithm for the agents to identify all black holes. They assume that they have *k* agents starting from the same node *s* and knowing the topology of the whole network. The agents move through the network in synchronous steps and can communicate only when they meet in a node. At the end of the exploration of the network, at least one agent must survive and must know the exact locations of the black holes. If the network has *n* nodes and *b* black holes, then any exploration algorithm needs $\Omega(n/k + D_b)$ steps in the worst-case, where $D_b$ is the worst case diameter of the network with at most *b* nodes deleted. They give a general algorithm which completes exploration in $O((n/k)\log n/\log \log n + bD_b)$ steps for arbitrary networks, if $b \leq k/2$. In the case when $b \leq k/2$, $bD_b = O(\sqrt{n})$ and $k = O(\sqrt{n})$, they give a refined algorithm which completes exploration in asymptotically optimal $O(n/k)$ steps.

- **Gathering Asynchronous Mobile Robots with Inaccurate Compasses** by Xavier Défago, Masafumi Yamashita and Samia Souissi.

  This paper considers a system of asynchronous autonomous mobile robots that can move freely in a two-dimensional plane with no agreement on a common coordinate system. Starting from any initial configuration, the robots are required to eventually gather at a single point, not fixed in advance (gathering problem).

  Prior work has shown that gathering oblivious (i.e., stateless) robots cannot be achieved deterministically without additional assumptions. In particular, if robots can detect multiplicity (i.e., count robots that share the same location) gathering is possible for *three or more* robots. Similarly, gathering of any number of robots is possible if they share a common direction, as given by compasses, with *no errors*.

  This work of the authors is motivated by the pragmatic standpoint that (1) compasses are error-prone devices in reality, and (2) multiplicity detection, while being easy to achieve, allows for gathering in situations with
more than two robots. Consequently, this paper focuses on gathering two asynchronous mobile robots equipped with inaccurate compasses. In particular, they provide a self-stabilizing algorithm to gather, in a finite time, two oblivious robots equipped with compasses that can differ by as much as $\pi/4$.

- **Gathering few fat mobile robots in the plane** by Jurek Czyzowicz, Leszek Gąsieniec and Andrzej Pelc.
  The authors consider the problem of autonomous identical robots represented by unit discs that move deterministically in the plane. The robots do not have any common coordinate system, do not communicate, do not have memory of the past and are totally asynchronous. Gathering such robots means forming a configuration for which the union of all discs representing them is connected. The authors solve the gathering problem for at most four robots. This is the first algorithmic result on gathering robots represented by two-dimensional figures rather than points in the plane: the authors call such robots *fat*.

### 2.7 Routing protocols, Leader Election and Self-Stabilization

The next paper deals with routing protocols.

- **Hop Chains: Secure Routing and the Establishment of Distinct Identities** by Rida A. Bazzi, Young-ri Choi and Mohamed G. Gouda.
  The authors present in this paper a secure routing protocol that is immune to Sybil attacks, and that can tolerate initial collusion of Byzantine routers, or runtime collusion of non-adjacent Byzantine routers in the absence of collusion between adjacent routers. For these settings, the calculated distance from a destination to a node is not smaller than the actual shortest distance from the destination to the node. The protocol can also tolerate initial collusion of Byzantine routers and runtime collusion of adjacent Byzantine routers but in the absence of runtime collusion between non-adjacent routers. For this setting, there is a bound on how short the calculated distance is compared to the actual shortest distance. The protocol makes very weak timing assumptions and requires synchronization only between neighbors or second neighbors. They propose to use this protocol for secure localization of routers using hop-count distances, which can be then used as a proof of identity of nodes.

The following two papers deal with the leader election problem in different contexts.
The Bulletin of the EATCS

• Computing on a Partially Eponymous Ring by Marios Mavronicolas, Loizos Michael and Paul Spirakis.

The authors study the partially eponymous model of distributed computation, which simultaneously generalizes the anonymous and the eponymous models. In this model, processors have identities, which are neither necessarily all identical, nor necessarily unique; processors receive inputs and must reach outputs that respect a relation. They focus on the partially eponymous ring $\mathbb{R}$, and they are interested in the computation of circularly symmetric relations on it.

The authors distinguish between solvability and computability: in solvability, processors must always reach outputs that respect the relation; in computability, they must reach outputs that respect the relation whenever possible, and report impossibility otherwise.

– They provide an efficient characterization of solvability of an arbitrary (circularly symmetric) relation on an arbitrary set of rings. The characterization is topological and can be expressed as a number-theoretic property that can be checked efficiently.

– They present a universal distributed algorithm for computing any arbitrary (circularly symmetric) relation on any set of rings.

Towards obtaining message complexity bounds, they derive a distributed algorithm for a natural generalization of Leader Election, in which a (non-zero) number of leaders are elected. They use this algorithm as a subroutine of their universal algorithm for collecting views; hence, they prove, as their main result, an upper bound on the message complexity of this particular instantiation of their universal algorithm to compute an arbitrary (circularly symmetric) relation on an arbitrary set of rings. The shown upper bound demonstrates a graceful degradation with the Least Minimum Base, a parameter indicating the degree of topological compatibility between the relation and the set of rings. They employ this universal upper bound to identify two interesting cases where an arbitrary relation can be computed with an efficient number of $O(|\mathbb{R}| \cdot \lg |\mathbb{R}|)$ messages: The set of rings is universal (which allows the solvability of Leader Election), or logarithmic (where each identity appears at most $\lg |\mathbb{R}|$ times).

• Self-stabilizing Leader Election in Networks of Finite-state Anonymous Agents by Michael Fischer and Hong Jiang.

This paper considers the self-stabilizing leader-election problem in a model of interacting anonymous finite-state agents. Leader election is a fundamen-
Many distributed problems are easily solved with the help of a central coordinator. Self-stabilizing algorithms do not require initialization in order to operate correctly and can recover from transient faults that obliterate all state information in the system.Anonymous finite-state agents model systems of identical simple computational nodes such as sensor networks and biological computers. Self-stabilizing leader election is easily shown to be impossible in such systems without additional structure.

An eventual leader detector $\Omega$ is an oracle that eventually detects the presence or absence of a leader. With the help of $\Omega$, uniform self-stabilizing leader election algorithms are presented for two natural classes of network graphs: complete graphs and rings. The first algorithm works under either a local or global fairness condition, whereas the second requires global fairness. With only local fairness, uniform self-stabilizing leader election in rings is impossible, even with the help of $\Omega$.

The remaining two regular papers deal with Self-Stabilization.

- **Robust Self-Stabilizing Clustering Algorithm** by Colette Johnen and Le Huy Nguyen.

  Ad hoc networks consist of wireless hosts that communicate with each other in the absence of a fixed infrastructure. Such networks cannot rely on centralized and organized network management. The clustering problem consists in partitioning network nodes into groups called clusters, giving a hierarchical organization of the network. A self-stabilizing algorithm, regardless of the initial system configuration, converges to legitimates configurations without external intervention. Due to this property, self-stabilizing algorithms tolerate transient faults. In this paper the authors present a robust self-stabilizing clustering algorithm for ad hoc network. The robustness property guarantees that, starting from an arbitrary configuration, in one round, network is partitioned into clusters. After that, the network stays partitioned during the convergence phase toward a legitimate configuration where the clusters partition ensures that any neighborhood has at most $k$ clusterheads ($k$ is a given parameter).

- **Self-Stabilizing Wireless Connected Overlays** by Vadim Drabkin, Roy Friedman and Maria Gradinariu.

  In this paper the authors propose the correctness proofs and the complexity analysis for the first self-stabilizing constructions of connected overlays for
wireless networks (eg. MANETs, WSN) based on the computation of \textit{Connected Dominating Set} (CDS). The basic idea is to construct an overlay that contains a small number of nodes, but still obtain full connectivity of the network while only relying on local exchanges of information and knowledge. They adopt two methodologies of construction: the first methodology consists of two parallel tasks, namely, computing a \textit{maximal independent set} (MIS) and then adding bridge nodes between the MIS nodes. The second methodology computes a connected dominating set using the observation that a dominator is a bridge between nodes that do not share the same neighborhood.

The proposed algorithms are fully decentralized and are designed in a self-stabilizing manner in order to cope with transient faults, mobility and nodes join/leave. In particular, they do not need to be (re)initialized after a fault or a physical topology change. That is, whatever the initial configuration is, the algorithms satisfy their specification after a stabilization period.

The convergence time of their algorithms is linear in the size of the network and they use only one extra bit of memory. They also present an optimization of their algorithms that reduces the number of nodes in the cover. However, the optimization increases the convergence time with a constant factor.

3 Brief Announcements

In addition to the regular papers three brief announcements were made at the conference.

- \textit{Conflict Managers for Self-Stabilization without Fairness Assumptions}. Authors: Maria Gradinariu and Sebastien Tixeuil.

  In this paper the authors specify the \textit{conflict manager} abstraction. Informally, a conflict manager guarantees that any two neighboring nodes cannot enter their \textit{critical section} simultaneously (safety) and that at least one node is able to execute its critical section (progress). The conflict manager problem is strictly weaker than the classical local mutual exclusion problem, where any node that requests to enter the critical section eventually does so (fairness).

  The authors argue that conflict managers are useful to transform a large class of self-stabilizing algorithms that operate in an essentially sequential model, into a self-stabilizing algorithm that operates in a completely asynchronous distributed model. The authors provide two implementations (one
deterministic, one probabilistic) of their abstraction, and provide a composition mechanism to obtain a generic transformer. Their transformers have low overhead - the deterministic transformer requires one memory bit, and guarantees time overhead in order of the network degree, the probabilistic transformer does not require extra memory. While the probabilistic algorithm performs in anonymous networks, it only provides probabilistic stabilization guarantees. In contrast, the deterministic transformer requires initial symmetry breaking but preserves the original algorithm guarantees.

- **A Provably Correct Scalable Concurrent Skip List** by Maurice Herlihy, Yossi Lev, Victor Luchangco and Nir Shavit.

  In this paper the authors propose a new concurrent skip list algorithm distinguished by a combination of simplicity and scalability. The algorithm employs optimistic synchronization, searching without acquiring locks, followed by short lock-based validation before adding or removing nodes. It also logically removes an item before physically unlinking it. Unlike some other concurrent skip list algorithms, this algorithm preserves the skip list properties at all times, which facilitates reasoning about its correctness. Experimental evidence shows that this algorithm performs as well as the best previously known algorithm under most circumstances.

- **An Algorithm for Distributing and Retrieving Information in Sensor Networks** by Hugo Miranda, Simone Leggio, Luis Rodrigues and Kimmo Raatikainen.

  Replication of data items among different nodes of a wireless infrastructureless network may be an efficient technique to increase data availability and improve access latency. This paper proposes a novel algorithm to distribute data items among nodes in these networks. The goal of the algorithm is to deploy the replicas of the data items in a way such that they are sufficiently distant from each other to prevent excessive redundancy but, simultaneously, that they remain close enough to each participant, such that data retrieval can be performed using a small number of messages. In most scenarios, the authors approach allows any node to retrieve a data item from a nearby node. The paper describes the algorithm and provides its performance evaluation for several different network configurations.

4 Social events

The conference also included several exciting social events. On the evening of December 13th all participants were invited to a cocktail reception hosted by the
On December 14th after the invited talk by Amir Pnueli the organizers had prepared an excursion into the wine country that surrounds Bordeaux. Namely they decided to invite the participants to visit the world famous village of St. Emilion. After a one hour bus ride to St. Emilion which was shrouded in an eerie and very atmospheric fog, we arrived hungry and ready at at the Le Bigarroux Restaurant in St. Emilion. An exquisite and delicious typical south-west french meal consisting of Pousse rapière & amuses bouches, Foie gras ‘a la gelée de Sauterne, Lamproie ‘a la bordelaise, Cote de boeuf grillée, Pommes sautees ‘a la graisse d’oie et cèpes, Tomate provençale, Plateau de fromage, Assortiment de desserts, was provided. The meal was so superb that the chair of the program committee, Alex Shvartsman took the opportunity to thank the local organizers - in the person of the general chair Ivan Lavallée - twice for their outstanding effort. He proclaimed it the best excursion and banquet meal OPODIS had seen so far.

After the more than three hour lunch meal we visited two wineries in St. Emilion. First the Chateau Carteau Cotes Duguay where we received an introduction to the art of wine making and were allowed to taste the chateau’s wines. Next we visited the troglodytic wine cellars of the Chateau Beausejour. These cellars were especially impressive. We were guided through a fascinating underground labyrinthian wine cellar that had been carved many, many years ago into the local limestone soil. The visit again concluded with a wine tasting. Finally we were invited to attend an initiation ceremony to the St. Emilion brotherhood in a gothic cellar in St. Emilion. In the course of this ceremony Ivan Lavallée, Michel Raynal, Alex Shvartsman and Marc Bui from the OPODIS group - based on their general merits and in particular their appreciation of good wine -were initiated into the aforementioned brotherhood. As symbol of the initiation they were invited to wear the brotherhoods colors and code of arms and received a diploma that certifies their membership. As members of the brotherhood they are now honorary citizens of St. Emilion.

After the ceremony we all happily returned to our hotels in Bordeaux.

Future OPODIS locations The 11th OPODIS, OPODIS 2007 will be held in Guadeloupe, French West Indies from December 17-20th, 2007. With its 10th edition the conference certainly has reached a mature stage and has become firmly established among the conferences that deal with principles of Distributed Computing. We look forward to it being as enjoyable scientifically as well as socially as this years conference.
THE FORMAL LANGUAGE THEORY COLUMN

by

ARTO SALOMAA

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

NINE OPEN PROBLEMS
ON CONJUNCTIVE AND BOOLEAN GRAMMARS

Alexander Okhotin*

Abstract

Conjunctive grammars are context-free grammars with an explicit conjunction operation in the formalism of rules; Boolean grammars are further equipped with an explicit negation. The paper surveys these grammars and proposes 9 most interesting and important research problems for them. An award of $240 Canadian is offered for the first correct solution of each of these problems.

1 Introduction

A context-free grammar is the most obvious formalism for specifying syntax, and actually one of the first formal objects encountered by mankind: Pāṇini’s treatise

---

*Academy of Finland, and Department of Mathematics, University of Turku, Turku, Finland. E-mail: alexander.okhotin@utu.fi. Supported by the Academy of Finland, grant 118540. This work was done while at Dept. of Romance Philology, Rovira i Virgili University, Tarragona, Spain.
on Sanskrit written around 5th century B. C. used context-free productions to specify parts of the grammar. In the mid-20th century context-free grammars were rediscovered by Chomsky [1] and by the Algol 60 committee [16], and their mathematical study began, leading to what we know as formal language theory.

The formalism of context-free grammars contains one logical operation, the disjunction, which is represented by multiple rules for a single nonterminal. This makes it easy to express the set of all strings that satisfy one of the several given syntactical conditions: two rules $A \rightarrow \alpha$ and $A \rightarrow \beta$ essentially say that whatever satisfies the condition $\alpha$ or the condition $\beta$ therefore satisfies the condition represented by the nonterminal $A$. However, any other logical operations, conjunction and negation in particular, are not expressible using context-free grammars: if we want to specify all strings that satisfy a condition $\alpha$ and at the same time another condition $\beta$, this might be impossible (as the intersection of two context-free languages is not necessarily context-free), or the grammar for such an intersection might be immensely large and completely unlike the original grammars.

These logical operations are an essential part of any formal reasoning, and being able to express them is important for any mathematical model of syntax. Extending context-free grammars to support these operations can be regarded as completing the incomplete definition of context-free grammars. Some early attempts to do this were undertaken by Latta and Wall [12] and by Heilbrunner and Schmitz [8], who proposed formalisms for specifying Boolean combinations of context-free languages. Latta and Wall, in particular, argued for the relevance of their formalism for linguistics. However, the use of conjunction and negation in these grammars was heavily restricted, and one still could not use them as freely as the disjunction.

Conjunctive grammars [17] are a natural extension of context-free grammars, which partially fills this gap: the conjunction of two syntactical conditions can be directly expressed in the form of a rule

$$A \rightarrow \alpha \& \beta$$

Boolean grammars [27] further extend conjunctive grammars by allowing an explicit negation, that is, finally, every operation of Boolean logic is directly expressible in their formalism. For instance, the set of strings that satisfy a condition $\alpha$ and at the same time do not satisfy a condition $\beta$ can be written as a rule

$$A \rightarrow \alpha \& \neg \beta$$

The language generated by a conjunctive grammar can be formally defined using derivations [17] that generalize context-free derivations, or, equivalently, by means of language equations [21]. The semantics of Boolean grammars has originally been defined using language equations [24, 27], while some alternative ap-
Boolean grammars can specify many abstract non-context-free languages, such as \{a^n b^n c^n | n \geq 0\} [17], \{ww | w \in \{a, b\}^*\} and \{a^{2n} | n \geq 0\} [27], the latter being outside of the Boolean closure of the context-free languages. Another evidence of their expressive power is given by a fairly compact grammar for the set of well-formed programs in a simple model programming language [30], which became the first specification of any programming language by a formal grammar from a computationally feasible class.

Though conjunctive and Boolean grammars have a greater expressive power than the context-free grammars, this increase in power does not lead to a complexity blowup: the languages generated by Boolean grammars are contained in $DTIME(n^3) \cap SPACE(n)$. Practical parsing techniques for context-free grammars, such as the recursive descent and the generalized LR, have been extended first to conjunctive and then to Boolean grammars [18, 19, 23, 28, 32], and the algorithms have been implemented in an ongoing research-oriented parser generator project [20]. Recently Megacz [15] started the development of a practically oriented parser generator for Boolean grammars.

Context-free grammars can be regarded as a particular case of Boolean grammars, in which the set of allowed Boolean operations is restricted to disjunction only. The studies conducted so far show that the most essential practical properties of this particular case carry on to the general case. Some theoretical properties of conjunctive and Boolean grammars were established in the first papers [17, 27]. Later a characterization of a subcase of conjunctive grammars by cellular automata was obtained [25], as well as a characterization of Boolean grammars by derivations of a special kind [29]. However, most of the theoretical problems on these grammars are still open, and solving these problems will significantly contribute to an emerging theory of Boolean grammars. The extensive theory made for the restricted case of context-free grammars deserves to be developed in the general case!

This paper proposes the most important open problems on conjunctive and Boolean grammars, which can be roughly stated as follows:

1. What languages are not generated by any Boolean grammar?
2. Are unary conjunctive languages always regular?
3. Can parsers for Boolean grammars run faster than in cubic time in general?
4. Can parsers for Boolean grammars use less than linear space in general?
5. Is the complement of every conjunctive language conjunctive as well?
6. Are there any inherently ambiguous Boolean languages?
7. Do LL($k$) Boolean grammars form an infinite hierarchy of languages?
8. Do $n$-nonterminal Boolean grammars form an infinite hierarchy of languages?
2 Definitions

2.1 Conjunctive grammars

 Conjunctive grammars are context-free grammars with an explicit conjunction operation that has semantics of intersection of languages. In addition to the implicit disjunction represented by multiple rules for a single nonterminal, which is the only logical operation expressible in context-free grammars, conjunctive grammars contain conjunction in the formalism of rules.

Definition 1. A conjunctive grammar \([17]\) is a quadruple \(G = (\Sigma, N, P, S)\), in which:

- \(\Sigma\) and \(N\) are disjoint finite nonempty sets of terminal and nonterminal symbols respectively;
- \(P\) is a finite set of grammar rules, each of the form
  \[A \rightarrow \alpha_1 \& \ldots \& \alpha_n (\text{where } A \in N, n \geq 1 \text{ and } \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*)\]
- \(S \in N\) is a nonterminal designated as the start symbol.

For every rule \(A \rightarrow \alpha_1 \& \ldots \& \alpha_n \in P\) and for every \(i (1 \leq i \leq n)\), an object \(A \rightarrow \alpha_i\) is called a conjunct.

A collection of rules for a single nonterminal can be written using the common notation

\[A \rightarrow \alpha_{i_1} \& \ldots \& \alpha_{i_m} | \ldots | \alpha_{m_1} \& \ldots \& \alpha_{m_n},\]

in which the vertical line is, in essence, disjunction.

Similarly to the context-free case, a conjunctive grammar is called linear conjunctive if every rule it contains is either of the form \(A \rightarrow u_i B_i v_i \& \ldots \& u_n B_n v_n\), where \(n \geq 1, u_i, v_i \in \Sigma^*\) and \(B_i \in N\), or of the form \(A \rightarrow w\), where \(w \in \Sigma^*\).

The semantics of conjunctive grammars is defined using derivation, generally in the same way as in the context-free case. The only difference is in the objects being transformed: while context-free derivations operate with strings over \(\Sigma \cup N\), a derivations in conjunctive grammars use terms over concatenation and conjunction.

Let us denote such terms as strings over an extended alphabet \(\Sigma \cup N \cup \{\text{"("}, \"\&\", \"\)\}\), assuming that none of the three special symbols is in \(\Sigma \cup N\). The set of valid string representations is defined inductively as follows:
1. \( \varepsilon \) is a term.
2. Every symbol from \( \Sigma \cup N \) is a term.
3. If \( A \) and \( B \) are nonempty terms, then \( AB \) is a term.
4. If \( A_1, \ldots, A_n (n \geq 1) \) are terms, then \( (A_1 & \ldots & A_n) \) is a term.

**Definition 2.** Given a grammar \( G \), define the relation \( \Rightarrow \) of immediate derivability on the set of terms:

1. A nonterminal can be rewritten by the body of some rule enclosed in parentheses, that is, for all \( s_1, s_2 \in (\Sigma \cup N \cup \{\varepsilon, \&\})^* \) and for all \( A \in N \), if \( s_1As_2 \) is a term, then, for every rule \( A \rightarrow \alpha_1 & \ldots & \alpha_n \in P \),
   \[ s_1As_2 \underset{G}{\Rightarrow} s_1(\alpha_1 & \ldots & \alpha_n)s_2 \]

2. A conjunction of several identical terminal strings enclosed in parentheses can be replaced by one such string without the parentheses, that is, for all \( s_1, s_2 \in (\Sigma \cup N \cup \{\varepsilon, \&\})^* \), for all \( w \in \Sigma^* \) and for all \( n \geq 1 \), if \( s_1(w & \ldots & w)s_2 \) is a term, then
   \[ s_1(w & \ldots & w)s_2 \underset{G}{\Rightarrow} s_1ws_2 \]

Let \( \Rightarrow^* \) be the reflexive and transitive closure of \( \Rightarrow \).

The language generated by a term \( A \) is the set of all strings over \( \Sigma \) derivable from its start symbol in a finite number of steps:

\[ L_G(A) = \{ w \mid w \in \Sigma^*, A \underset{G}{\Rightarrow}^* w \} \]

The language generated by the grammar is the language generated by the term \( S \):

\[ L(G) = L_G(S) = \{ w \mid w \in \Sigma^*, S \underset{G}{\Rightarrow}^* w \} \]

Let us construct a conjunctive grammar for the most common example of a non-context-free language.

**Example 1** ([17]). The following conjunctive grammar generates the language \( \{a^n b^n c^n \mid n \geq 0\} \):

\[
\begin{align*}
S & \rightarrow AB&DC \\
A & \rightarrow aA \mid \varepsilon \\
B & \rightarrow bBc \mid \varepsilon \\
C & \rightarrow cC \mid \varepsilon \\
D & \rightarrow aDb \mid \varepsilon
\end{align*}
\]
The grammar is based upon the representation of this language as an intersection of two context-free languages:

\[
\{a^i b^j c^k \mid n \geq 0\} = \{a^i b^j c^k \mid j = k\} \cap \{a^i b^j c^k \mid i = j\}
\]

According to this grammar, the string \(abc\) can be derived in the following way:

\[
S \Rightarrow (AB\&DC) \Rightarrow ((aA)b\&bCc\&DC) \Rightarrow ((aA)(b(b)c)&(aDb)cC) \Rightarrow ((aA)(b(b)c)&(aAb)(cC)) \Rightarrow ((a)(b(b)c)&(ab)(cC)) \Rightarrow (abc&abc) \Rightarrow abc
\]

In essence, here two context-free derivations are done in parallel, and the same string has to be derived from \(AB\) and from \(DC\) in order to do the last step of the derivation.

An important property of conjunctive grammars is that every derivation can be represented in the form of a tree with shared leaves, which generalizes context-free parse trees. The tree corresponding to the above derivation is given in Figure 1, and one can clearly see how it combines two interpretations of the same string according to two conjuncts of the rule for \(S\). A formal definition of such trees can be found in the literature [17, 19].

Another common example of a non-context-free language, \(\{wcw \mid w \in \{a, b\}^*\}\), forms a more interesting case, because, as proved by Wotschke [38], it is not expressible as a finite intersection of context-free languages. Let us give a linear conjunctive grammar for this language and explain how it works.
Example 2 ([17]). The following conjunctive grammar generates the language 
\{wcw | w ∈ \{a, b\}^*\}:

\[
S \rightarrow C&D
C \rightarrow aCa | aCb | bCa | bCb | c
D \rightarrow aA&aD | bB&bD | cE
A \rightarrow aAa | aAb | bAa | bAb | cEa
B \rightarrow aBa | aBb | bBa | bBb | cEb
E \rightarrow aE | bE | e
\]

The nonterminal C generates \{xcy | x, y ∈ \{a, b\}^*; |x| = |y|\} and thus ensures that the string consists of two equal-length parts separated by a center marker. D takes one symbol from the left and uses A or B to compare it to the corresponding symbol at the right. At the same time, D recursively refers to itself in order to apply the same rule to the rest of the string. Formally, A generates \{xcvby | x, v, y ∈ \{a, b\}^*, |x| = |y|\}, B generates \{xcvby | x, v, y ∈ \{a, b\}^*, |x| = |y|\} and therefore D produces \{uczu | u, z ∈ \{a, b\}^*\} (the last result may be obtained by a straightforward induction on the length of the string). Finally,

\{xcy | x, y ∈ \{a, b\}^*, |x| = |y| \} ∩ \{uczu | u, z ∈ \{a, b\}^*\} = \{wcw | w ∈ \{a, b\}^*\}

Let us construct a derivation of the string abcab and thus formally demonstrate that it is generated by the given grammar:

\[
S \Rightarrow (C&D) \Rightarrow ((aCh)&D) \Rightarrow ((a(bCa)b)&D) \Rightarrow ((a(b(c)a)b)&D) \Rightarrow \\
((abca(b)&D) \Rightarrow (abcab&D) \Rightarrow (abcab&(aA&aD)) \Rightarrow \\
(abcab&(a(b(c)a)b)&aD)) \Rightarrow (abcab&(a(b(c)a)b)&aD)) \Rightarrow \\
(abcab&(a(bc)a)&aD)) \Rightarrow (abcab&(a(bc)a)&aD)) \Rightarrow \\
(abcab&(abcab&abD)) \Rightarrow (abcab&(abcab&abD)) \Rightarrow \\
(abcab&(abcab&abD)) \Rightarrow (abcab&(abcab&abD)) \Rightarrow \\
(abcab&(abcab&abD)) \Rightarrow (abcab&(abcab&abD)) \Rightarrow \\
(abcab&(abcab&abD)) \Rightarrow (abcab&(abcab&abD)) \Rightarrow \\
(abcab&(abcab&abD)) \Rightarrow (abcab&(abcab&abD)) \Rightarrow \\
(abcab&(abcab&abD)) \Rightarrow (abcab&(abcab&abD)) \Rightarrow \\
(abcab&(abcab&abD)) \Rightarrow (abcab&(abcab&abD)) \Rightarrow
\]

It is important to note that the construction essentially uses the center marker, and therefore this method cannot be applied to writing a conjunctive grammar for the language \{ww | w ∈ \{a, b\}^*\}. The question of whether \{ww | w ∈ \{a, b\}^*\} can be specified by a conjunctive grammar remains an open problem.
The Bulletin of the EATCS

Let us now consider a representation of conjunctive grammars by language equations, which generalizes the well-known characterization of the context-free grammars due to Ginsburg and Rice [5].

Definition 3. For every conjunctive grammar \( G = (\Sigma, N, P, S) \), the associated system of language equations [21] is a system of equations in variables \( N \), in which each variable assumes a value of a language over \( \Sigma \), and which contains the following equation for every variable \( A \):

\[
A = \bigcup_{A \rightarrow \alpha_1 \& \ldots \& \alpha_m \in P} \bigcap_{i=1}^{m} \alpha_i \quad (\text{for all } A \in N) \quad (1)
\]

Each instance of a symbol \( a \in \Sigma \) in such a system defines a constant language \{a\}, while each empty string denotes a constant language \{\varepsilon\}. A solution of such a system is a vector of languages \( (L_C)_C \in N \), such that the substitution of \( L_C \) for \( C \), for all \( C \in N \), turns each equation (1) into an equality.

It is known that every such system has solutions, and among them the least solution with respect to componentwise inclusion, and this solution consists of exactly the languages generated by the nonterminals of the original conjunctive grammar: \((L_C)_C \in N \) [21].

This representation by language equations constitutes an equivalent semantics of conjunctive grammars, and it is this semantics, and not the fairly artificial derivation, that accounts for the intuitive clarity of conjunctive and context-free grammars.

2.2 Boolean grammars

Boolean grammars are context-free grammars augmented with all propositional connectives, or, in other words, conjunctive grammars with negation.

Definition 4. A Boolean grammar [27] is defined as a quadruple \( G = (\Sigma, N, P, S) \), where \( \Sigma \) and \( N \) are disjoint finite nonempty sets of terminal and nonterminal symbols respectively; \( P \) is a finite set of rules of the form

\[
A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n \quad (A \in N, m + n \geq 1, \alpha_i, \beta_j \in (\Sigma \cup N)^*) \quad (2)
\]

while \( S \in N \) is the start symbol of the grammar.

For each rule (2), the objects \( A \rightarrow \alpha_i \) and \( A \rightarrow \neg \beta_j \) (for all \( i, j \)) are called conjuncts, positive and negative respectively. A conjunct with an unknown sign can be denoted \( A \rightarrow \pm \gamma \), which means \( "A \rightarrow \gamma \) or \( A \rightarrow \neg \gamma" \).
A Boolean grammar becomes a conjunctive grammar if negation is never used, that is, \( n = 0 \) for every rule (2); it degrades to a standard context-free grammar if neither negation nor conjunction are allowed, that is, \( m = 1 \) and \( n = 0 \) for all rules. As in the case of conjunctive grammars, let us adopt a short notation \( A \rightarrow \varphi_1 | \ldots | \varphi_{\ell} \) for \( \ell \) rules \( A \rightarrow \varphi_i \) of the form (2) for a single nonterminal \( A \).

Intuitively, a rule (2) can be read as “if a string satisfies the syntactical conditions \( \alpha_1, \ldots, \alpha_m \) and does not satisfy any of the syntactical conditions \( \beta_1, \ldots, \beta_n \), then this string satisfies the condition represented by the nonterminal \( A \)”. This intuitive interpretation is not yet a formal definition, but this understanding is sufficient to construct grammars.

**Example 3** (cf. Example 1). The following Boolean grammar generates the language \( \{a^m b^n c^m | m, n \geq 0, m \neq n\} \):

\[
\begin{align*}
S & \rightarrow AB \& \neg DC \\
A & \rightarrow aA \mid \varepsilon \\
B & \rightarrow bBc \mid \varepsilon \\
C & \rightarrow cC \mid \varepsilon \\
D & \rightarrow aDb \mid \varepsilon
\end{align*}
\]

The rules for the nonterminals \( A, B, C \) and \( D \) are context-free, so, according to the intuitive semantics, they should generate the same languages as in Example 1. Then the propositional connectives in the rule for \( S \) specify the following combination of the conditions given by \( AB \) and \( DC \) (see Example 1):

\[
L(S) = \{a^i b^j c^k | m, n \geq 0, m \neq n\} = L(AB) \cap \overline{L(DC)}
\]

**Example 4.** The following Boolean grammar generates the language \( \{ww | w \in \{a, b\}^*\} \):

\[
\begin{align*}
S & \rightarrow \neg AB \& \neg BA \& C \\
A & \rightarrow XAX \mid a \\
B & \rightarrow XBX \mid b \\
C & \rightarrow XXC \mid \varepsilon \\
X & \rightarrow a \mid b
\end{align*}
\]

Again, according to the intuitive semantics, the nonterminals \( A, B, C \) and \( X \) should generate the appropriate context-free languages, and

\[
\begin{align*}
L(A) & = \{uav | u, v \in \{a, b\}^*, |u| = |v|\}, \\
L(B) & = \{ubv | u, v \in \{a, b\}^*, |u| = |v|\}.
\end{align*}
\]

This implies

\[
L(AB) = \{uavxb | u, v, x, y \in \{a, b\}^*, |u| = |x|, |v| = |y|\},
\]

This implies
that is, \( L(AB) \) contains all strings of even length with a mismatched \( a \) on the left and \( b \) on the right (in any position). Similarly, 

\[
L(BA) = \{ uvxay \mid u, v, x, y \in \{ a, b \}^*, |u| = |x|, |v| = |y| \}
\]

specifies the mismatch formed by \( b \) on the left and \( a \) on the right. Then the rule for \( S \) specifies the set of strings of even length without such mismatches:

\[
L(S) = L(AB) \cap L(BA) \cap \{ aa, ab, ba, bb \}^* = \{ ww \mid w \in \{ a, b \}^* \}.
\]

Though such a common-sense interpretation of Boolean grammars is clear for “reasonably written” grammars, the use of negation can, in general, lead to logical contradictions (consider the grammar \( S \rightarrow \neg S \)), and for that reason the task of defining a mathematically sound formal semantics for Boolean grammars is far from being trivial. All existing definitions of Boolean grammars [27, 39, 10] start with representing a grammar as a system of language equations with concatenation, union, intersection and complementation.

**Definition 5** (cf. Definition 3). For every Boolean grammar \( G = (\Sigma, N, P, S) \), the associated system of language equations is defined by analogy with the conjunctive case, with the following equations:

\[
A = \bigcup_{\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n \in P} \left[ \bigcap_{i=1}^m \alpha_i \cap \bigcap_{j=1}^n \beta_j \right] \tag{3}
\]

In general, systems of language equations of the form (3) have a high expressive power and the associated undecidability results [24]. The notion of a least solution is no longer useful [27]. The class of languages represented by their unique solutions is exactly the class of recursive languages, and the way these languages are represented does not well correspond to the intuitive semantics of Boolean grammars defined above. However, different restrictions upon these equations lead to feasible semantics for Boolean grammars [27, 39]. Let us define the simplest of these restrictions:

**Definition 6.** Let \( G = (\Sigma, N, P, S) \) be a Boolean grammar, let (3) be the associated system of language equations. Suppose that for every finite substring-closed language \( M \subset \Sigma^* \) (that is, for every \( w \in M \) all substrings of \( w \) are also in \( M \)) there exists a unique vector of languages \( (\ldots, L_C, \ldots)_{C \in N} \) (\( L_C \subseteq M \)), such that a substitution of \( L_C \) for \( C \), for each \( C \in N \), turns every equation (3) into an equality modulo intersection with \( M \). Then \( G \) complies to the semantics of a strongly unique solution, and, for every \( A \in N \), the language \( L_G(A) \) can be defined as \( L_A \) from the unique solution of this system. The language generated by the grammar is \( L(G) = L_G(S) \).
A simple example of a grammar deemed invalid according to Definition 6 is the grammar \( \{ S \to \neg S \land aA, A \to A \} \). Here the associated system of language equations \( \{ S = S \land aA, A = A \} \) has a unique solution \( S = A = \emptyset \): supposing \( w \in A \), a contradiction of the form “\( aw \in S \) if and only if \( aw \notin S \)” is obtained. However, this system has two solutions modulo every language \( \{ e, a, \ldots, a' \} \), namely, \( (S = \emptyset, A = \emptyset) \) and \( (S = \emptyset, A = \{ a' \}) \). This makes it invalid.

The simplest example of a grammar that does not meet the condition of this definition is \( S \to S \). However, it is valid according to alternative semantics for Boolean grammars, see Wrona [39], Kountouriotis et al. [10] and the author [27, 32]. The grammar \( S \to \neg S \) is invalid according to all known semantics. But it should be noted that, with the exception of Wrona’s semantics [39], these semantics disagree only on extremal examples and ultimately define the same family of languages. It is also possible to extend the applicability of conjunctive parse trees to Boolean grammars.

Returning to Example 4, the corresponding system of language equations is

\[
\begin{align*}
S &= AB \cap BA \cap C \\
A &= XAX \cup \{ a \} \\
B &= XBX \cup \{ b \} \\
C &= XXC \cup \{ e \} \\
X &= \{ a \} \cup \{ b \}
\end{align*}
\]

and the following assignment of languages to variables is its unique solution: \( S = \{ ww | w \in \{ a, b \}^* \} \), \( A = \{ uav | u, v \in \{ a, b \}^*, |u| = |v| \} \), \( B = \{ abv | u, v \in \{ a, b \}^*, |u| = |v| \} \), \( C = \{ aa, ab, ba, bb \}^* \), \( D = \{ a, b \} \). It is not hard to verify that the solution modulo every finite language, in the sense of the above definition, is unique, and hence \( L(G) = \{ ww | w \in \{ a, b \}^* \} \).

Despite the increased descriptive power, the theoretical upper bound for the parsing complexity for Boolean grammars is still \( O(n^3) \) [27], the same as in the context-free case, which is obtained by an extension of the Cocke–Kasami-Younger algorithm. This algorithm uses cubic time for every language generated by a Boolean grammar and on every input, and it requires that the grammar is transformed to the following extension of Chomsky normal form [27]:

**Definition 7.** A Boolean grammar \( G = (\Sigma, N, P, S) \) is in the binary normal form if every rule in \( P \) is of the form

\[
A \to B_1C_1 \& \ldots \& B_mC_m \& \neg D_1E_1 \& \ldots \& \neg D_tE_t \& \neg e \quad (m \geq 1, n \geq 0)
\]

\[
A \to a
\]

\[
S \to e \quad \text{(only if \( S \) does not appear in right-hand sides of rules)}
\]

Other algorithms that do not require grammar transformation have been proposed [28, 32, 39].
2.3 Linear conjunctive grammars and trellis automata

The family of languages defined by linear conjunctive grammars has actually been known for almost thirty years before these grammars were introduced [25]: this is the family defined by one of the simplest types of cellular automata. These are one-way real-time cellular automata, also known as *trellis automata*, studied by Dyer [4], Culik, Gruska and Salomaa [3], Ibarra and Kim [9], and others. Let us explain this concept following Culik, Gruska and Salomaa [3], who proposed it as a model of parallel computation in some electronic circuits.

A trellis automaton, defined as a quintuple $(\Sigma, Q, I, \delta, F)$, processes an input string of length $n$ using a uniform array of $n(n+1)/2$ processor nodes, as in Figure 2. Each processor computes a value from a fixed finite set $Q$. The processors in the bottom row obtain their values directly from the input symbols using a function $I: \Sigma \to Q$. The rest of the processors compute the function $\delta: Q \times Q \to Q$ of the values in their predecessors. The string is accepted if and only if the value computed by the topmost processor belongs to the set of accepting states $F \subseteq Q$.

![Figure 2: Computation done by a trellis automaton.](image)

Evidently, trellis automata are one of the simplest computational models one can imagine, and they were proved to be computationally equivalent to conjunctive grammars:

**Theorem 1** ([25]). A language $L \subseteq \Sigma^*$ is generated by a linear conjunctive grammar if and only if $L$ is recognized by a trellis automaton.

One can attain complete equivalence by extending the definition of a trellis automaton by a single bit that determines whether $\varepsilon$ is recognized or not. The proof of Theorem 1 is by an effective construction of a trellis automaton out of a grammar, and vice versa. Furthermore, the following refinement of the automaton-to-grammar construction is known:

**Theorem 2** ([26]). For every trellis automaton $M$ there exists and can be effectively constructed a two-nonterminal linear conjunctive grammar generating the same language.
The entire machinery of a trellis automaton is simulated using just one nonterminal, while the other decodes $L(M)$ from the language of the first nonterminal.

One can define linear Boolean grammars by the same restriction as for linear conjunctive grammars. These grammars actually define the same class of languages as linear conjunctive grammars and trellis automata [27].

2.4 Comparison of the families

Three families of languages have been considered. One of them, the linear conjunctive languages, turned out to be known, while the languages generated by conjunctive grammars and by Boolean grammars are new. Let us summarize the relation between these families shown in Figure 3.

![Figure 3: Hierarchy of languages.](image)

Obviously, both the context-free languages (CF) and the linear conjunctive languages (LinConj) contain the linear context-free languages (LinCF). Both inclusions are strict, which is witnessed, for instance, by the languages $\{a^m b^n a^m | m, n \geq 0\}$ and $\{a^n b^n c^n | n \geq 0\}$. It is known from Terrier [36] that CF and LinConj are incomparable: there exists a linear context-free language $L_T$, such that $L_T$ (obviously a context-free language) is not recognized by any trellis automaton. The language $L_T$ also certifies proper containment of LinConj in the conjunctive languages (Conj).

Since conjunctive grammars are a special case of Boolean grammars, the containment of one language family in the other is obvious. However, it is not known whether the inclusion is strict. Also, while it is known that the languages generated by Boolean grammars (Bool) are contained in DSPACE($n$), or, in other words, they are all deterministic context-sensitive (DetCS) [27], there is no proof of the strictness of this inclusion. Finally, it is a long-standing open problem whether DSPACE($n$) is smaller than NSPACE($n$), the latter also known as the family of context-sensitive languages (CS).
Decidability of most common decision problems for the families mentioned above, as well as the closure of the resulting families of languages under standard operations, are compared in Table 1.

\begin{table}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline
 & Reg & LinCF & CF & LinConj & Conj & Bool & DetCS & CS \\
\hline
\text{Closure properties} & + & + & + & + & + & + & + & + \\
\hline
\text{Emptiness} & + & - & - & + & + & + & + & + \\
\hline
\text{Universality} & + & + & + & - & - & - & - & - \\
\hline
\text{Equivalence} & + & + & + & + & ? & + & + & + \\
\hline
\end{tabular}
\end{table}

Table 1: Closure properties and decidability of decision problems.

3 The problems

3.1 Limitations of Boolean grammars

The limitations of the expressive power of the context-free grammars have been investigated quite well. Besides the complexity upper bounds, there exist direct techniques of proving non-context-freeness of particular languages, such as the pumping lemma and its variants, as well as Parikh’s theorem, which show that some computationally very easy languages cannot be generated by any context-free grammar. Simple examples of non-context-free languages include \( \{ a^n b^n c^n \mid n \geq 0 \} \), \( \{ wcw \mid w \in \{ a, b \}^* \} \), \( \{ a^n b^{2n} \mid n \geq 0 \} \) and \( \{ a^{3n} \mid n \geq 0 \} \).

In contrast, no methods of proving nonrepresentability of languages by Boolean grammars are currently known, and this can be regarded as the most important theoretical problem on these grammars. Note that even for conjunctive grammars no techniques for proving non-existence of grammars for particular languages are known, and only for linear conjunctive grammars one can use fairly sophisticated counting arguments developed for the trellis automaton representation [36].

Of course, the known upper bounds on the complexity of parsing for Boolean
grammars (deterministic cubic time or, using a different algorithm, deterministic linear space [27]) already imply that computationally harder languages are beyond their scope (by the time and the space hierarchies). The question is, whether any computationally easy languages cannot be specified by Boolean grammars, and how can one generally prove statements of this kind? Let us formally state this question as follows:

**Problem 1.** Are there any languages recognized by deterministic linear bounded automata working in time \(O(n^2)\) that cannot be specified by Boolean grammars?

Following is quite an interesting candidate:

\[
\{u_1 \ldots u_n \mid \text{for every } i, \text{either } u_i \in a^*c, \text{or } \exists j,k: u_i = b^k c \text{ and } u_j = a^j c\}
\]

This is an abstract language, which represents declaration of identifiers before or after their use; substrings of the form \(a^i c\) represent declarations, while every substring \(b^k c\) is a reference to a declaration of the form \(a^j c\). It is known that Boolean grammars can specify a related language, in which declarations must precede references, that is, the number \(j\) is required to be always less than \(i\) [30]. However, the corresponding grammar does not generalize for the given case.

Other languages possibly not representable by Boolean grammars can be sought for in the domain of unary languages: consider \(\{a^{2^n} | n \geq 0\}\) or \(\{a^n | n \geq 0\}\). Actually, some related nonrepresentability results are known for a certain subfamily of Boolean grammars [33]. The case of unary languages is the subject of the second problem.

### 3.2 Conjunctive grammars over a one-letter alphabet

It is well-known that context-free grammars over a one-letter alphabet generate only regular languages [5]. On the other hand, Boolean grammars can generate some nonregular unary languages, as shown by the following example:

**Example 5.** The following Boolean grammar generates the language \(\{a^{2^n} | n \geq 0\}\):

\[
\begin{align*}
S & \rightarrow A &\& \neg aA &\mid aB &\& \neg B &\mid aC &\& \neg C \\
A & \rightarrow aBB \\
B & \rightarrow \neg CC \\
C & \rightarrow \neg DD \\
D & \rightarrow \neg A
\end{align*}
\]
The rules for A, B, C and D represent the following language equation:

\[ X = aX^2 \]

This equation was studied by Leiss [13], and its unique solution was found to be \([a^n | \exists k \geq 0 : 2^k \leq n < 2^{k+1}]\). Therefore, the corresponding nonterminals of the above grammar generate the following languages:

- \(L(A) = \{a^n | \exists k \geq 0 : 2^k \leq n < 2^{k+2}\}\),
- \(L(B) = \{a^n | \exists k \geq 0 : 2^k - 1 \leq n < 2^{k+1}\}\),
- \(L(C) = \{a^n | \exists k \geq 0 : 2^{k+1} \leq n < 2^{k+2}\}\), and
- \(L(D) = \{a^n | \exists k \geq 0 : 2^{k+2} \leq n < 2^{k+3}\} \cup \{\varepsilon\}\).

Then it is easy to verify that

\[
L(S) = (L(A) \cap aL(A)) \cup (aL(B) \cap L(B)) \cup (aL(C) \cap L(C)) = \{a^{2^k} | n \geq 0\}
\]

Conjunctive grammars stand in the middle between context-free and Boolean grammars, and their expressive power in the case of a unary alphabet remains unknown. Do they generate only regular languages, like context-free grammars, or can they generate some nonregular language, like Boolean grammars? The Boolean grammar in Example 5 essentially uses negation, and it looks that there is no obvious way to replicate these constructions using conjunctive grammars. On the other hand, the regularity proof does not generalize from the context-free case, since it relies upon the pumping lemma.

**Problem 2.** Do conjunctive grammars over a one-letter alphabet generate only regular languages?

If they can generate any nonregular language, this would be a surprise. On the other hand, if only regular unary languages are generated, then some new idea would be needed for the proof. Perhaps one could first establish a pumping lemma for conjunctive grammars over a one-letter alphabet, but this seems to be not an easy task.

### 3.3 Time complexity

The membership of a given string in a context-free language can be tested in time \(\Theta(n^3)\) using the well-known Cocke–Kasami–Younger algorithm. At the same
time, some asymptotically more efficient methods of context-free recognition are known: Valiant [37] reduced context-free membership problem to matrix multiplication, which allowed him to apply Strassen’s [35] fast matrix multiplication algorithm to obtain a context-free recognizer working in time $O(n^{2.807})$. Using an asymptotically better matrix multiplication method due to Coppersmith and Winograd [2], the complexity of Valiant’s recognizer can be improved to $O(n^{2.376})$.

However, already for conjunctive grammars there seems to be no way to reduce the membership problem to matrix multiplication. Therefore, the $DTIME(n^3)$ upper bound for the complexity given by the extension of the Cocke–Kasami–Younger algorithm remains the best known, and to improve this bound one would have to invent an entirely new algorithm. The question is, can this be done?

**Problem 3.** Are the languages generated by Boolean grammars contained in $DTIME(n^{3-\varepsilon})$ for any $\varepsilon > 0$?

### 3.4 Space complexity

The Cocke–Kasami–Younger algorithm for context-free grammars uses space $\Theta(n^2)$, and its generalization for Boolean grammars fits its data in the same amount of memory [27]. In both cases this is the best known upper bound for practically useful algorithms applicable to grammars of the general form. In the context-free case it was established by Lewis, Stearns and Hartmanis [14] that it is possible to trade time for space and use as little as $O(\log^2 n)$ memory.

Adapting the method of Lewis, Stearns and Hartmanis to Boolean grammars, and even to linear conjunctive grammars, does not seem to be possible, since $O(\log^2 n)$ space complexity is achieved by a binary search in a context-free derivation, while the generation of a string by a conjunctive or a Boolean grammar is not known to have such a structurally simple representation. Also, consider that already linear conjunctive grammars can specify a P-complete language [9], which makes polylogarithmic-space recognition not very likely.

The best known upper bound for the space complexity of Boolean grammars is $O(n)$ [27]. This suggests the task of improving this result, if that is possible:

**Problem 4.** Are the languages generated by Boolean grammars contained in $SPACE(n^{1-\varepsilon})$ for any $\varepsilon > 0$?

If this were proved for any $\varepsilon > 0$, this would, in particular, separate Boolean grammars from the context-sensitive grammars. On the other hand, if the converse is the case, proving that would involve a lower bound technique interesting
3.5 Greibach normal form

A context-free grammar is said to be in Greibach normal form if every rule is either $A \rightarrow \varepsilon$, or $A \rightarrow a\alpha$ for some $a \in \Sigma$ and $\alpha \in (\Sigma \cup N)^*$. It is known from Greibach [6] that every context-free grammar can be transformed to an equivalent grammar with rules of this form.

A generalization for Boolean grammars will have rules of the form

$$A \rightarrow a\alpha_1 & \ldots & a\alpha_m \& \neg a\beta_1 \& \ldots \& \neg a\beta_n,$$

(4)

where $a \in \Sigma$, $m + n \geq 1$ and $\alpha_i, \beta_i \in (\Sigma \cup N)^*$. However, it is not known whether the family of languages generated by Boolean grammars in Greibach normal form is the same as the entire family generated by Boolean grammars. This is proposed as a research problem:

**Problem 5.** Is it true that for every Boolean grammar there exists a Boolean grammar in Greibach normal form that generates the same language?

For potentially simpler problems, consider the case of conjunctive grammars. Let us say that a conjunctive grammar is in Greibach normal form if its rules are

$$A \rightarrow a\alpha_1 \& \ldots \& a\alpha_m,$$

where $a \in \Sigma$, $m \geq 1$ and $\alpha_i \in (\Sigma \cup N)^*$. Can every conjunctive grammar be transformed to Greibach normal form?

In the same way one can define a linear conjunctive grammar in Greibach normal form, in which all rules must be of the form $A \rightarrow w$ ($w \in \Sigma^*$) or

$$A \rightarrow aB_1u_1 & \ldots & aB_mu_m,$$

where $a \in \Sigma$, $m \geq 1$ and $\alpha_i \in (\Sigma \cup N)^*$. Here it should be rather easy to prove that the linear conjunctive language $\{a^n b^{2n} \mid n \geq 1\}$ [25] cannot be generated by any linear conjunctive grammar in Greibach normal form.

The question whether the language $\{a^n b^{2n} \mid n \geq 1\}$ can be represented by a Boolean grammar in Greibach normal form might be a good starting point in approaching Problem 5. The answer to this question is likely negative, and a negative solution to the problem can be thus obtained.
3.6 Complementation of conjunctive grammars

The family of languages generated by Boolean grammars is closed under all Boolean operations and concatenation simply by virtue of having the corresponding operators as a part of the formalism. However, conjunctive grammars do not have an explicit negation operator, and the question whether for every conjunctive grammar $G$ there exists a grammar for the complement of $L(G)$ is open:

**Problem 6.** Is the family of conjunctive languages closed under complementation?

If the answer is negative, a possible witness language is the one from Example 4: the language $\{ww \mid w \in \{a, b\}^*\}$ is known to be context-free, while its complement might be nonrepresentable by conjunctive grammars.

Let us note that linear conjunctive languages are closed under complementation, which easily follows from their automaton representation [25], and can also be proved by an explicit construction [22].

3.7 Inherent ambiguity

Let us say that a Boolean grammar $G = (\Sigma, N, P, S)$ is unambiguous [34] if

1. For every nonterminal $A$ and for every string $w$ there exists at most one rule

   $$A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n,$$

   such that $w \in L_G(\alpha_1) \cap \ldots \cap L_G(\alpha_m) \cap L_G(\beta_1) \cap \ldots \cap L_G(\beta_n)$.

2. For every conjunct $A \rightarrow \pm s_1 \ldots s_\ell$ and for every string $w$ there exists at most one factorization $w = u_1 \ldots u_\ell$, such that $u_i \in L_G(s_i)$ for all $i$.

A language $L$ can be called inherently ambiguous with respect to Boolean grammars if every Boolean grammar generating it is ambiguous.

The ambiguity of the first type can be effectively eliminated by supplying every rule with an additional conjunct that expresses the condition of nonrepresentability by all other rules for this nonterminal. However, the ambiguity of the second type might be necessary to represent some languages. Is it truly necessary? This question can be stated as follows:

**Problem 7.** Do there exist any inherently ambiguous languages with respect to Boolean grammars?
There are strong reasons to expect a positive answer, that is, that there exist such languages. Note that an $O(n^2)$-time parsing algorithm for unambiguous Boolean grammars is known [34], and so if there are no inherently ambiguous languages, then the languages generated by Boolean grammars are contained in $\text{DTIME}(n^2)$ (see also Problem 3), which is an upper bound unheard of even for context-free languages of the general form. If the answer to Problem 7 is negative, this would be an extremely surprising result.

If we consider obtaining a positive answer, the languages $\{ww \mid w \in \{a, b\}^\ast\}$ and $\{a^{2n} \mid n \geq 0\}$ are possible candidates for being inherently ambiguous. Let us show that the existing grammars given in Examples 4 and 5 are ambiguous. In Example 4, consider the string $w = aabb$ and the conjunct $S \rightarrow \neg AB$: there are two factorizations $w = a \cdot abb = aab \cdot b$, such that $a \in L(A)$, $abb \in L(B)$, $aab \in L(A)$ and $b \in L(B)$. For Example 5 it is sufficient to take $w = aa$ and $A \rightarrow aBB$: there exist factorizations $w = a \cdot \varepsilon \cdot a$ and $w = a \cdot a \cdot \varepsilon$, where $\varepsilon, a \in L(B)$. But perhaps there could still exist unambiguous Boolean grammars for these languages.

As a special case of this problem, one can consider ambiguity with respect to conjunctive grammars, and investigate whether there exist inherently ambiguous languages in this sense. Let us note that all linear conjunctive languages are unambiguous already with respect to linear conjunctive grammars, which easily follows from the form of grammars obtained from trellis automata [25].

### 3.8 Hierarchy of Boolean LL($k$) languages

A certain subclass of Boolean grammars can be parsed using a generalized form of the well-known recursive descent method [18, 28].

Let $k \geq 1$ be the length of parser’s lookahead, and let us say that a Boolean grammar $G = (\Sigma, N, P, S)$ is LL($k$) if for every nonterminal $A$, for every lookahead string $u \in \Sigma^\ast$ ($|u| \leq k$), and for every sequence of conjuncts $B_0 \rightarrow \pm \eta_1 B_1 \theta_1, \ldots, B_{\ell-1} \rightarrow \pm \eta_\ell B_\ell \theta_\ell$, such that $B_0 = S$ and $B_\ell = A$, there exists at most one rule $A \rightarrow \alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n$, such that

$L_G((\alpha_1 \& \ldots \& \alpha_m \& \neg \beta_1 \& \ldots \& \neg \beta_n) \theta_1 \ldots \theta_\ell)$

contains $u$ or, if $|u| = k$, any strings that begin with $u$.

The class of Boolean LL($k$) grammars is more powerful than it seems. Following is a small and simple LL(1) Boolean grammar, which generates a P-complete language [31]; the proof of its P-completeness can be found in the cited technical report.
Example 6. The following LL(1) Boolean grammar generates a P-complete language:
\[
\begin{align*}
S & \rightarrow B \& \neg AbS \& \neg CS \\
A & \rightarrow aA \mid \varepsilon \\
B & \rightarrow aB \mid bB \mid \varepsilon \\
C & \rightarrow aCAb \mid b
\end{align*}
\]

It is known that all Boolean LL(\(k\)) grammars over a one-letter alphabet generate only regular languages [28], and therefore this subfamily is strictly weaker than the general family of Boolean grammars. The question is, do the languages generated by LL(\(k\)) Boolean grammars form an infinite hierarchy with respect to the length of the lookahead \(k\), or does this hierarchy collapse at some point \(k_0\)?

**Problem 8.** Does there exist a number \(k_0 > 0\), such that, for all \(k \geq k_0\), Boolean LL(\(k\)) grammars generate the same family of languages as Boolean LL(\(k_0\)) grammars?

To compare, for LL(\(k\)) context-free grammars, an infinite hierarchy with respect to \(k\) was established by Kurki-Suonio [11].

One can also study the intermediate case of LL(\(k\)) conjunctive grammars, which might have lesser expressive power than LL(\(k\)) Boolean grammars. In particular, no LL(\(k\)) conjunctive grammars for any P-complete languages are known.

### 3.9 Nonterminal complexity of Boolean grammars

It has long been known that \(n\)-nonterminal context-free languages form an infinite hierarchy [7], in the sense that for every \(n \geq 2\) there exists a language representable using \(n\) nonterminals and not representable using \(n - 1\) nonterminals. Consider the families of languages generated by \(n\)-nonterminal Boolean grammars, for all \(n \geq 1\). Do these families form an infinite hierarchy, or does this hierarchy collapse at some point?

**Problem 9.** Does there exist a number \(k > 0\), such that every language generated by any Boolean grammar can be generated by a \(k\)-nonterminal Boolean grammar?

For conjunctive grammars the answer to this question is also unknown. On the other hand, for linear conjunctive grammars the hierarchy collapses, as 2 nonterminals are sufficient to describe every language from this family [26].
4 Conclusion and award announcement

Context-free grammars are a very natural and important particular case of Boolean grammars, which was discovered early in the history of mankind, and which has been studied rather extensively in the last fifty years. It turns out that several key properties of this particular case, in which only disjunction is allowed, are the same in the general case of grammars with all Boolean operations. This makes the general case of Boolean grammars both theoretically and practically important. It is time to study this general case!

To promote the research on conjunctive and Boolean grammars, I decided to offer awards for the first correct solutions of any of the nine problems proposed in this paper. The award is $240 Canadian per problem, which is equally distributed between the authors of the solution. If two papers solving the same question appear simultaneously, the award is split between them. Each author receives a handwritten certificate and an award cheque. Every lady among the authors in addition receives a flower. In most cases a solution must be published in a recognized journal or presented at a recognized conference to qualify for the award.

Acknowledgements

I am grateful to Artiom Alhazov, Michael Domaratzki, Alexander Krassovitskiy, Michal Kunc, Mark-Jan Nederhof, Kai Salomaa and Detlef Wotschke for their comments on the manuscript and for discussions related to the problems. Thanks are due to Oleg Krapilsky for producing Figure 1 in 2000 A.D.; let me apologize for failing to acknowledge this until now, despite using the figure several times.

References

The Bulletin of the EATCS


THE FORMAL SPECIFICATION COLUMN

BY

HARTMUT EHRIG

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

FORMAL MODELING AND ANALYSIS OF FLEXIBLE PROCESSES IN MOBILE AD-HOC NETWORKS

Julia Padberg, Hartmut Ehrig, Kathrin Hoffmann
Technische Universität Berlin
Fakultät IV – Informatik und Elektrotechnik
padberg@cs.tu-berlin.de

Abstract

Our new project has started in October 2006 and concerns modeling and analysis of processes in mobile ad-hoc networks. In this contribution our work plan is discussed and first results are presented.

1 Introduction

Mobile Ad-hoc NETworks (MANETS) consist of mobile nodes which communicate with each other independently from a stable infrastructure, while the topology of the network constantly changes depending on the current position of the nodes and their availability. Unfortunately there are almost no approaches so far for the modeling and the analysis of those MANETS which are urgently needed for a correct course of the relevant processes.
Here, we introduce our project where we aim at modeling flexible processes that are the changing workflows at the nodes of a MANET.

It is the goal of this project to develop a formal technique which on the one hand enables the modeling of flexible processes in MANETs and on the other hand supports changes of the network topology and the transformation of processes. This can be achieved by an appropriate integration of graph transformation, Petri nets and processes in high level net classes. In this project a successful application on MANETs requires a specific advancement concerning structuring, process modeling, analysis, methodology and tool support.

The achieved results will be validated by a case study in the area of emergency management that is developed in cooperation with some research projects where an adaptive workflow management system for MANETs, specifically targeted to emergency scenarios, is partly realized resp. going to be implemented.

As a running example we use a scenario in archaeological disaster/recovery: after an earthquake, a team (led by a team leader) is equipped with mobile devices (laptops and PDAs) and sent to the affected area to evaluate the state of archaeological sites and the state of precarious buildings. The goal is to draw a situation map in order to schedule restructuring jobs. The team is considered as an overall MANET in which the team leader’s device coordinates the other team member devices by providing suitable information (e.g. maps, sensible objects, etc.) and assigning activities.

A typical cooperative process to be enacted by the team is shown in Fig. 1.a.

---

1Formal modeling and analysis of flexible processes in mobile ad-hoc networks, funded by the German Research Council (DFG) (2006 - 2008) http://tfs.cs.tu-berlin.de/projekte/formalnet/.

where the team leader has to select a building based on previously stored details of the area while team member 1 could take some pictures of the precarious buildings and team member 2 (after a visual analysis of a building) could fill in some specific questionnaires. Finally, these results have to be analyzed by the team leader in order to schedule next activities.

In a particular scenario the movement of the device equipped with the camera could result in a disconnection from the others. To maintain the network connectivity and ensuring a path among devices a layered architecture should be able to alert the mobility layer to select a possible “bridge” device (e.g., the one owned by team member 2) to follow the “going-out-of-range” camera device. In general this may result in a change of the MANET topology. Specifically, the current mobility net and the P/T-net of team member 2 have to be transformed in order to adapt it to the evolving network topology.

In [2] we introduced the paradigm "nets and rules as tokens" using a high-level model with suitable data type part. This model called algebraic higher-order (AHO) system exploits some form of control not only on rule application but also on token firing. In general an AHO-system is defined by an algebraic high-level net with system places and rule places as for example shown in Fig. 2, where the marking is given by a suitable P/T-system respectively rule on these places. For a detailed description of the data type part, i.e. the AHO-SYSMEN-signature and corresponding algebra A we refer to [2].

2 Work Plan

The goal of this project is an adequate specification technique for multi-level modeling of workflows in MANETS.
The collective task of a team using the MANET is divided into individual workflows of team members. We develop a formal technique which on the one hand enables the modeling of flexible processes in MANETS and on the other hand supports changes of the network topology and the transformation of the processes. Changes of flexible processes are induced by the need to maintain the infrastructure, by changes of the collective workflow, and – most interesting – by changes induced by individual team members.

The main tasks in the project can be grouped into this areas:

Characterization of main properties: The requirements for formal modeling of MANETS and of flexible processes in MANETS need to be analyzed exhaustively to consolidate the choice of the formalisms and to consider possible extensions from the beginning. One of the recent developments is a suitable restriction of AHO nets and the choice of reconfigurable Petri net systems [2] for the description of the flexible processes in MANETS.

Structuring and transformation of AHO nets: Tokens of AHO nets are nets and rules forming reconfigurable nets that combine the usual firing behavior with dynamic changes of the net structure given in terms of graph transformations. This combination yields a high modeling power as well as intriguing theoretic questions concerning structuring, transformation, and their compatibility with each other and their semantics. Possible choices for the underlying net formalism are place/transition (as in [2]), algebraic high-level or open nets.

Process modeling and analysis in AHO nets: Reconfigurable nets need to be developed into a well-defined and well-equipped modeling technique. This comprises a suitable semantics that supports the process intention and adequate analysis techniques as known both from the area of Petri nets, e.g. invariants, deadlocks, and the area of graph transformations, e.g. confluence, parallelism.

Methodology and (prototypes of) tool support: A methodology comprising tool environments and suitable modeling techniques is necessary for any practical application, we will provide the fundamental concepts and a prototype of tool support.

---

3http://tfs.cs.tu-berlin.de/agg/
graph transformation engine as well as a tool for the analysis of graph transformation properties like termination and rule independence. Furthermore, the token net properties could be analyzed using the Petri Net Kernel\(^4\), a tool infrastructure for Petri nets different net classes.

3 First Results

Formal modeling of workflows in MANETS using algebraic higher order nets (A\(\omega\) nets) was first introduced in [1] based on [2]. A\(\omega\) nets are Petri nets with complex tokens, namely Petri net systems as well as rules and net transformations for changing these nets. On this basis we have present a layered architecture of the model that allows the separation of support activities concerning the network from activities concerning the intended workflow in [3]. This yields better and conciser models, since supporting the network connectivity has a much finer granularity than the more or less fixed workflow execution. The layered architecture of A\(\omega\) net models of workflows in MANETS distinguishes three layers, the workflow layer, the mobility layer and the team layer. The workflow layer describes the overall workflow that is to be achieved by the whole team. The mobility layer describes the workflows in order to maintain the MANETS connectivity. The team layer describes the individual activities of the team members. Moreover, we provide a set of rules in each layer for the transformation of corresponding P/T-nets expressing different system states.

References


\(^4\)http://www2.informatik.hu-berlin.de/top/pnk/
THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

YURI GUREVICH

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

ZERO-ONE LAWS: THESAURI AND PARAMETRIC CONDITIONS

Andreas Blass∗ Yuri Gurevich

Abstract

The 0-1 law for first-order properties of finite structures and its proof via extension axioms were first obtained in the context of arbitrary finite structures for a fixed finite vocabulary. But it was soon observed that the result and the proof continue to work for structures subject to certain restrictions. Examples include undirected graphs, tournaments, and pure simplicial complexes. We discuss two ways of formalizing these extensions, Oberschelp’s (1982) parametric conditions and our (2003) thesauri. We show that, if we restrict thesauri by requiring their probability distributions to be uniform, then they and parametric conditions are equivalent. Nevertheless, some situations admit more natural descriptions in terms of thesauri, and the thesaurus point of view suggests some possible extensions of the theory.

∗Mathematics Department, University of Michigan, Ann Arbor, MI 48109–1043, U.S.A., abliss@umich.edu
Quisani: I’ve been thinking about zero-one laws for first-order logic. I know it’s a rather old topic, but I noticed something in the literature that I’d like to understand better. The first proof [5] established that, for any first-order sentence \( \sigma \) in a finite relational vocabulary \( \Upsilon \), the proportion of models of \( \sigma \) among all \( \Upsilon \)-structures with base set \{1, 2, \ldots, n\} approaches 0 or 1 as \( n \) tends to infinity. Fagin [4] rediscovered the result (with a simpler proof) and added, near the end of his paper, some remarks about what happens if, instead of considering all \( \Upsilon \)-structures, we consider only those satisfying some specified sentence \( \tau \). He pointed out that for some but not all choices of \( \tau \), there is still a 0-1 law: The proportion of models of \( \sigma \) among models of \( \tau \) with base set \{1, 2, \ldots, n\} approaches 0 or 1 as \( n \) tends to infinity. He gave two examples of such \( \tau \), both in the language with just a single binary relation symbol \( E \), the language of digraphs. One example was the sentence saying that \( \tau \) is symmetric and irreflexive, so the models are undirected loopless graphs. The other example defined the class of tournaments. The case of undirected graphs was rediscovered in [2], where another example was added, pure \( d \)-dimensional simplicial complexes, formulated using a completely symmetric and completely irreflexive \((d + 1)\)-ary relation.

I’d think there should be some general result explaining these variants of the 0-1 law.

Authors: There is such a result in Oberschelp’s paper [6], but it seems he never published the proof. His result, expressed in terms of what he calls parametric conditions, covers the variants that you mentioned as well as others, for example involving graphs with several colors of edges. It is based on the same approach, via extension axioms, as the work in [4] and many later works. So it doesn’t cover the 0-1 laws obtained by other methods, for example by Compton (see [3] and the references given there) for slowly-growing classes of structures.

Later, not knowing of Oberschelp’s work, we introduced in [1] the notion of a thesaurus as a suitable context for Shelah’s proof of the 0-1 law for choiceless polynomial time. It also provides a suitable context for the 0-1 law in the more restrictive context of first-order logic, once that logic is appropriately defined for thesauri.

Q: Does the thesaurus approach also depend on the extension axioms? And should it be combined with parametric conditions to produce a common generalization?

A: Both of your questions are answered — the first affirmatively and the second negatively — by the fact that the two approaches, parametric conditions and thesauri, are essentially equivalent, at least when applied to the case of uniform probability distributions on structures of any given size.

Q: That leaves me with a lot of questions: What are parametric conditions? What are thesauri? What exactly does “essentially equivalent” mean in this context? And what happens when the probability distributions aren’t uniform?
A: Let’s start with Oberschelp’s parametric conditions. These are conjunctions of first-order universal formulas, for a relational vocabulary, having the special form

$$\forall x_1 \ldots \forall x_k (D(\vec{x}) \rightarrow C(\vec{x})),$$

where $\vec{x}$ stands for the $n$-tuple of variables $x_1, \ldots, x_k$, where $D(\vec{x})$ is the formula $\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j$ saying that the values of these variables are distinct, and where $C(\vec{x})$ is a propositional combination of atomic formulas such that, in each atomic subformula of $C(\vec{x})$, all $k$ of the variables occur.

Q: I assume that $k$ is allowed to vary from one conjunct to another in a parametric condition, and that when $k \leq 1$ the empty conjunction $D$ is interpreted as true. So, for example, irreflexivity of a binary relation is expressed by the parametric condition $\forall x (\text{true} \rightarrow \neg R(x, x))$.

A: That’s right, and it’s easy to express the other conditions you mentioned earlier — symmetry for undirected graphs, asymmetry for tournaments, and complete irreflexivity and symmetry for simplicial complexes — as parametric conditions.

Q: I see that, but I don’t yet see the significance of the requirement that all atomic subformulas of $C$ use all the variables.

A: The simplest explanation is that if you drop this requirement then the extension axioms need not have asymptotic probability 1. For example, for almost all finite partially ordered sets, the longest chain has length three (see the proof of [3, Theorem 5.4]). So there are configurations, like a four-element chain, that can arise in partial orders but are absent with asymptotic probability 1. Traditional extension axioms, in contrast, imply that any configuration permitted by the underlying assumption $\tau$ must occur. The trouble comes from the transitivity clause in the definition of partial orders; it involves three variables but each atomic subformula uses only two of them.

Q: The example shows that some requirement is needed to eliminate the case of partial orders, but how does the “use all the variables” requirement connect with extension axioms? I guess what I’m really asking for is a sketch of Oberschelp’s proof.

A: The crucial contribution of parametricity is that it permits a reformulation of the uniform probability measure on structures of a fixed size $n$ in terms of independent choices of the truth values of instances of the relations.

Recall that, when we consider the class of all structures (of a given relational vocabulary) with universe $\{1, 2, \ldots, n\}$, the uniform probability measure on these structures can be described by saying that each instance $R(a_1, \ldots, a_r)$ (where $R$ ranges over the relations of the structure and $a$ over tuples of appropriate length from $\{1, 2, \ldots, n\}$) is independently assigned truth value true or false, with equal
probability. When we deal with, say, loopless undirected graphs, this description must be modified, since \( R(a, a) \) must be false and since \( R(a_1, a_2) \) must have the same truth value as \( R(a_2, a_1) \). Nevertheless, the uniform distribution can still be described in terms of independent flips of a fair coin: flip a coin for each 2-element subset \( \{a_1, a_2\} \) of \( \{1, 2, \ldots, n\} \) to determine both \( R(a_1, a_2) \) and \( R(a_2, a_1) \). Similarly in the case of tournaments, a single flip of a fair coin decides which one of \( R(a_1, a_2) \) and \( R(a_2, a_1) \) shall hold. And similarly in the other examples.

Something similar happens for arbitrary parametric conditions \( \tau \). To describe it, we need the notion of a \( k \)-type relative to \( \tau \). Temporarily fix a positive integer \( k \), less than or equal to the maximum arity of the relation symbols in the vocabulary of \( \tau \). Consider all the atomic formulas that use exactly the variables \( x_1, \ldots, x_k \), possibly more than once. A \( k \)-type is an assignment of truth values to these atomic formulas that makes \( C(\vec{x}) \) true whenever \( \forall \vec{x}(D(\vec{x}) \rightarrow C(\vec{x})) \) is a conjunct of \( \tau \) (up to renaming bound variables, so that \( \vec{x} \) is \( x_1, \ldots, x_k \)). In other words, a \( k \)-type is an assignment of truth values that can be realized by a \( k \)-tuple of distinct elements in a model of \( \tau \).

Now the uniform distribution on models of \( \tau \) with base set \( \{1, 2, \ldots, n\} \) admits the following equivalent description: For each \( k \) and each \( k \)-element subset \( \{a_1 < a_2 < \cdots < a_k\} \subseteq \{1, 2, \ldots, n\} \), choose, uniformly at random, a \( k \)-type to be realized by the \( k \)-tuple \( (a_1, \ldots, a_k) \). This works because each of these types is realized by \( (a_1, \ldots, a_k) \) in equally many models of \( \tau \) with base set \( \{1, 2, \ldots, n\} \) and because different increasing tuples \( (a_1, \ldots, a_k) \) behave independently. Furthermore, once the \( k \)-types of increasing tuples \( \vec{a} \) are chosen, they determine all the relations of the structure.

**Q:** What about instances of the relations where the arguments are not in increasing order?

**A:** They’re included, because the atomic formulas to which a \( k \)-type assigns truth values include those in which the variables \( \vec{x} \) occur out of order.

Once one has this alternative description of the uniform distribution on models, one can easily imitate the traditional proof of 0-1 laws. There is an extension axiom for each \( k \)-type with \( k > 0 \); it says that, for any distinct \( x_1, \ldots, x_{k-1} \), there is an \( x_k \), distinct from all of them, such that the tuple \( (x_1, \ldots, x_{k-1}, x_k) \) realizes the given \( k \)-type. It is easy to check that each extension axiom has asymptotic probability 1 and that the theory axiomatized by the extension axioms is complete. (To prove completeness, one can proceed as in [4] because the theory is \( \aleph_0 \)-categorical, or one can eliminate quantifiers as in [2].)

The role of parametricity in this argument is to ensure that all the information about any \( k \)-tuple of distinct elements can be isolated in its \( k \)-type and the \( k' \)-types of its subtuples, a finite amount of information, whose size is independent of the size \( n \) of the base set. That allows us to formulate extension axioms and
verify their asymptotic validity. Contrast this with the situation for, say, partial orders. Here the requirement of transitivity imposes correlations between a truth value $R(a, b)$ and many other truth values $R(a, c)$ and $R(b, c)$, for all $c$ in the base set. The number of relation instances correlated with a single $R(a, b)$ thus grows with the structure and the proof described above breaks down. Oberschelp [6] summarizes this (in the case of a vocabulary with only one relation symbol) by saying “A parametric property defines a class of relations which can be determined by the independent choice of values (parameters) in fixed regions of the adjacency array.”

Q: OK. I see that parametricity seems to be just what’s needed to carry out the traditional proof of the 0-1 law via extension axioms. Now what are thesauri?

A: A thesaurus is a finite set of signa, so of course we have to say what a signum is, but let’s first deal with a simplified notion of signum, which turns out to have the same generality as parametric conditions. A signum in this simplified sense consists of

- a symbol $R$ (assumed to be different for the different signa in a thesaurus),
- a natural number $j$ called the arity,
- a finite set $V$ called the value set
  \footnote{In [1], the value set was always of the form \{1, 2, \ldots, $v$\} for some positive integer $v$. Allowing arbitrary finite sets of values makes no essential difference but is technically convenient.},
- a group $G$ of permutations of \{1, 2, \ldots, $j$\}, and
- a homomorphism from $G$ to the group of permutations of $V$.

For notational convenience, one often writes simply the symbol $R$ when one really means the whole signum.

Q: That sounds pretty complicated; what’s really going on here?

A: The symbol $R$ and the arity $j$ are analogous to what you have in a relational vocabulary of ordinary first-order logic — a symbol and the number of its argument places. Our $R$’s, however, will not necessarily be 2-valued as in first-order logic, but $v$-valued, where $v$ is the cardinality of the value set $V$. So we could, for example, treat a graph with colored edges by having a single binary signum where $V$ is the set of colors plus one additional value to indicate the absence of an edge.

The other two constituents of the signum, the group $G$ and the homomorphism $h$, describe the symmetry properties that we intend $R$ to satisfy. The idea is that permuting the $j$ arguments of $R$ by a permutation $\pi$ in $G$ results in a change of the value given by $h(\pi)$. More precisely, a structure $\mathfrak{A}$ for a thesaurus consists of a
base set $A$ together with, for each signum $⟨R, j, v, G, h⟩$ (often abbreviated as just $R$), an interpretation $R^A$ assigning to each $j$-tuple of distinct elements $a_1, \ldots, a_j$ in $A$ a value $R^A(\vec{a})$, subject to the symmetry requirement

$$R^A(a_1, \ldots, a_j) = h(σ)R^A(σ(a_1), \ldots, σ(a_j)).$$

**Q:** The following variation seems more natural to me:

$$R^A(σ(a_1), \ldots, σ(a_j)) = h(π)R^A(a_1, \ldots, a_j)$$

It explains how to obtain $R^A(σ(a_1), \ldots, σ(a_j))$ from $R^A(a_1, \ldots, a_j)$.

**A:** This doesn’t work unless you either put $π^{-1}$ on one side of the equation or make $h$ an anti-homomorphism. Here’s the calculation, using your proposed variation. Let $π$ and $σ$ be two permutations in the group, let $\vec{a}$ be a $j$-tuple of elements of $A$, and let $\vec{b}$ be the $j$-tuple defined by $b_i = a_{σi}$.

$$h(π)h(σ)R^A(a_1, \ldots, a_j) = h(π)R^A(σ(a_1), \ldots, σ(a_j))$$

$$= h(π)R^A(b_1, \ldots, b_j)$$

$$= R^A(b_1, \ldots, b_j)$$

$$= R^A(σ(a_1), \ldots, σ(a_j))$$

$$= h(σ)R^A(a_1, \ldots, a_j),$$

where we’ve applied your variation three times, once with $σ$, once with $π$, and once with $σπ$. So for this to work, we’d need $h(σπ) = h(π)h(σ)$, i.e., $h$ should be an anti-homomorphism.

**Q:** I suppose using an anti-homomorphism wouldn’t be a disaster, but it would defeat the purpose of my suggestion, increased naturality.

Now why does $R^A$ apply only to $j$-tuples of distinct elements?

**A:** Distinctness is technically convenient. For example, in tournaments, one wants the truth value of $R(a, b)$ to be negated if $a$ and $b$ are interchanged, except when $a = b$. So we think of a binary relation $R$ as being given by two signa, one binary signum for distinct arguments and one unary signum for equal arguments. Similarly, a relation of higher arity would be represented by several signa, one for each way of partitioning the argument places into blocks with equal arguments.

**Q:** I see that, just as with parametric conditions, you can represent the uniform probability distributions on structures with base set $\{1, 2, \ldots, n\}$ in terms of independent random choices for some instances $R(\vec{a})$. For each signum $R$, choose a representative from each $G$-orbit of $j$-tuples of distinct elements, and assign $R$ random values at these representatives. Then propagate these assignments through the whole orbits by means of the symmetry requirement.
A: That’s right. To be precise about these $G$-orbits, one should say that $G$ acts naturally on the set of $j$-tuples of elements from any set by

$$\pi(a_1, \ldots, a_j) = (a_{\pi^{-1}(1)}, \ldots, a_{\pi^{-1}(j)}).$$

Q: With this formulation in terms of independent random choices, it should be possible to prove something analogous to extension axioms for the thesaurus context. I’d expect almost all $\Upsilon$-structures to have the following property, for each $n$:

Given any $n$ distinct points $a_1, \ldots, a_n$, there is a point $b$, distinct from all the $a_i$, and giving prescribed values for all signum instances $R^\Upsilon(a_1, \ldots, c)$ where one of the $a_i$ is $b$ and the others are distinct elements of $\{a_1, \ldots, a_n\}$.

A: That’s right, provided the prescribed values obey the symmetry requirement for thesaurus models. We’re pleased that you remembered that the arguments of a signum are supposed to be distinct, so that $b$ should occur only once in $R^\Upsilon(a_1, \ldots, c)$ and each $a_i$ should occur at most once.

Q: This result should yield 0-1 laws for thesauri, except that you haven’t yet defined first-order logic in the context of thesauri.

A: Indeed, we have not introduced a syntax to go with these semantical notions in [1], but it is not difficult to do so. Take atomic formulas to be $R(\vec{x}) = c$ where $R$ is a signum (or the symbol part of it), $\vec{x}$ is a sequence of variables of length equal to the arity of $R$, and $c \in V$. Also allow equality as usual in first-order logic. Then form compound formulas using propositional connectives and quantifiers, just as in ordinary first-order logic. The semantics is obvious. (If the values of the variables in $\vec{x}$ are not all distinct, then $R(\vec{x}) = c$ is naturally taken to be false.)

If one is willing to stretch the notion of syntax a bit, then it would be appropriate to identify the atomic formulas $R(x_1, \ldots, x_j) = c$ and $R(x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(j)}) = h(\pi)(c)$ for any $\pi$ in the group of the signum $R$, since the symmetry requirement for structures implies that these will always have the same truth value.

Once these definitions are in place, it is, as you said, not difficult to show, via extension axioms, that first-order sentences have asymptotic probabilities 0 or 1 over the class of all structures of a thesaurus.

Q: This should also be clear for another reason, once you explain how thesauri and parametric conditions are essentially equivalent. Having the 0-1 law for parametric conditions, we should be able to use the essential equivalence to deduce the 0-1 law for thesauri. But what exactly did you mean by essential equivalence?

A: Essential equivalence has several components. First, for each thesaurus $\Upsilon$, there is a parametric condition $\tau$ (in some vocabulary) such that the $\Upsilon$-structures with any particular base set (for example $\{1, 2, \ldots, n\}$) are in (natural) one-to-one correspondence with models of $\tau$ on the same base set. Second, for each first-order
sentence of the thesaurus, there is a first-order sentence of the vocabulary of \( \tau \) such that the models of these sentences match up under the correspondence above. Third, conversely, for each parametric condition \( \tau \) there is a thesaurus \( \Upsilon \) with a one-to-one correspondence as before. And fourth, for every first-order sentence of the vocabulary of \( \tau \), there is a first-order sentence of \( \Upsilon \) with the corresponding models.

Q: That seems to be exactly what’s needed in order to convert 0-1 laws from either of the two contexts to the other. So how do these correspondences work?

A: One direction is implicit in the syntax for thesauri described above. Given a thesaurus \( \Upsilon \), form a first-order vocabulary \( \Upsilon' \) with the same atomic formulas. That is, for each \( j \)-ary signum \( R \) of \( \Upsilon \) and each value \( c \), let \( R_c \) be a \( j \)-ary relation symbol in \( \Upsilon' \). The intended interpretation is that \( R_c(\bar{x}) \) should mean \( R(\bar{x}) = c \).

Every \( \Upsilon \)-structure gives, in this way, a structure (in the ordinary, first-order sense) for \( \Upsilon' \). The converse is in general false, but the collection of \( \Upsilon' \)-structures arising from \( \Upsilon \)-structures in this way can be described by a parametric condition \( \tau \). The conjuncts in \( \tau \) express the symmetry requirements of the thesaurus, i.e.,

\[
\forall \bar{x}(D(\bar{x}) \to (R_c(\bar{x}) \to R(\bar{x})((x_{\pi\square_1(i)}, \ldots, x_{\pi\square_1(j)})))
\]

for each signum \( R \) and each \( \pi \) in its group. There are also conjuncts saying that every \( j \)-tuple of distinct elements satisfies \( R_c \) for exactly one \( c \in V \) and that \( R_c \) is false whenever two of its arguments are equal. We trust this makes the first two parts of “essentially equivalent” clear.

Q: Yes; “essentially equivalent” is now half clear. But I suspect that this was the easier half. How do you handle the reverse direction?

A: Here we have to convert a parametric condition \( \tau \) into a thesaurus \( \Upsilon \). Let \( \Upsilon \) consist of one \( j \)-ary signum \( T_j \) for each \( j \) up to the maximum arity of the relation symbols in the vocabulary of \( \tau \).

Q: Just one signum per arity, no matter how rich the vocabulary of \( \tau \) is?

A: That’s right. We compensate by using a rich set of values. Take the values of the \( j \)-ary signum \( T_j \) to be the \( j \)-types relative to \( \tau \). (This is one place where it’s convenient to allow a signum to have any finite set of values, rather than only an initial segment of the positive integers as in [1].)

The group associated to the \( j \)-ary signum \( T_j \) is the symmetric group of all permutations of \( \{1, 2, \ldots, j\} \). To describe its action on the set of values, i.e., on the set of \( j \)-types, just let it act on the \( j \) variables occurring in the types. That is, the truth value assigned to an atomic formula \( \theta \) by the type \( h(\pi)(c) \) is the same as the truth value assigned by \( c \) to the formula obtained from \( \theta \) by substituting \( x_{\pi\square_1(i)} \) for \( x_i \) for all \( i \).
A structure for this thesaurus provides, for each \( j \)-tuple \( \vec{a} \) of elements of the base set, a \( j \)-type to be realized by this \( j \)-tuple. This specifies which atomic formulas are to be true of this \( \vec{a} \) and of all permutations of \( \vec{a} \). The symmetry requirement on thesaurus structures is exactly what is needed to ensure that these specified truth values for the various permutations of \( \vec{a} \) are consistent and thus describe a structure for the vocabulary of \( \tau \). Furthermore, since we use only \( j \)-types relative to \( \tau \), the resulting structures will be models of \( \tau \). And it is easy to check that every model of \( \tau \) arises from exactly one \( \Upsilon \)-structure.

Q: That takes care of the third part of essential equivalence. For the fourth part, you have to translate formulas in the vocabulary of \( \tau \) into \( \Upsilon \)-formulas. Since the syntax and semantics of thesauri treats connectives and quantifiers the same way as first-order logic does, it suffices to consider atomic formulas.

A: Right, and handling these is mainly just bookkeeping. Given an atomic formula \( \theta \) in the vocabulary of \( \tau \), let \( \{x_1, \ldots, x_k\} \) be the set of distinct free variables in it. For each equivalence relation \( E \) on this set of variables, we can write a quantifier-free formula \( \theta_E' \), in the syntax associated to the thesaurus \( \Upsilon \), with variables \( x_1, \ldots, x_k \), saying that

- the equality pattern of the \( x_i \) is given by \( E \), i.e.,
  \[
  \bigwedge_{(i,j)\in E} (x_i = x_j) \land \bigwedge_{(i,j)\not\in E} \neg(x_i = x_j),
  \]
  and

- the type realized by distinct values of \( x_i \)'s gives \( \theta \) the value true, i.e.,
  \[
  \bigvee_{c} (T_r(x_{i_1}, \ldots, x_{i_r}) = c),
  \]
  where \( x_{i_1}, \ldots, x_{i_r} \) are chosen representatives of the equivalence classes of \( E \) and where \( c \) ranges over those \( r \)-types that assign true to the formula obtained from \( \theta \) by replacing each variable by the chosen representative of its equivalence class.

Then the disjunction of these formulas \( \theta_E' \), over all equivalence relations \( E \), is satisfied by a tuple of elements \( \vec{a} \) in exactly those \( \Upsilon \)-structures that correspond to models of \( \tau \) in which \( \theta \) is satisfied by \( \vec{a} \).

Q: This bookkeeping sounds complicated, but I think I get it. Your \( \Upsilon \)-translation of \( \theta \) just says that the values of the \( x_i \)'s satisfy some pattern of equations and negated equations (an equality-type) and that the distinct ones among those values give \( T_r \) a value that corresponds to \( \theta \) being true.
A: Right. So this finishes the explanation of how parametric conditions and thesauri are essentially equivalent.

Q: Yes, except you said earlier that you were using a simplified notion of signum. What’s the full-scale notion?

A: In [1], our definition of a signum included, in addition to $R$, $j$, $V$, $G$, $h$ as above\(^2\), a probability distribution $p$ on the set of $V$ values, subject to the requirements that each value has non-zero probability and that the distribution is invariant under the group $h(G)$.

The purpose of $p$ is to modify the probability distribution on the structures with a fixed base set $\{1, 2, \ldots, n\}$. Previously, we chose a representative $j$-tuple from each $G$-orbit in $\{1, 2, \ldots, n\}^j$, and we chose the values of $R$ at these representatives uniformly at random. Now, we choose these values according to the probability distribution $p$. And then, as before, we propagate the chosen values to all the other $j$-tuples by means of the symmetry requirement for thesaurus-structures.

Q: You require that $p$ is invariant under $h(G)$ to ensure that the probability distribution on structures is independent of the choice of representative tuples.

A: Exactly. And of course the requirement that each value have non-zero probability is just a normalization. Any value whose probability is zero could be omitted, since it is unused in almost all structures.

Q: Is this more general notion of thesaurus equivalent to anything in the parametric condition world?

A: As far as we know, parametric conditions have been used only in connection with the uniform probability distribution on the structures that satisfy the conditions. But there is nothing to prevent one from introducing non-uniform distributions in this context, in a way that is equivalent to general thesauri, via the essential equivalence described above.

Q: Apart from non-uniform probabilities, are the other advantages to thinking in terms of thesauri rather than in terms of parametric conditions?

A: Yes, we see a few.

First, certain structures are naturally thought of as having multi-valued relations. For example, colored graphs, where the values of the edge relation would be the colors and “false” (the latter for where there is no edge). Similarly, a tournament in which the outcome of a game can be a tie or a win for either player seems to be naturally viewed as a three-valued binary relation on the set of players.

\(^2\)Actually, as noted in an earlier footnote, it had a number $v$ rather than a set $V$, but the difference is irrelevant.
Second, thesauri suggest a generalization that may be worth exploring. Instead of a fixed set of values for each signum, we could let the number of values grow slowly with the size of the structure.

Q: Won’t that mess up the extension axioms?
A: Not if “slowly” is taken seriously. The usual computation showing that extension axioms have asymptotic probability 1 still works if the number of values of any relation grows more slowly than $n^\varepsilon$ for each positive real number $\varepsilon$, as $n$, the number of elements in the structure, tends to infinity. So for example, $\log n$ values would be OK.

Q: What does it mean that the computation still works?
A: It means that, using the same ideas as in the proof of the usual first-order extension axioms, one finds that almost all finite models for a generalized thesaurus of this sort have the following property for each fixed natural number $k$. Take $k$ variables, say $x_1, \ldots, x_k$ and specify possible values for all the (finitely many) atomic formulas that use only these variables, at most once each, and really do use $x_k$. Here “possible values” means that

- if an atomic formula begins with the signum $R$ then the value must be in the value set of that signum, and
- if two atomic formulas begin with $R$, and their variables differ only by a permutation $\pi$ of argument places, and $\pi$ is in the group $G$ of the signum $R$, then the values must correspond via $h(\pi)$, as required in the definition of thesaurus structures.

Then, if one interprets $x_1, \ldots, x_{k-1}$ as any $k - 1$ distinct elements of the structure, there will be an interpretation for $x_k$, distinct from these, and realizing all the given values for atomic formulas.

Q: OK, so it’s really analogous to traditional extension axioms. How does the “usual computation” work in this situation.
A: Well, fix $k$ and fix an assignment of values to atomic formulas as above. We want to show that, with asymptotic probability 1, given any distinct $x_1, \ldots, x_{k-1}$, there is some $x_k$ realizing the given values.

Consider a structure of size $n$, and let $v$ be the largest of the cardinalities of the value sets, in this structure, for all the signa. Notice that the number of atomic formulas to which values are assigned is a constant $f$, because $k$ and the thesaurus are fixed. So there are at most $v^f$ possible assignments of values to these atomic formulas.

Temporarily, fix the interpretations of $x_1, \ldots, x_{k-1}$. There are $n - k + 1$ possible interpretations, distinct from these, for $x_k$. Each of these will, with the given
interpretations of $x_1, \ldots, x_k$, realize one of the $\leq v^f$ possible assignments of values to atomic formulas, so it has probability $\geq 1/v^f$ of realizing the assignment we want. Therefore, the probability that no interpretation of $x_k$ realizes our desired assignment is at most

$$\left(1 - \frac{1}{v^f}\right)^{n-k+1}.$$ 

Since $1 - t \leq e^{-t}$ for all $t$, this is at most $e^{-n/(2v^f)}$, where the factor 2 (more than) compensates for omitting $-k + 1$ once $n$ is larger than $2k$. Our assumption that $v$ grows slowly compared with $n$ means, in particular, that $2v^f < \sqrt{n}$ once $n$ is large enough. For such large $n$, we conclude that the probability that no $x_k$ realizes the desired values is $< e^{-\sqrt{n}}$.

Now un-fix the interpretation of $x_1, \ldots, x_{k-1}$. There are at most $n^{k-1}$ such interpretations, so the probability that at least one of them has no suitable $x_k$ is $< n^{k-1}e^{-\sqrt{n}}$. That’s the probability that our analog of the $k$th extension axiom fails, and the upper bound $n^{k-1}e^{-\sqrt{n}}$ approaches 0 as $n \to \infty$.

We have not studied possible applications of this idea, or even the appropriate extensions of the syntax and semantics of first-order logic. Once the number of values isn’t constant, relations seem intuitively to behave more like functions than like relations in the traditional first-order world. One could also equip the set $V$ of values with some relations or functions so that it becomes a structure in its own right.

**Q:** You’ll need some structure on $V$, at least implicitly, in order to formulate first-order sentences over such a generalized thesaurus. For ordinary thesauri, your first-order language had, in effect, names for all the values $c \in V$. But with $V$ now allowed to grow, that would make the language vary with the structure — not a good idea if you want to talk about the asymptotic probability of fixed sentences as the structure grows.

**A:** The idea could still work if, as the structure grows, the language also grows, so that any fixed sentence would make sense in all sufficiently large structures. But in fact, in at least one situation, $V$ is naturally a relational structure.

**Q:** What situation is that?

**A:** Consider a tournament in which the result of each game (i.e., the relationship between a pair of players) is not merely a win for one or the other (as in traditional tournaments) or a tie (as in a generalization mentioned above) but can be any one of several “degrees of victory”, where these degrees come from a small, linearly ordered set $V$. As before, “small” should mean of cardinality $< n^\varepsilon$ for each fixed $\varepsilon > 0$ and all large $n$. The outcomes of the games are modeled by a $V$-valued function $R$ subject to the (anti-)symmetry requirement that $R(x, y)$ and $R(y, x)$ are
The Bulletin of the EATCS

symmetrically located in $V$, i.e., each is the image of the other under the unique order-reversing permutation of $V$.

In this situation, it is natural to admit the linear ordering of $V$ as part of the structure, so that there are atomic formulas like $R(x, y) < R(u, v)$.

Q: Do you also want to allow names for the elements of $V$? If so, then it seems that your suggestion of a growing language depends on making some arbitrary conventions here. How shall the interpretation of a particular name vary while $n$ and therefore $V$ grow? There would be no problem with names for the first element, the second, the last, etc., but what about names for the element one third of the way up, or the element in position $\lfloor \sqrt{|V|} \rfloor$?

A: For the purposes of the present discussion, we’d want a language for which the 0-1 law holds. Apart from that, the choice of language would be guided by potential applications.

Notice, though, that as long as we have the ordering relation on $V$ available, names for the first element, the second, the last, etc. are not really needed, with asymptotic probability 1, because these elements are definable in terms of the ordering.

Q: Wait a minute. Those definitions use quantifiers. Do you intend to allow quantification over $V$ in your language?

A: With asymptotic probability 1, we have quantification over $V$ automatically, since the elements of $V$ are almost surely the same as the values of $R$. So, for example, we can almost surely express “$R(x, y)$ is the first element of $V$” by the formula

$$\forall u \forall v \neg(R(u, v) < R(x, y)).$$

Q: I get it. This formula doesn’t express the desired property in all structures, but it does in those structures where, as $u$ and $v$ range over the players, $R(u, v)$ ranges over all elements of $V$ — and that’s almost all structures.

A: Such expressive power requires some caution in the definition of structures, specifically in the choice of how $V$ is allowed to grow. For example, we had better insist that the cardinality of $V$ has the same parity in almost all structures, because that parity is definable by the truth value of the sentence

$$\exists x \exists y (R(x, y) = R(y, x)).$$

Fortunately, this parity issue turns out to be the only such problem, in this particular example. That is, if we restrict $|V|$ to have a fixed parity and to grow slowly with $n$, then there will be a 0-1 law for the first-order properties of tournaments of this sort. (Here the atomic sentences are of the forms $x = u$ and $R(x, y) < R(u, v)$,
where $x$ and $y$ are distinct variables and $u$ and $v$ are distinct variables, but other pairs of variables might coincide. It would make no difference if we also allow $R(x, y) = R(u, v)$, since it can be defined in terms of $\prec$. We’ll put a sketch of the argument into an appendix of this paper.

There is another possibility suggested by thesauri, namely the possibility of playing with the groups in the signa, and perhaps bringing some group theory to bear on these topics.

Q: That reminds me of something that occurred to me while you were proving the equivalence between thesauri and parametric conditions. Although thesauri allow arbitrary groups of permutations of $\{1, 2, \ldots, j\}$ for a $j$-ary signum, you used only the full symmetric group in the thesauri that simulate given parametric conditions. Combining this with the simulation in the other direction, you’ve shown, in effect, that any thesaurus is equivalent to one in which all the groups involved in the signa are full symmetric groups. I’d think this equivalence, involving only thesauri, should have a proof that involves only thesauri, rather than going from thesauri to parametric conditions and back.

A: Yes, one can also obtain the same result by working only with thesauri. Let $\langle R, j, V, G, h \rangle$ be a signum, and let $S$ be the full symmetric group of all permutations of $\{1, 2, \ldots, j\}$. Then the signum $\langle R, j, V, G, h \rangle$ is essentially equivalent to a signum $\langle R_+, j, V_+, S, h_+ \rangle$, where of course we have to define $V_+$ and $h_+$ (not $R_+$ because it’s just a label).

Q: Before you start defining $V_+$ and $h_+$, please tell me what “essentially equivalent” means here.

A: It means that there is, for each set $A$, a canonical bijection between the possible interpretations in $A$ for the two signa.

Q: OK. Now what are $V_+$ and $h_+$.

A: $V_+$ is the set of functions $f : S \to V$ that are $G$-equivariant in the sense that, for each $\pi \in G$ and each $\sigma \in S$,

$$f(\pi \sigma) = h(\pi)(f(\sigma)).$$

The action $h_+$ of $S$ on $V_+$ is given by

$$(h_+(\sigma)(f)) (\tau) = f(\tau \sigma).$$

It is straightforward to calculate that $h_+$ is a homomorphism\(^3\) from $S$ into the group of permutations of $V$.

\(^3\)Category theorists would describe the operation sending any $G$-set $(V, h)$ to the $S$-set $(V_+, h_+)$ as the right adjoint of the forgetful functor from $S$-sets to $G$-sets.
The Bulletin of the EATCS

Q: On the basis of my recent experience, I suppose that, if you had written $\sigma \tau$ instead of $\tau \sigma$, then $h_+$ would have been an anti-homomorphism.

A: That’s right.

The essential equivalence between the two signs is obtained as follows. Given an interpretation $R^\mathbb{G}$ of the original signum, i.e., a $G$-equivariant map $A^j \to V$, we obtain an interpretation $R'^G$ of the new signum, an $S$-equivariant map $A^j \to V$, by letting $R'^G(A^j(\vec{a}))$ be the element of $V$ defined by

$$R'^G(A^j(\vec{a})) = R^\mathbb{G}(a_{\sigma^{-1}(1)}, \ldots, a_{\sigma^{-1}(j)}).$$

The transformation in the other direction takes any $S$-equivariant map $R'_G : A^j \to V$ to the $G$-equivariant map $R^\mathbb{G} : A^j \to V$ defined by

$$R^\mathbb{G}(A^j(\vec{a})) = (R'_G(\vec{a}))(1),$$

where 1 is the identity permutation. Of course, there’s a lot to be checked here: that the two transformations are well-defined and that they’re inverse to each other. That should be a good exercise for you.

Q: OK, I’ll check it when I have some spare time. But, in view of this equivalence, why did you define thesauri in terms of arbitrary permutation groups $G$ rather than just the full symmetric group?

A: We weren’t aware of the equivalence until recently. But it seems worthwhile to retain the generality of arbitrary groups, since some forms of symmetry are more naturally expressed using groups smaller than the full symmetric group. Notice also that the conversion from a thesaurus using arbitrary groups to one using full symmetric groups makes the value sets $V$ considerably larger and more complicated, especially if the original groups were small.
Appendix

We outline here the proof of the 0-1 law for the generalized tournaments described above. In fact, we prove somewhat more, namely the 0-1 law for first-order sentences in certain two-sorted structures. Here is the precise formulation.

Define (for the purposes of this appendix only) a generalized tournament to be a finite two-sorted structure $\mathcal{A} = (A, V, <, \pi, R)$, where $A$ and $V$ interpret the two sorts, where $<$ is a linear ordering of $V$, where $\pi$ is the unique order-reversing permutation of $V$, and where $R$ is a function $A^2 \to V$, subject to the requirements that

- $R(a, b) = \pi(R(b, a))$ for all $a, b \in A$ and
- $|V|$ is odd.

The main reason for the second of these requirements is that, as we saw above, the parity of $|V|$ is definable and must therefore be fixed (at least almost surely) if we are to have a 0-1 law. We chose “odd” rather than “even” so that the first requirement makes sense even when $a = b$. For even $|V|$, a similar argument would apply, but we would have to either make $R$ a partial function defined only when the arguments are distinct, or modify the first requirement to accommodate some convention for the case of equal arguments.

We say that a generalized tournament $\mathcal{A}$ as above has size $(n, v)$ if $|A| = n$ and $|V| = v$. For a first-order sentence $\theta$ in the language of generalized tournaments, we define the $(n, v)$-probability of $\theta$ as the proportion of models of $\theta$ among generalized tournaments of size $(n, v)$. We say that $\theta$ has asymptotic probability 1 if, for each $\varepsilon > 0$, there exist $M \in \mathbb{N}$ and $\delta > 0$ such that, whenever $M < v < n^\delta$, then the $(n, v)$-probability of $\theta$ is at least $1 - \varepsilon$. Note that our definition of asymptotic probability incorporates the requirement that $v$ is small compared to $n$, namely $v < n^\delta$; just how small this is (i.e., the choice of $\delta$) depends on $\theta$ and $\varepsilon$. This definition of asymptotic probability is intended specifically for use with generalized tournaments; we do not propose it for more general contexts, though we expect that something similar would be appropriate in greater generality.

**Theorem 1.** Let $\theta$ be any sentence in the first-order language of generalized tournaments. Then one of $\theta$ and $\neg \theta$ has asymptotic probability 1.

**Proof.** Consider the first-order theory $\mathcal{RGT}$ (which stands for “random generalized tournaments”), in the language of generalized tournaments, given by the following axioms.

- $R$ is a binary function from $A$ to $V$.
- $<$ is a linear ordering of $V$.
The Bulletin of the EATCS

- $V$ has a first element and a last element.
- Each element of $V$ but the first (resp. last) has an immediate predecessor (resp. successor).
- $V$ is infinite (i.e., infinitely many axioms, saying $|V| > k$ for each $k \in \mathbb{N}$).
- $\pi$ is an order-reversing permutation of $V$.
- $R(x, y) = \pi(R(y, x))$.
- $\pi$ has a fixed point.
- The extension axioms

$$\forall \vec{x} \forall \vec{u} \left[ D(\vec{x}) \rightarrow (\exists y) \bigwedge_{i=1}^{k} (y \neq x_i \land R(x_i, y) = u_i) \right]$$

where the variables $\vec{x} = x_1, \ldots, x_k$ and $y$ range over the first sort and $\vec{u} = u_1, \ldots, u_k$ over the second. As before, $D(\vec{x})$ is the formula saying that all the components $x_i$ are interpreted as distinct elements.

We computed above that each extension axiom has asymptotic probability 1. Each of the axioms saying $|V| > k$ has asymptotic probability 1; just take $M > k$ in the definition of asymptotic probability. The remaining axioms of $\mathcal{RG}T$ have asymptotic probability 1 for the trivial reason that they are true in all generalized tournaments.

Just as with the usual (one-sorted) notion of asymptotic probability, one sees that any conjunction of finitely many sentences of asymptotic probability 1 also has asymptotic probability 1 (if there are $k$ conjuncts and if $\epsilon > 0$ is given, then just take the largest of the relevant $M$’s and the smallest of the $\delta$’s for the conjuncts, using $\epsilon/k$ in place of $\epsilon$) and that any logical consequence of a sentence of asymptotic probability 1 also has asymptotic probability 1. Therefore, the theorem will follow if we can show that the theory $\mathcal{RG}T$ is complete.

We prove completeness by presenting a winning strategy for Duplicator in an Ehrenfeucht-Fraïssé game of an arbitrary but specified length (i.e., number of rounds) $r$, between two models of $\mathcal{RG}T$. The essential part of Duplicator’s work takes place in $V$. Here, Duplicator uses a familiar strategy for discrete linear orders with endpoints. The involution $\pi$ and the $A$ of the models are handled by suitable bookkeeping devices, described below in terms of imaginary pebbles.

For convenient reference, we first summarize the usual strategy for the Duplicator in the $r$-round Ehrenfeucht-Fraïssé game on two discrete linear orders without endpoints. In order to win, Duplicator must (by definition of the game)
ensure that corresponding pebbles are ordered the same way in both models. His winning strategy is to ensure that, in addition, the distances between corresponding pebbles are not too different. Specifically, if, after \( m \) moves, two pebbles are at a distance \( \leq 2^{-m} \) on one model, then the corresponding pebbles on the other model are the same distance apart. Distances greater than \( 2^{-m} \) need not be matched exactly, though of course if the distance between two pebbles is \( > 2^{-m} \) on one model then the distance between the corresponding pebbles on the other model will also be \( > 2^{-m} \). The key to the proof is that, because the distance \( 2^{-m} \), below which matching is required, decreases by a factor 2 from each move to the next, Duplicator can always move so as to maintain the required matching. By doing so, he ensures that he wins the game.

For infinite discrete linear orders with endpoints, Duplicator’s strategy is the same except that he pretends that, already at the start of the game, there is a pair of corresponding pebbles on the first elements of the models and another pair of corresponding pebbles on the last elements. (In fact, the strategy doesn’t really need that the models are infinite; it suffices that they have large enough cardinalities so that the Duplicator’s matching requirements are satisfied by the initial imaginary pebbles.)

With these preliminaries, we now present Duplicator’s strategy for models of \( \mathcal{RGT} \). It involves a more elaborate scheme of imaginary pebbles. At the start of the game, Duplicator should imagine pebbles already placed on the first, last, and middle elements of the \( V \) sort in both models. Here “middle element” means the unique element fixed by \( \pi \). In addition, during the course of the game, whenever a pebble is on an element \( q \in V \), Duplicator should imagine an associated pebble at \( \pi(q) \). Furthermore, whenever pebbles are on two elements \( a \) and \( b \) of \( A \), Duplicator should imagine an associated pebble at \( R(a, b) \).

As in the proof for plain linear orderings, Duplicator must ensure that corresponding pebbles on the \( V \) sorts of the two models are ordered the same way, and he will, in addition, voluntarily ensure that sufficiently small distances between corresponding pebbles match exactly. But now, “sufficiently small” does not mean \( \leq 2^{-m} \) (after \( m \) rounds of an \( r \)-round game) but rather \( \leq 2^{2^{-m}} \). The reason for this change will become clear in a moment, but for now notice that, like \( 2^{-m} \), this new boundary of smallness, \( 2^{2^{-m}} \), decreases by at least a factor 2 at every move.

It is clear that, if Duplicator can follow this strategy, then doing so guarantees that he wins. It remains to show that Duplicator always has a move that maintains the required matching of distances. The proof of this is in two cases, depending on which sort Spoiler adds a pebble to.

Suppose first that Spoiler puts a new pebble on the \( V \) sort of one of the two models. Then, exactly as in the proof for plain linear orders, Duplicator can find a suitable spot for the corresponding pebble on the other model, because the dis-
The Bulletin of the EATCS

tance below which matching is required has decreased by at least a factor 2. The
new pebbles played in this round give rise to a new pair of imaginary pebbles,
located at \( \pi \) of the real pebbles. But the required matching of distances between
these imaginary pebbles and the played pebbles at earlier moves is automatic be-
cause \( \pi \) preserves distances. We must also consider the distance between the new
real pebble and the new imaginary pebble, but since these locations are related by
\( \pi \) they lie on opposite sides of the middle of \( V \). Their distance is twice the dis-
tance from either of them to the (initially imagined) pebble at the middle. Since
the distance to the middle pebble is matched (if small), so is the distance between
the real and imaginary new pebbles. This completes the proof for the case where
Spoiler’s new pebble is on the \( V \) part.

Now suppose that Spoiler puts a new pebble on the \( A \) sort of one of the models.
This results in many new imaginary pebbles on the \( V \) part. If there were already
\( m \) moves before the current one, then there could be as many as \( m \) pebbles already
in the \( A \) part, say at elements \( a_1, \ldots, a_m \). The new pebble, say at \( b \), gives rise to as
many as \( 2m \) new imaginary pebbles, at the elements \( R(a_i, b) \) and (their \( \pi \)-images)
\( R(b, a_i) \). In this situation, Duplicator should proceed as follows. Pretend that,
instead of a single move of Spoiler producing all these imaginary pebbles, there
were \( m + 1 \) moves, that during the first \( m \) of these moves Spoiler put pebbles on
the elements \( R(a_i, b) \) one at a time (resulting in imaginary pebbles at \( R(b, a_i) \)), and
that at the last of the \( m + 1 \) moves Spoiler put the (real) pebble at \( b \). Let Duplicator
pretend to respond to the first of these \( m \) moves as described in the preceding
paragraph, i.e., let him put imaginary new pebbles in the appropriate places. At
each step, the maximum distance below which he can maintain exact matching
decreases by a factor 2, so in these \( m \) moves, it has decreased by \( 2^m \). Fortunately,
this is still large enough, because \( 2^{r^2 - m^2} \) (from before these moves) decreased by
a factor \( 2^m \) is still larger than \( 2^{r^2 -(m+1)^2} \), the required distance for matching after
these moves. (This is of course why we used \( 2^{r^2 - m^2} \) rather than \( 2^{r^2 - m} \). Our choice
is not the smallest, since it could accommodate a decrease by a factor \( 2^{2m+1} \), but
it seems to be the simplest.) Finally, after all the imaginary pebbles have been
placed in \( V \), Duplicator must find a place for his real pebble in \( A \). It together
with previous pebbles must produce, as values of \( R \), the elements of \( V \) bearing
the imaginary pebbles just placed. Fortunately, an extension axiom guarantees the
existence of an appropriate element to receive this pebble.

Bibliographic Remark: Two useful survey papers on 0-1 laws, Oberschelp’s
[6] and Compton’s [3], were not treated kindly by the printing process. The title
of [3], though given correctly in the table of contents of the book in which it
appears, had “0-1” deleted on the first page of the paper itself. As a result, “0-1”
is also missing in the MathSciNet entry and perhaps elsewhere. In [6], some of the
pages were printed out of order, and numbered in the printed order rather than the
intended order. Fortunately, all the pages are present and can be found in correct order by a local search.

References


THE NATURAL COMPUTING COLUMN

BY

GRZEGORZ ROZENBERG

Leiden University, Leiden Center for Natural Computing
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
rozenber@liacs.nl

SPIKING NEURAL P SYSTEMS: A TUTORIAL

Gheorghe PĂUN

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 București, Romania
Email: george.pau@imar.ro, gpaun@us.es

Abstract. We briefly present (basic ideas, some examples, classes of spiking neural P systems, some results concerning their power, research topics) a recently initiated branch of membrane computing with motivation from neural computing. Further details can be found at the web page of membrane computing, from http://psystems.disco.unimib.it.

1 The General Framework

The most intuitive way to introduce spiking neural P systems (in short, SN P systems) is by watching the movie available at http://www.igi.tugraz.at/tnatschl/spike_trains_eng.html, on the web page of Wolfgang Maass at Graz, Austria: neurons are sending to each others spikes, electrical impulses of identical shape (duration, voltage, etc.), with the information “encoded” in the frequency of these impulses, hence in the time passes between consecutive spikes. For neurologists, this is nothing new, related drawings already appears in papers by Ramón y Cajal, a pioneer of neuroscience at the beginning of the last century, but in the recent years “computing by spiking” is a vivid research area, with the hope to lead to a neural computing “of the third generation” – see [12], [21], etc.
For membrane computing it is somehow natural to incorporate the idea of spiking neurons (already neural-like P systems exist, based on different ingredients – see [23], efforts to compute with a small number of objects were recently made in several papers – see, e.g., [2], using the time as a support of information, for instance, taking the time between two events as the result of a computation, was also considered – see [3]), but still important differences exist between the general way of working with multisets of objects in the compartments of a cell-like membrane structure – as in membrane computing – and the way the neurons communicate by spikes. A way to answer this challenge was proposed in [18]: neurons as single membranes, placed in the nodes of a graph corresponding to synapses, only one type of objects present in neurons, the spikes, with specific rules for handling them, and with the distance in time between consecutive spikes playing an important role (e.g., the result of a computation being defined either as the whole spike train of a distinguished output neuron, or as the distance between consecutive spikes). Details will be given immediately.

What is obtained is a computing device whose behavior resembles the process from the neuron nets, meant to generate strings or infinite sequences (like in formal language theory), to recognize or translate strings or infinite sequences (like in automata theory), to generate or accept natural numbers, or to compute number functions (like in membrane computing). Results of all these types will be mentioned below. Nothing is said here, because nothing was done so far, about using such devices in “standard” neural computing applications, such as pattern recognition. Several open problems and research topics will be mentioned below (a long list of such topics, prepared for the Fifth Brainstorming Week on Membrane Computing, Sevilla, January 29-February 2, 2006, can be found in [24]), but probably this is the most important one: connecting SN P systems with neural computing, more generally, looking for applications of SN P systems.

It is worth mentioning here that “general” membrane computing is now an area of intense research related to applications, mainly in biology/medicine, but also in economics, distributed evolutionary computing, computer graphics, etc. (see [11], [27], [28]), but this happens after a couple of years of research of a classic language-automata-complexity type; maybe this will be the case also for the spiking neural P systems, which need further theoretical investigation before passing to applications.

2 An Informal Overview – With An Example

Very shortly, an SN P system consists of a set of neurons (cells, consisting of only one membrane) placed in the nodes of a directed graph and sending signals (spikes, denoted in what follows by the symbol $a$) along synapses (arcs of the
The Bulletin of the EATCS

graph). Thus, the architecture is that of a tissue-like P system, with only one kind of objects present in the cells. The objects evolve by means of spiking rules, which are of the form $E[a^c \rightarrow a; d]$, where $E$ is a regular expression over $\{a\}$ and $c,d$ are natural numbers, $c \geq 1, d \geq 0$. The meaning is that a neuron containing $k$ spikes such that $a^k \in L(E)$, $k \geq c$, can consume $c$ spikes and produce one spike, after a delay of $d$ steps. This spike is sent to all neurons to which a synapse exists outgoing from the neuron where the rule was applied. There also are forgetting rules, of the form $a^s \rightarrow \lambda$, with the meaning that $s \geq 1$ spikes are removed, provided that the neuron contains exactly $s$ spikes. We say that the rules “cover” the neuron, all spikes are taken into consideration when using a rule.

The system works in a synchronized manner, i.e., in each time unit, each neuron which can use a rule should do it, but the work of the system is sequential in each neuron: only (at most) one rule is used in each neuron. One of the neurons is considered to be the output neuron, and its spikes are also sent to the environment. The moments of time when a spike is emitted by the output neuron are marked with 1, the other moments are marked with 0. The binary sequence obtained in this way is called the spike train of the system – it might be infinite if the computation does not stop.

Figure 1 recalls an example from [18], and this also introduces the standard way to represent an SN P system (note that the output neuron, $\sigma_7$ in this case, is indicated by an arrow pointing to the environment), and a simplification in writing the spiking rules: if we have a rule $E[a^c \rightarrow a; d]$ with $L(E) = \{a^c\}$, then we write simply $a^c \rightarrow a; d$. If all rules are of this form, then the system is called bounded (or finite), because it can handle only finite numbers of spikes in the neurons.

In the beginning, only neurons $\sigma_1, \sigma_2, \sigma_3$, and $\sigma_7$ contain spikes, hence they fire in the first step – and spike immediately. In particular, the output neuron spikes, hence a spike is also sent to the environment. Note that in the first step we cannot use the forgetting rule $a \rightarrow \lambda$ in $\sigma_1, \sigma_2, \sigma_3$, because we have more than one spike present in each neuron.

The spikes of neurons $\sigma_1, \sigma_2, \sigma_3$ will pass to neurons $\sigma_4, \sigma_5, \sigma_6$. In step 2, $\sigma_1, \sigma_2, \sigma_3$ contain no spike inside, hence will not fire, but $\sigma_4, \sigma_5, \sigma_6$ fire. Neurons $\sigma_5, \sigma_6$ have only one rule, but neuron $\sigma_4$ behaves non-deterministically, choosing between the rules $a \rightarrow a; 0$ and $a \rightarrow a; 1$. Assume that for $m \geq 0$ steps we use here the first rule. This means that three spikes are sent to neuron $\sigma_7$, while each of neurons $\sigma_1, \sigma_2, \sigma_3$ receives two spikes. In step 3, neurons $\sigma_4, \sigma_5, \sigma_6$ cannot fire, but all $\sigma_1, \sigma_2, \sigma_3$ fire again. After receiving the three spikes, neuron $\sigma_7$ uses its forgetting rule and gets empty again. These steps can be repeated arbitrarily many times.

In order to have neuron $\sigma_7$ firing again, we have to use sometimes the rule $a \rightarrow a; 1$ of neuron $\sigma_4$. Assume that this happens in step $t$ (it is easy to see that $t = 2m + 2$). This means that at step $t$ only neurons $\sigma_5, \sigma_6$ emit their spikes. Each
of neurons \(\sigma_1, \sigma_2, \sigma_3\) receives only one spike – and forgets it in the next step, \(t+1\). Neuron \(\sigma_7\) receives two spikes, and fires again, thus sending the second spike to the environment. This happens in moment \(t + 1 = 2m + 2 + 1\), hence between the first and the second spike sent outside have elapsed \(2m + 2\) steps, for some \(m \geq 0\). The spike of neuron \(\sigma_4\) (the one “prepared-but-not-yet-emitted” by using the rule \(a \rightarrow a; 1\) in step \(t\)) will reach neurons \(\sigma_1, \sigma_2, \sigma_3\), and \(\sigma_7\) in step \(t + 1\), hence it can be used only in step \(t + 2\); in step \(t + 2\) neurons \(\sigma_1, \sigma_2, \sigma_3\) forget their spikes and the computation halts. The spike from neuron \(\sigma_7\) remains unused, there is no rule for it. Note the effect of the forgetting rules \(a \rightarrow \lambda\) from neurons \(\sigma_1, \sigma_2, \sigma_3\): without such rules, the spikes of neurons \(\sigma_5, \sigma_6\) from step \(t\) will wait unused in neurons \(\sigma_1, \sigma_2, \sigma_3\) and, when the spike of neuron \(\sigma_4\) will arrive, we will have two spikes, hence the rules \(a^2 \rightarrow a; 0\) from neurons \(\sigma_1, \sigma_2, \sigma_3\) would be enabled again and the system will continue to work.

Let us return to the general presentation. In the spirit of spiking neurons, in the basic variant of SN P systems introduced in [18], the result of a computation is defined as the distance between consecutive spikes sent into the environment by the (output neuron of the) system. In [18] only the distance between the first two spikes of a spike train was considered, then in [25] several extensions were examined: the distance between the first \(k\) spikes of a spike train, or the distances between all consecutive spikes, taking into account all intervals or only intervals that alternate, all computations or only halting computations, etc.

Therefore, as seen above, the system \(\Pi\) from Figure 1 computes the set \(N_2(\Pi) = \{2n \mid n \geq 1\}\) – where the subscript 2 reminds that we consider the distance between the first two spikes sent to the environment.
Systems working in the accepting mode were also considered: a neuron is designated as the input neuron and two spikes are introduced in it, at an interval of \( n \) steps; the number \( n \) is accepted if the computation halts.

Two main types of results were obtained: computational completeness in the case when no bound was imposed on the number of spikes present in the system, and a characterization of semilinear sets of numbers in the case when a bound was imposed (hence for finite SN P systems).

Another attractive possibility is to consider the spike trains themselves as the result of a computation, and then we obtain a (binary) language generating device. We can also consider input neurons and then an SN P system can work as a transducer. Such possibilities were investigated in [26]. Languages – even on arbitrary alphabets – can be obtained also in other ways: following the path of a designated spike across neurons, or using extended rules. Specifically, with a step when the system sends out \( i \) spikes, we associate a symbol \( b_i \), and thus we get a language over an alphabet with as many symbols as the number of spikes simultaneously produced. This case was investigated in [9].

The proofs of all computational completeness results known up to now in this area are based on simulating register machines. Starting the proofs from small universal register machines, as those in [20], one can find small universal SN P systems. This idea was explored in [22] – the results are recalled in Theorem 5.4.

In the initial definition of SN P systems several ingredients are used (delay, forgetting rules), some of them of a general form (general synapse graph, general regular expressions). As shown in [15], rather restrictive normal forms can be found, in the sense that some ingredients can be removed or simplified without losing the computational completeness. For instance, the forgetting rules or the delay can be removed, both the indegree and the outdegree of the synapse graph can be bounded by 2, while the regular expressions from firing rules can be of very restricted forms.

There were investigated several other types of SN P systems: with several output neurons ([16], [17]), with a non-synchronous use of rules ([4]), with an exhaustive use of rules (whenever enabled, a rule is used as much as possible for the number of spikes present in the neuron, [19]), with packages of spikes sent along specified synapse links ([1]), etc. We refer the reader to the bibliography of this note, with many papers being available at [27].

### 3 A Formal Definition

We introduce the SN P systems in a general form, namely, in the extended (i.e., with the rules able to produce more than one spike) computing (i.e., able to take an input and provide an output) version.
A computing extended spiking neural P system, of degree \( m \geq 1 \), is a construct of the form

\[ \Pi = (O, \sigma_1, \ldots, \sigma_m, \text{syn}, \text{in}, \text{out}), \]

where:

1. \( O = \{a\} \) is the singleton alphabet (\( a \) is called spike);
2. \( \sigma_1, \ldots, \sigma_m \) are neurons, of the form \( \sigma_i = (n_i, R_i) \), \( 1 \leq i \leq m \), where:
   a) \( n_i \geq 0 \) is the initial number of spikes contained in \( \sigma_i \);
   b) \( R_i \) is a finite set of rules of the following two forms:
      1. \( E/\alpha^c \rightarrow a^p; d \), where \( E \) is a regular expression over \( a \) and \( c \geq p \geq 1, d \geq 0 \);
      2. \( a^s \rightarrow \lambda \), for \( s \geq 1 \), with the restriction that for each rule \( E/\alpha^c \rightarrow a^p; d \) of type (1) from \( R_i \), we have \( a^s \not\in \text{L}(E) \);
3. \( \text{syn} \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \) with \( i \neq j \) for all \( (i, j) \in \text{syn}, 1 \leq i, j \leq m \) (synapses between neurons);
4. \( \text{in}, \text{out} \in \{1, 2, \ldots, m\} \) indicate the input and the output neurons, respectively.

The rules of type (1) are firing (we also say spiking) rules, those of type (2) are called forgetting rules. An SN P system whose firing rules have \( p = 1 \) (they produce only one spike) is said to be of the standard type (non-extended).

The firing rules are applied as follows. If the neuron \( \sigma_i \) contains \( k \) spikes, and \( a^s \in \text{L}(E), k \geq c \), then the rule \( E/\alpha^c \rightarrow a^p; d \) \( \in \text{R}_i \) can be applied. This means consuming (removing) \( c \) spikes (thus only \( k - c \) spikes remain in \( \sigma_i \); this corresponds to the right derivative operation \( \text{L}(E)/\alpha^c \)), the neuron is fired, and it produces \( p \) spikes after \( d \) time units (a global clock is assumed, marking the time for the whole system, hence the functioning of the system is synchronized). If \( d = 0 \), then the spikes are emitted immediately, if \( d = 1 \), then the spikes are emitted in the next step, etc. If the rule is used in step \( t \) and \( d \geq 1 \), then in steps \( t, t + 1, t + 2, \ldots, t + d - 1 \) the neuron is closed (this corresponds to the refractory period from neurobiology), so that it cannot receive new spikes (if a neuron has a synapse to a closed neuron and tries to send a spike along it, then that particular spike is lost). In the step \( t + d \), the neuron spikes and becomes again open, so that it can receive spikes (which can be used starting with the step \( t + d + 1 \), when the neuron can again apply rules). Once emitted from neuron \( \sigma_i \), the spikes reach immediately all neurons \( \sigma_j \) such that \( (i, j) \in \text{syn} \) and which are open, that is, the \( p \) spikes are replicated and each target neuron receives \( p \) spikes; spikes sent to a closed neurons are “lost”.

The forgetting rules are applied as follows: if the neuron \( \sigma_i \) contains exactly \( s \) spikes, then the rule \( a^s \rightarrow \lambda \) from \( \text{R}_i \) can be used, meaning that all \( s \) spikes are removed from \( \sigma_i \).
If a rule $E/a^c \rightarrow a^p; d$ of type (1) has $E = a^c$, then we write it in the simplified form $a^c \rightarrow a^p; d$.

In each time unit, if a neuron $\sigma_i$ can use one of its rules, then a rule from $R_i$ must be used. Since two firing rules, $E_1/a^c_1 \rightarrow a^p_1; d_1$ and $E_2/a^c_2 \rightarrow a^p_2; d_2$, can have $L(E_1) \cap L(E_2) \neq \emptyset$, it is possible that two or more rules can be applied in a neuron, and in that case, only one of them is chosen non-deterministically. Note however that, by definition, if a firing rule is applicable, then no forgetting rule is applicable, and vice versa.

Thus, the rules are used in the sequential manner in each neuron, at most one in each step, but neurons function in parallel with each other. It is important to notice that the applicability of a rule is established based on the total number of spikes contained in the neuron.

The initial system configuration is described by the numbers $n_1, n_2, \ldots, n_m$, of spikes present in each neuron, with all neurons being open. During the computation, a configuration is described by both the number of spikes present in each neuron and by the state of the neuron, more precisely, by the number of steps to count down until it becomes open (this number is zero if the neuron is already open). Thus, $\langle r_1/t_1, \ldots, r_m/t_m \rangle$ is the configuration where neuron $\sigma_i$ contains $r_i \geq 0$ spikes and it will be open after $t_i \geq 0$ steps, $i = 1, 2, \ldots, m$; with this notation, the initial configuration is $C_0 = \langle n_1/0, \ldots, n_m/0 \rangle$.

A computation in a system as above starts in the initial configuration. In order to compute a function $f : \mathbb{N}^k \rightarrow \mathbb{N}$, we introduce $k$ natural numbers $n_1, \ldots, n_k$ in the system by "reading" from the environment a binary sequence $z = 10^{n_1-1}10^{n_2-1}1 \ldots 10^{n_k-1}1$. This means that the input neuron of $\Pi$ receives a spike in each step corresponding to a digit 1 from the string $z$ and no spike otherwise. Note that we input exactly $k + 1$ spikes, i.e., after the last spike we assume that no further spike is coming to the input neuron. The result of the computation is also encoded in the distance between two spikes: we impose the restriction that the system outputs exactly two spikes and halts (sometimes after the second spike), hence it produces a train spike of the form $0^b_110^{r-1}1b_2^c$, for some $b_1, b_2 \geq 0$ and with $r = f(n_1, \ldots, n_k)$ (the system outputs no spike a non-specified number of steps from the beginning of the computation until the first spike).

The previous definition covers many types of systems/behaviors. If the neuron $\sigma_{in}$ is not specified, then we have a generative system: we start from the initial configuration and we collect all results of computations, which can be the distance between the first two spikes (as in [18]), the distance between all consecutive spikes, between alternate spikes, etc. (as in [25]), or it can be spike train itself, either taking only finite computations, hence generating finite strings (as in [5], [9], etc.), or also non-halting computations (as in [26]). Similarly, we can ignore the output neuron and use an SN P system in the accepting mode: a number introduced in the system as the distance between two spikes entering the input
neuron is accepted if and only if the computation halts. In the same way we can accept input binary strings or strings over arbitrary alphabets. In the second case, a symbol $b_i$ is taken from the environment by introducing $i$ spikes in the input neuron.

4 Two Examples

Not all types of SN P systems will be discussed below, and only two of them are illustrated in this section.

The first example, given in Figure 2, is actually of a more general interest, as it is a part of a larger SN P system which simulates a register machine. The figure presents the module which simulates a SUB instruction; moreover, it does it without using forgetting rules (the construction is part of the proof that forgetting rules can be avoided – see [15]).

![Diagram](image-url)

**Figure 2:** Module SUB (simulating $l_i : (\text{SUB}(r), l_j, l_k)$)

The idea of simulating a register machine $M = (n, H, l_0, l_h, R)$ ($n$ registers, set of labels, initial label, halt label, set of instructions) by an SN P system $\Pi$ is to associate a neuron $\sigma_r$ with each register $r$ and a neuron $\sigma_l$ with each label $l$ from $H$ (there also are other neurons – see the figure), and to represent the fact that register
This page contains the number $k$ by having $2^k$ spikes in neuron $\sigma_r$. Initially, all neurons are empty, except neuron $\sigma_{l_0}$, which contains one spike. During the computation, the simulation of an instruction $l_i : \langle \text{OPP}(r), l_j, l_k \rangle$ starts by introducing one spike in the corresponding neuron $\sigma_{l_i}$, and this triggers the module associated with this instruction.

For instance, in the case of a subtraction instruction $l_i : \langle \text{SUB}(r), l_j, l_k \rangle$, the module is initiated when a spike enters neuron $\sigma_{l_i}$. This spike causes neuron $\sigma_{l_i}$ to immediately send a spike to neurons $\sigma_{l_i1}, \sigma_{l_i2}$, and $\sigma_r$. If register $r$ is not empty, then the rule $a(aaa)/a^3 \rightarrow a;0$ will be applied and the spike emitted will cause neurons $\sigma_{l_i1}, \sigma_{l_i2}$, and finally neuron $\sigma_{l_i}$ to spike. (In this process, neuron $\sigma_{l_i2}$ has two spikes added during one step and it cannot spike.) If register $r$ is empty, hence neuron $\sigma_r$ contains only the spike received from $\sigma_{l_i}$, then the rule $a \rightarrow a;1$ is applied and the subsequent spikes will cause neurons $\sigma_{l_i1}, \sigma_{l_i2}$, and finally neuron $\sigma_{l_i}$ to spike. (In this process, neuron $\sigma_{l_i2}$ has two spikes added during one step and does not spike.) After the computation of the entire module is complete, each neuron is left with either zero spikes or an even number of spikes, allowing the module to be run again in a correct way.

The second example deals with an SN P system used as a transducer, and it illustrates the following result from [26]: Any function $f : \{0, 1\}^k \rightarrow \{0, 1\}$ can be computed by an SN P system with $k$ input neurons (also using further $2^k + 4$ neurons, one being the output one).

The idea of the proof of this result is suggested in Figure 3, where a system is presented which computes the function $f : \{0, 1\}^3 \rightarrow \{0, 1\}$ defined by

$$f(b_1, b_2, b_3) = 1 \text{ iff } b_1 + b_2 + b_3 \neq 2.$$  

The three input neurons, $\sigma_{in1}, \sigma_{in2}, \sigma_{in3}$, are continuously fed with bits $b_1, b_2, b_3$, and the output neuron will provide the value of $f(b_1, b_2, b_3)$ with a 3-steps delay.

## 5 Some Results

Several parameters describe the complexity of an SN P system: number of neurons, of rules, of spikes consumed or forgotten by a rule, etc. Here we consider only some of them and we denote by $N_{SNP_{m}(\text{rules}, \text{cons}, \text{forg})}$ the family of all sets $N_2(\Pi)$ computed as specified in Section 3 by SN P systems with at most $m \geq 1$ neurons, using at most $k \geq 1$ rules in each neuron, with all spiking rules $E/a^t \rightarrow a; t$ having $r \leq p$, and all forgetting rules $a^t \rightarrow \lambda$ having $s \leq q$. When any of the parameters $m, k, p, q$ is not bounded, it is replaced with *. When we work only with SN P systems whose neurons contain at most $s$ spikes at any step of a computation (finite systems), then we add the parameter $\text{bound}_s$ after $\text{forg}_q$. 

---
Figure 3: Computing a Boolean function of three variables

(Corresponding families are defined for other definitions of the result of a computation, as well as for the accepting case, but the results are quite similar, hence we do not give details here.)

By $\mathcal{NFIN}, \mathcal{NREG}, \mathcal{NRE}$ we denote the families of finite, semilinear, and Turing computable sets of (positive) natural numbers (number 0 is ignored); they correspond to the length sets of finite, regular, and recursively enumerable languages, whose families are denoted by $\mathcal{FIN}, \mathcal{REG}, \mathcal{RE}$. We also invoke below the family of recursive languages, $\mathcal{REC}$ (those languages with a decidable membership).

The following results were proved in [18] and extended in [25] to other ways of defining the result of a computation.

**Theorem 5.1.** (i) $\mathcal{NFIN} = \mathcal{N}_2\mathcal{SNP}_1(\text{rule}_e, \text{cons}_1, \text{forg}_0) = \mathcal{N}_2\mathcal{SNP}_2(\text{rule}_e, \text{cons}_e, \text{forg}_e)$.

(ii) $\mathcal{N}_2\mathcal{SNP}_k(\text{rule}_e, \text{cons}_p, \text{forg}_q) = \mathcal{NRE}$ for all $k \geq 2$, $p \geq 3$, $q \geq 3$.

(iii) $\mathcal{NSLIN} = \mathcal{N}_2\mathcal{SNP}_k(\text{rule}_e, \text{cons}_p, \text{forg}_q, \text{bound}_s)$, for all $k \geq 3$, $q \geq 3$, $p \geq 3$, and $s \geq 3$.

Point (ii) was proved in [18] also for the accepting case, and then the sys-
The Bulletin of the EATCS

tems used can be required to be deterministic (at most one rule can be applied in each neuron in each step of the computation). In turn, universality results were proved in [19] and [4] also for the exhaustive and for the non-synchronized modes of using the rules, respectively, but only for extended rules. The universality of standard systems remains open for these cases.

Let us now mention some results about languages generated by SN P systems, starting with the restricted case of binary strings, [5]. We denote by \( L(\Pi) \) the set of strings over the alphabet \( B = \{0, 1\} \) describing the spike trains associated with halting computations in \( \Pi \); then, we denote by \( LSNP_m(\text{rule}_i, \text{cons}_p, \text{forg}_q) \) the family of languages \( L(\Pi) \), generated by SN P systems \( \Pi \) with the complexity bounded by the parameters \( m, k, p, q \) as specified above. When using only systems with at most \( s \) spikes in their neurons (finite), we write \( LSNP_m(\text{rule}_i, \text{cons}_p, \text{forg}_q, \text{bound}_i) \) for the corresponding family. As usual, a parameter \( m, k, p, q, s \) is replaced with * if it is not bounded.

**Theorem 5.2.** (i) There are finite languages (for instance, \( \{0^j, 1^j\} \), for any \( k \geq 1 \), \( j \geq 0 \)) which cannot be generated by any SN P system, but for any \( L \in \text{FIN} \), \( L \subseteq B^* \), we have \( L[1] \in LSNP_1(\text{rule}_i, \text{cons}_p, \text{forg}_q, \text{bound}_i) \), and if \( L = \{x_1, x_2, \ldots, x_n\} \), then we also have \( \{0^{n+1}x_i \mid 1 \leq i \leq n\} \in LSNP_1(\text{rule}_i, \text{cons}_p, \text{forg}_q, \text{bound}_i) \).

(ii) The family of languages generated by finite SN P systems is strictly included in the family of regular languages over the binary alphabet, but for any regular language \( L \subseteq V^* \) there is a finite SN P system \( \Pi \) and a morphism \( h : V^* \rightarrow B^* \) such that \( L = h^{-1}(L(\Pi)) \).

(iii) \( LSNP_m(\text{rule}_i, \text{cons}_p, \text{forg}_q) \subseteq \text{REC} \), but for every alphabet \( V = \{a_1, a_2, \ldots, a_k\} \) there are two symbols \( b, c \) not in \( V \), a morphism \( h_1 : (V \cup \{b, c\})^* \rightarrow B^* \), and a projection \( h_2 : (V \cup \{b, c\})^* \rightarrow V^* \) such that for each language \( L \subseteq V^* \), \( L \in \text{RE} \), there is an SN P system \( \Pi \) such that \( L = h_2(h_1^{-1}(L(\Pi))) \).

These results show that the language generating power of SN P systems is rather eccentric; on the one hand, finite languages (like \( \{0^j, 1^j\} \)) cannot be generated, on the other hand, we can represent any RE language as the direct morphic image of an inverse morphic image of a language generated in this way. This eccentricity is due mainly to the restricted way of generating strings, with one symbol added in each computation step. This restriction does not appear in the case of extended spiking rules. In this case, a language can be generated by associating the symbol \( b_i \) with a step when the output neuron sends out \( i \) spikes, with an important decision to take in the case \( i = 0 \): we can either consider \( b_0 \) as a separate symbol, or we can assume that emitting 0 spikes means inserting \( \lambda \) in the generated string. Thus, we both obtain strings over arbitrary alphabets, not only over the binary one, and, in the case where we ignore the steps when no spike is emitted, a considerable freedom is obtained in the way the computation proceeds.
This latter variant (with \( \lambda \) associated with steps when no spike exits the system) is considered below.

We denote by \( LSN^e P_m(\text{rule}_k, \text{cons}_p, \text{prod}_q) \) the family of languages \( L(\Pi) \), generated by SN P systems \( \Pi \) using extended rules, with the parameters \( m,k,p,q \) as above.

The next counterparts of the results from Theorem 5.2 were proved in [9].

**Theorem 5.3.**

(i) \( FIN = LSN^e P_1(\text{rule}_*, \text{cons}_*, \text{prod}_*) \) and this result is sharp, as \( LSN^e P_2(\text{rule}_2, \text{cons}_2, \text{prod}_2) \) contains infinite languages.

(ii) \( LSN^e P_3(\text{rule}_*, \text{cons}_*, \text{prod}_*) \subseteq REG \subset LSN^e P_4(\text{rule}_*, \text{cons}_*, \text{prod}_*) \); the second inclusion is proper, because \( LSN^e P_3(\text{rule}_3, \text{cons}_3, \text{prod}_3) = REG \neq \emptyset \); actually, \( LSN^e P_3(\text{rule}_*, \text{cons}_*, \text{prod}_*) \) contains non-semilinear languages.

(iii) \( RE = LSN^e P_4(\text{rule}_*, \text{cons}_*, \text{prod}_*) \).

It is an open problem to find characterizations or representations in this setup for families of languages in the Chomsky hierarchy different from \( FIN, REG, RE \). We close this section by mentioning the results from [22]:

**Theorem 5.4.** There is a universal computing SN P system with (i) standard rules and 84 neurons and with (ii) extended rules and 49 neurons, and there is a universal SN P system used as a generator of sets of numbers with (iii) standard rules and 76 neurons and with (iv) extended rules and 50 neurons.

These values can probably be improved (but the feeling is that this improvement cannot be too large).

Tool-kits for handling strings or infinite sequences, on the binary or on the arbitrary alphabet, are provided in [26] and [10]. For instance, in this latter paper one gives constructions of SN P systems for computing the union and concatenation of two languages generated by SN P systems, the intersection with a regular language, while the former paper shows how length preserving morphisms (codings) can be computed; the problem remains open for arbitrary morphisms, Kleene *, inverse morphisms.

An interesting result is reported in [6]: SAT can be decided in constant time by using an arbitrarily large pre-computed SN P system, of a very regular shape (as synapse graph) and with empty neurons, after plugging the instance of size \((n,m)\) \((n\) variables and \(m\) clauses) of the problem into the system, by introducing a polynomial number of spikes in \((\text{polynomially many})\) specified neurons. This way of solving a problem, by making use of a pre-computed resource given for free, on the one hand, resembles the supposed fact that only part of the brain neurons are active (involved in “computations”) at each time, on the other hand, is not very common in computability and request further research efforts (what kind of pre-computed resource is allowed, so that no “cheating” is possible? how the
given resources should be activated? define and study complexity classes for this framework).

6 Plenty of Research Topics

Many problems were already mentioned above, many others can be found in the papers listed below, and further problems are given in [24]. We recall only some general ideas: bring more ingredients from neural computing, especially related to learning/training/efficiency; incorporate other facts from neurobiology, such as the role played by astrocytes, the way the axon not only transmit impulses, but also amplifies them; consider not only “positive” spikes, but also inhibitory impulses; define a notion of memory in this framework, which can be read without being destroyed; provide ways for generating an exponential working space (by splitting neurons? by enlarging the number of synapses?), in such a way to trade space for time and provide polynomial solutions to computationally hard problems; define systems with a dynamical synaptic structure; compare the SN P systems as generator/acceptor/transducers of infinite sequences with other devices handling such sequences; investigate further systems with exhaustive and other parallel ways of using the rules, as well as systems working in a non-synchronized way; find classes of (accepting) SN P systems for which there is a difference between deterministic and non-deterministic systems; find classes which characterize levels of computability different from those corresponding to finite automata (semilinear sets of numbers or regular languages) or to Turing machines (recursively enumerable sets of numbers or languages).

We close with a more technical idea: use more general types of rules, for instance, of the form $E[a^n \rightarrow a^{f(n)}]; d,$ where $f$ is a partial function from natural numbers to natural numbers (maybe with the property $f(n) \leq n$ for all $n$ for which $f$ is defined), and used as follows: if the neuron contains $k$ spikes such that $a^k \in L(E)$, then $c$ of them are consumed and $f(c)$ are created, for $c = \max\{n \in \mathbb{N} \mid n \leq k, \text{ and } f(n) \text{ is defined}\}$; if $f$ is defined for no $n$ smaller than or equal to $k$, then the rule cannot be applied. This kind of rules looks both adequate from a neurobiological point of view (the sigmoid excitation function can be captured) and mathematically powerful.

References


The Programming Languages Column

By

Ian Mackie

Department of Computer Science
King’s College London, Strand, London, WC2R 2LS, UK
Ian.Mackie@kcl.ac.uk, http://www.dcs.kcl.ac.uk/staff/ian

Web Services, Mobile Processes and Types

Kohei Honda
Department of Computer Science
Queen Mary, University of London
United Kingdom
kohei@dcs.qmul.ac.uk

Nobuko Yoshida
Department of Computing
Imperial College London
United Kingdom
yoshida@doc.ic.ac.uk

Marco Carbone
Department of Computing
Imperial College London
United Kingdom
carbonem@doc.ic.ac.uk

Abstract

This tutorial outlines the use of mobile processes and their type theory for developing basic programming constructs for Web Services. It draws on our experience in participating, as invited experts, in a W3C standardisation of a web service description language, Web Services Choreography Description Language (WS-CDL). We discuss the significance of using global message flows for describing Web Service interactions and of modelling this idea as a
Introduction: Challenges of Web Services

Recent years have seen the emergence of a new style of distributed software system designed to support interoperable machine-to-machine interaction over a network, using the infrastructure of the world-wide web. This system is called Web Services. Web services start from a machine-readable web site providing information to a software application in the same way that a conventional web site offers information to a human user. The term “Web Services” sometimes just means a service thus provided, but in this tutorial, we use the term for indicating a broader class of applications which arise through mutual interactions among these web sites using universally shared standards such as HTTP, TCP/IP and XML. These interactions can make up a sophisticated application whose major mode of computation is communication among distributed computing entities. At the heart of the development of these applications is the design and implementation of application-level protocols among distributed sites.

The advent of Web Services, together with other trends such as the emergence of multicore processors and ubiquitous computing, is contributing to a shift in the software development paradigm, where communication and concurrency are a norm rather than exceptions. This new paradigm however still lacks a mature programming methodology. Even sequential programming is hard, especially if we wish to guarantee safety and security. Programming communication and concurrency is harder, exposing programmers and designers to the new level of complexity including composition of communication behaviours, deadlock, livelock, and diverse forms of partial failure. But the development of major programming languages in use today has focussed on the description of sequential behaviour mixed with moderate shared-variable concurrency. And major application areas for Web Services such as business protocols demand stringent safety and security.

Thus Web Services pose major technical challenges in programming methodologies. At a most basic level, these challenges may be summarised as follows.

1. We should be able to describe communication-centred behaviour clearly, accurately and modularly.

2. We should be able to validate and detect critical properties of programs with respect to their communication behaviour, such as deadlock freedom, during its development.
3. We should be able to control run-time behaviour of programs including their composition.

These challenges are related to each other: for example, without good programming abstraction from (1), modular validation becomes impossible, making (2) intractably harder; and the control of behaviour in (3) is assisted by (2), often crucially, just as type-safety of bytecode guarantees the lack of basic security holes in Java Virtual Machine.

How can we tackle these challenges? Recalling that a long accumulation of research and practice has been necessary to reach the now mature traditional high-level programming methodology, we may as well expect that this may not be an easy task. We need to find a comprehensive methodology for programming communication-centred software on the web based on the good understanding of this new paradigm, which can realise a rich collection of behaviours far surpassing sequential ones.

In this tutorial, we discuss one possible approach to these challenges, based on the $\pi$-calculus, a theory of communicating mobile systems introduced by Milner, Parrow and Walker [42]. The $\pi$-calculus is built on a small set of ideas: processes, their parallel composition, localisation of names, and communication of names. A surprising discovery is that the facility for communication of names leads to a universal theory of communicating processes: the calculus can represent, with preservation of structure and dynamics, the core notions of sequential and concurrent computing [41, 53]. The approach is based on the long-running study of theories of mobile processes and their type theory. It is also based on an extensive dialogue with web service engineers through our participation in the W3C working group for web service language standardisation in the team of $\pi$-calculus experts (led by Robin Milner, including the present authors), on which we shall discuss in Section 2.

In the rest of the tutorial, the historical and conceptual background of the presented technical ideas is illustrated in Section 2, followed by an outline of the formalism in Section 3 and its relationship with the $\pi$-calculus in Section 4. Section 5 summarises closely related preceding works and discusses further applications. The tutorial does not assume preliminary knowledge of either the $\pi$-calculus or Web Services.

Part of Section 2.1 is written based on Steve Ross-Talbot’s summary of a history of WS-CDL [50]. The technical parts of the tutorial have been developed with Marco Carbone. A prior paper [15] illustrates the use of the global calculus for describing communication behaviour through examples; whereas [16] presents an associated theory. More examples, further discussions and full technical developments including detailed proofs are in [17], a contribution by the whole of the $\pi$-calculus experts team to the aforementioned W3C standard.
2 Global Description of Interactions

2.1 Describing Business Protocols: a historical background

One of the main application areas of Web Services is business protocols. A business protocol is a series of structured and automated interactions among two or more business entities. Describing and controlling complex business protocols is not an easy task: the engineers want to automate their specified protocols in a scalable way; to ensure that the business protocols that they enact reflect what they want to happen; and to manage risk in the event of failure during the execution.

Since the early 1990’s, among industries, there have been many attempts at describing and modelling business processes and protocols to achieve these goals. The first attempt back in the early 1990’s (such as Fidelio project by IBM which later became IBM WebSphere MQ-Workflow [32]) was based on individual, proprietary description languages, which was not desirable from the viewpoint of interoperability, leading to such issues as vendor lock-in\(^1\) (at the time, no de facto/de jure standards existed that could integrate existing systems in a unified manner). These proprietary technologies also suffered from high integration costs.

In the latter part of the 1990’s, a standardisation group, the Work Flow Management Coalition (WFMC) [3], was formed. Its task was to standardise the description of workflows to enable its interoperability. This had solved part of the issues, i.e. the lack of standards in description languages, but could not remove the problems of integration because the underlying workflow technologies were still bound to the proprietary solutions.

In 2002, IBM proposed the Web Service Flow Language (WSFL) [33] and Microsoft released Xlang [39] for Biztalk. These two originally proprietary offerings were merged in the development of BPEL (later WS-BPEL, Web Services Business Process Execution Language) in 2003. BPEL solved the problem of integration by utilising Web Service standards, WSDL (Web Services Description Language) and SOAP to make legacy integration part of the standards solution.

In BPEL, one can describe a workflow internal to a company, providing a server-centric orchestration mechanism. In orchestration, one master component, “conductor”, directly controls activity of one or more slave components. This mechanism is useful when one wishes to describe a workflow among components which sit within a common administrative domain. It is, however, not intended for peer-to-peer distributed communications, and assumes a single point of control (hence of failure). In WS-BPEL, we can combine orchestration with communication actions which a business will be engaged in with other business entities.

\(^1\)This term, also known as proprietary lock-in or lock-in, is a situation in which a customer is dependent on a vendor for products and services so that he or she cannot move to another vendor without substantial switching costs [2].
During the latter part of the 1990's, most large institutions had started investing heavily in distributed computing technologies for their business processes, for the purpose of reducing the risk of centralised controls and replacing them with distributed controls.

2.2 WS-CDL, a Language for Choreography

Against these backgrounds, the Web Services Choreography Description Language Working Group (WS-CDL WG) [59] was formed by W3C in 2003 for standardising a language for specifying web service business protocols. The language aims to be used for describing distributed interactions among peers (business entities), a departure from the orchestration view. Recognising the need for a foundational theory on which the design and infrastructure of the language is to be built on, the working group has taken a strong interest in the $\pi$-calculus, leading to the involvement of Milner, Honda and Yoshida as invited scientists in the standardisation. While WS-CDL's design is informed by the $\pi$-calculus in both communication primitives and structuring constructs, WS-CDL differs from the $\pi$-calculus in that it describes message flows among multiple participants globally, seen from a vantage viewpoint.

Historically, engineers have found it essential to use various notations for describing interactions globally, such as the notations for cryptographic protocols, UML sequence diagrams and message sequence charts. This is because a global description presents the information on behaviour of systems not immediately available from the corresponding endpoint-based descriptions: how conversations among multiple participants evolve and interleave, what are synchronisation/communication points among participants, and how they together induce a desired global scenario.

WS-CDL follows these preceding global notations, and in addition offers a fully expressive language for channel based communication, equipped with general control constructs (for example, sequencing, conditionals and recursion). This aspect of WS-CDL is clearly embodied in the term choreography, whose underlying intuition may be summarised thus:

"Dancers dance following a global scenario (choreography) without a single point of control."

WS-CDL is conceived as a language for describing such a "global scenario". Once specified, this scenario is to be executed by individual distributed processes without orchestrating nodes. For a further introduction to WS-CDL, see a recent volume of ACM Queue [4], where Steve Ross-Talbot, the W3C chair of Web Services and the WS-CDL co-chair, is interviewed by the CIO of Morgan Stanley. In the next section, we present a formalism which distills the core ideas of WS-CDL.
As we have just observed, a global description is meant to be executed by distributed interactions among end-point processes. Thus each global description should be projected onto processes at each end-point, whose mutual communications precisely realise the original global scenario. This translation from a global description to end-point processes is called end-point projection (EPP), a terminology from the WS-CDL WG. At a theoretical level, the EPP is formalised as an encoding of the global calculus onto the $\pi$-calculus which preserves (among others) the original dynamics. In turn, having a theoretically well-founded EPP turns out to be fundamental for the practice of business protocols, for addressing the concerns of interoperability, implementability, run-time monitoring, and other concerns, as we shall illustrate in Section 4.

2.3 Types for Mobile Processes and Session Types

We now turn to types. One of the purposes of WS-CDL is to make the best of accumulated theories of types for the $\pi$-calculus. Why types? Generally speaking, types for programming languages abstract and classify behaviours compositionally. Notable points are:

1. Types offer clean and precise abstractions of behaviour, aiding modularisation and structuring.

2. Types offer a foundation for more refined verification methods.

3. Types can often be inferred statically by efficient algorithms, exploiting their compositional nature.

Regarding the first point, a basic observation is that types are almost always associated with the central abstraction mechanisms a language provides for programming. For example, the following type annotation for a method/procedure as found in Java:

```java
bool foobar(int i) {...}
```

says that this method/procedure inputs an integer and returns a boolean value. The type being used here, which is formally the standard arrow type, is tightly coupled with the associated fundamental abstraction mechanism, the method or procedure call.

Theories of types for the $\pi$-calculus encompass a wide variety of behavioural properties (in fact, it even embeds many of the notions of types for the $\lambda$-calculus [10, 61], another calculus with a rich theory of types). It starts from a basic notion of types which only ensures the standard type correctness to more advanced behavioural properties such as deadlock freedom. In the formalism we shall discuss
in the present tutorial, we consider a relatively simple notion of types for mobile processes, so-called session types [54, 31]. Session types offer a useful abstraction to communication-centred programming and can serve as a building block of other type theories for processes. Its choice in the present context is partly based on the observations which the authors could obtain through the participation of the design process of WS-CDL (which is informally based on a variant of sessions).

Informally, a session is a series of communications between two parties which form a meaningful logical unit (just like a web session between a browser and a web server when a human user interacts with an E-commerce site). Session types abstract a session in this sense as an abstract structure. The session types have been studied for the π-calculus [12, 25, 24, 31, 54, 57, 23], Ambients [19], CORBA interfaces [55], Concurrent Haskell [44], operating systems [22], multi-threaded functional languages [58, 57] and distributed [21] and multi-threaded Java [20]. In the context of web services, the structures of interactions of a service may not be restricted to RPC-like request-replies, which is why thinking in terms of sessions rather than the request-reply is significant.

Session can be implemented efficiently in both distributed and shared memory environments [22] and allow efficient type checking through simple algorithms [58]. Further, the session types and their extensions facilitate static and compositional verifications by providing a structured view of communication behaviour, ranging from freedom from communication errors to deadlocks, livelocks and race conditions – all of which are common in distributed systems.

The global calculus presented in the next section organises its communication primitives based on sessions and session types. As we shall see in Section 4, session types in the global calculus also play a central role when we relate the global calculus to the π-calculus.

3 A Global Calculus

3.1 Buyer-Seller Protocol

We outline key technical ideas of the global calculus using a running example. The example is a simple business protocol from [51], the “Buyer-Seller Protocol”. In the global calculus, we describe interactions as those among participants, i.e. the subjects (modelling business entities) who send and receive messages as well as doing internal computation. The participants involved in the present scenario are a Buyer, a Seller and a Shipper. We first describe the protocol informally, using a notation close to the UML sequence diagram.
The Bulletin of the EATCS

The diagram is ambiguous at the branching (+) actions in (4) and (5): the purpose of such diagrams is to offer an informal overview: they naturally omit detailed control structures (choices, loops, etc.) and manipulation of values/states. The reason why such global descriptions are practised in engineering is because they enable a clear grasp of the whole interaction structure, carrying out which is often the central purpose of the application. Another technical reason is that, by describing the whole interaction structure, it gives how communications at each site should be interleaved, lessening synchronisation and other errors at the design stage.

The global calculus is intended to offer a distillation of these virtues as an expressive formal calculus, equipped with general control constructs and typed communication primitives. Our starting point is to understand, and make explicit, the key engineering principles which may govern the programming of Web Services. While diverse concerns exist for software development on the web, we have singled out the following two as most fundamental.

Service Channel Principle (SCP): Invocation channels (for example, a channel at which Buyer first communicates to Seller, or Seller to Shipper) can be shared and invoked repeatedly.

Session Principle (SP): A conversation — a meaningful chunk of a sequence of interactions — in a protocol should not be confused with other concurrent conversations in this or other protocols by the same participants: each such sequence should form a logical unit of a conversation, or a session.

(SCP) is a natural principle arising from the origin and infrastructural basis of Web Services, and is also related with the standard abstraction for sequential programs such as objects (which can be invoked many times albeit sequentially). Concretely, one may consider such channels as public URLs offering services.
Figure 1 Business Protocols in the Global Calculus

(a) Buyer-Seller Example

1. Buyer → Seller : quoteCh(s).
2. Seller → Buyer : s(quote, 300, x).
3. |{Buyer → Seller : s(accept).}
4. Seller → Shipper : delivCh(t).
5. Shipper → Seller : t(details, v, x).
6. Seller → Buyer : s(details, x, y).0
7. +
8. |{Buyer → Seller : s(reject).0}|

(b) Protocol with Recursion

1. Buyer → Seller : quoteCh(s).
2. rec x. { }
3. Seller → Buyer : s(quote, q, x).
4. if reasonable(x)@Buyer then
5. |{Buyer → Seller : s(accept).}
7. Shipper → Seller : t(details, v, x).
8. Seller → Buyer : s(details, x, y).0
9. else
10. |{Buyer → Seller : s(reject). q@Seller := q@Seller−1.X}|

Theoretically, it corresponds to the repeated availability of replicated input channels in the π-calculus (called uniformly receptive [52] and server channels in [10]). (SP) is a basic principle in many communication-centred software. A simplest incarnation of this idea is RPC and RMI, in which case a conversation consists of an invocation and a subsequent reverse communication. Note two such conversations are never confused, i.e. when a program calls the same remote object twice, two sessions are kept distinct by some correlation mechanism.

The global calculus is built from formalisation of these two principles, as well as basic combinators for composing descriptions. These constructs are closely related to those found in WS-CDL, but are made explicit and formalised in the distilled presentation as a formal calculus. Before introducing the syntax formally, we first outline its basic ideas using an example.

Figure 1 (a) gives a description of the Buyer-Seller Protocol in the global calculus. In (a), Line 1 describes Action (1) in the protocol. The quoteCh is a service channel, which may be considered as a public URL for a specific service. The invocation marks the start of a session between the buyer and the seller: the ν-bound s is a session channel, a fresh name to be used for later communication in this session. Unlike standard process calculi, the syntax no longer describes input and output actions separately: the information exchange between two parties is directly described as one interaction. This point is a key departure from the π-calculus.

Line 2 describes Action (2) in the protocol, Seller’s reply to Buyer. The session has already been started and now the two participants communicate using the session channel s. In addition, three factors involved: quote identifies the particular operation used in this communication (i.e. request for quote), 300 is the quote sent by Seller; x is a variable located at Buyer where the communicated value will be stored.
Lines 3/8 describe Action (3), where Buyer communicates its choice (accept or reject) to Seller through s. Two series of actions which follow these choices are combined by + in Line 7. If accept is chosen, Seller sends Shipper the Buyer’s details via the service channel \( \text{delivCh} \) of Shipper, creating a fresh session channel t (Line 4). Then in Line 5, Shipper sends back the shipping details through t. Finally in Line 6, Seller forwards the details to Buyer by sending the value stored in variable x: here the protocol terminates. In Line 8, Buyer communicates reject, in which case the protocol immediately terminates.

The code in Figure 1 (a) offers a precise global description of the previous informal scenario. The distinction between service channels and session channels implements (SCP) and (SP). Sessions offer logical grouping of threads of interactions, where each thread starts with a procedure-call-like service invocation at a service channel and carry out in-session communications at associated session channels. This point can be seen more clearly in Figure 1 (b), a refinement of (a). In (b), if Buyer chooses reject, the protocol recurs to Line 3, after decrementing the quote. In Line 4, a unary predicate \( \text{reasonable(x)} \) is evaluated at Seller’s site (“@” indicates a location, similarly in Line 10). The session notation makes it clear that all \( \text{quote} \)-messages from Seller to Buyer in the recursion are done within a single session. Section 4 shall show that such session information plays a crucial role for correct implementations of the global description.

### 3.2 Syntax and Dynamics

The syntax of the global calculus is given by the following BNF.

\[
I ::= A \rightarrow B : ch(\tilde{s}).I \quad \text{(init)} \quad | \quad (\nu s) I \quad \text{(new)} \\
| \quad A \rightarrow B : s(op, e, y).I \quad \text{(comm)} \quad | \quad X \quad \text{(recvar)} \\
| \quad x@A := e . I \quad \text{(assign)} \quad | \quad I_1 + I_2 \quad \text{(sum)} \\
| \quad I_1 | I_2 \quad \text{(par)} \quad | \quad \text{rec } X . I \quad \text{(rec)} \\
| \quad \text{if } e@A \text{ then } I_1 \text{ else } I_2 \quad \text{(cond)} \quad | \quad 0 \quad \text{(inaction)}
\]

I, I’, . . . denote terms of the calculus, also called interactions. ch, ch’ . . . range over service channels; s, t, . . . range over session channels; \( \tilde{s} \) indicates a vector of session channels; A, B, C, . . . range over participants; x, y, z, . . . over local variables in each participant; X, X’, . . . over term variables; and e, e’, . . . over arithmetic and other first-order expressions. The term (init) denotes a session initiation by A on B’s service channel ch with fresh session channels \( \tilde{s} \) and continuation I. The interaction (comm) is the in-session communication over a session channel s. op denotes the operator, which is used as a label for selecting communication (just like a label in records and tagged unions). Note that y is free and does not bind in I. “|” and “+” denote respectively parallel and choice. \((\nu s) I\) is the \(\pi\)-calculus-like
name restriction, binding $s$ in $I$. Since such a hiding can only get generated by session initiation, we stipulate that a hiding never occurs under a prefix or inside a sum/conditional. (cond) and (assign) are the standard conditional and assignment ($e@A$ indicates $e$ is located at $A$). $\text{rec } X, I$ is recursion, where the variable $X$ is bound in $I$. $\textbf{0}$ denotes termination. The free and bound session channels and term variables are defined in the usual way. We often omit $\textbf{0}$ and empty vectors.

The dynamics of the global calculus is given by a reduction relation close to that of imperative languages. A state $\sigma$ assigns a value to the variables located at each participant. We shall write $\sigma@A$ to denote the portion of $\sigma$ local to $A$, and $\sigma[y@A \mapsto v]$ to denote a new state which is identical with $\sigma$ except that $\sigma'[y@A(y)]$ is equal to $v$. The reduction is generated by the rules in Table 1. “$(\sigma, I) \rightarrow (\sigma', I')$” says that $I$ in the state $\sigma$ performs one-step computation and becomes $I'$ with the new state $\sigma'$.

(G-Intr) is for session initiation: after $A$ initiates a session with $B$ on service channel $ch$, $A$ and $B$ share $\bar{s}$ locally (indicated by $(v\bar{s})$ from the $\pi$-calculus [41]), and the next $I$ is unfolded. The initiation channel $ch$ will play an important role for typing and the end-point projection later. (G-Com) is a key rule: the expression $e$ is evaluated into $v$ in the $A$-portion of the state $\sigma$ and then assigned to the variable $x$ located at $B$ resulting in new the state $\sigma[x@B \mapsto v]$. Note that the same variable

<table>
<thead>
<tr>
<th>Table 1 Reduction Relation for the Global Calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(G\text{-Intr})$</td>
</tr>
<tr>
<td>$(\sigma, A \rightarrow B : ch(\bar{s}), I) \rightarrow (\sigma, (v\bar{s}) I)$</td>
</tr>
<tr>
<td>$(G\text{-Assign})$</td>
</tr>
<tr>
<td>$\sigma \vdash e@A \parallel_v \sigma' = \sigma[x@A \mapsto v]$</td>
</tr>
<tr>
<td>$(G\text{-If})$</td>
</tr>
<tr>
<td>$\sigma \vdash e@A \parallel \text{tt}$</td>
</tr>
<tr>
<td>$(G\text{-Par})$</td>
</tr>
<tr>
<td>$(\sigma, I_1) \rightarrow (\sigma', I'_{1})$</td>
</tr>
<tr>
<td>$(\sigma, I_1 \parallel I_2) \rightarrow (\sigma', I'<em>{1} \parallel I'</em>{2})$</td>
</tr>
<tr>
<td>$(\sigma, \text{rec } X, I) \rightarrow (\sigma', I')$</td>
</tr>
<tr>
<td>$(G\text{-Struct})$</td>
</tr>
<tr>
<td>$I \equiv I'' \quad (\sigma, I) \rightarrow (\sigma', I') \quad I' \equiv I''$</td>
</tr>
</tbody>
</table>
(say \(x\)) can be located at different participants, so that \(\sigma \sqsubseteq A(x)\) and \(\sigma \sqsubseteq B(x)\) are distinct. Similarly to the session initiation, the session channel \(s\) is attached. The rule (G-STRUCT) makes use of structural congruence. The structural congruence relation is standard \([41]\); it is the least congruence relation \(\equiv\) on \(I\) such that | and + are commutative monoids, satisfies alpha-conversion and scope-opening (\(\nu s\)) \(I_1 \vdash I_2 \equiv (\nu s) (I_1 \vdash I_2)\) for \(s\) not in \(I_2\).

Consider, for instance, the interaction

\[
\text{Buyer} \rightarrow \text{Seller} : \text{quoteCh}(s), \text{Seller} \rightarrow \text{Buyer} : s\langle \text{quote}, 300, x \rangle.
\]

and let us evaluate it in the state \(\sigma\). By applying rule (Intr), we get

\[
(\sigma, (\nu s) \text{Seller} \rightarrow \text{Buyer} : s\langle \text{quote}, 300, x \rangle, I').
\]

Now, by applying rules (Rtas) and (Com) together in the state \(\sigma\) we get the pair \((\sigma[x@\text{Buyer} \mapsto 300], (\nu s) I')\).

### 3.3 Session Types for Global Descriptions

In this subsection, we summarise the types for the global calculus with examples. The grammar of types follow.

\[
\alpha ::= s \triangleright \Sigma \text{op}_i(\theta_i), \alpha_i \mid s \triangleleft \Sigma \text{op}_i(\theta_i), \alpha_i \mid \alpha_1 | \alpha_2 \mid \text{rec } t.\alpha \mid t \mid \text{end}
\]

\(\alpha, \alpha', \ldots\) are session types. \(s \triangleright \Sigma \text{op}_i(\theta_i), \alpha_i\) is a branching input type at session channel \(s\), indicating possibilities for receiving any of the operators from \(\text{op}_i\) (which are pairwise distinct) with a value of type \(\theta_i\); \(s \triangleleft \Sigma \text{op}_i(\theta_i), \alpha_i\), a branching output type at \(s\), is the exact dual of the above. \(\text{op}_i\) is used as a label in these types. The type \(\alpha_1 | \alpha_2\) is a parallel composition of \(\alpha_1\) and \(\alpha_2\), abstracting parallel composition of two sessions. We take | to be commutative and associative, with \(\text{end}\), the inaction type indicating session termination, being the identity. We demand session channels in \(\alpha_1\) and those in \(\alpha_2\) to be disjoint: this guarantees a linear use of session channels. \(t\) is a type variable, while \(\text{rec } t.\alpha\) is a recursive type, where \(\text{rec } t\) binds free occurrences of \(t\) in \(\alpha\). In recursive types, we assume each recursion is guarded, i.e., in \(\text{rec } t.\alpha\), \(\alpha\) is an \(n\)-ary parallel composition of input/output types. Recursive types are regarded as regular trees in the standard way \([24]\).

The essential point of session types is a clean notion of duality, which represents how the partner process interacts with it. Duality is convenient to modularly check a typability of parallel composed processes. The co-type, or dual, of \(\alpha\), written \(\text{co-}\alpha\), is given as follows.

\[
s \triangleright \Sigma \text{op}_i(\theta_i), \alpha_i = s \triangleleft \Sigma \text{op}_i(\theta_i), \text{co-}\alpha_i\quad \text{rec } t.\alpha = \text{rec } t.\text{co-}\alpha\quad \text{if } t = \text{co}\text{end} = \text{end}
\]
In this tutorial, we omit the typing system; instead we illustrate how session types abstract flows of global interactions through concrete examples.

### 3.4 Examples of Session Types

**Example 1.** (Session Type: basics) Consider the following interaction assuming \(adr\) and \(prd\) are variables of string type, located at both Buyer and Seller.

\[
\begin{align*}
\text{Buyer} \rightarrow \text{Seller}: & \quad s_1 \langle \text{quoteReq}, \text{prd}, \text{prd} \rangle. \\
\text{Seller} \rightarrow \text{Buyer}: & \quad s_2 \langle \text{quoteRep}, 100, y \rangle. \\
\text{Buyer} \rightarrow \text{Seller}: & \quad s_1 \langle \text{purchase}, \text{adr}, \text{adr} \rangle.
\end{align*}
\]

The interface which Seller offers (as far as this interaction goes) can be described by the following session type:

\[
\begin{align*}
& s_1 \triangleright \text{quoteReq}(\text{string}). \\
& s_2 \triangleleft \text{quoteRep}(\text{int}). \\
& s_1 \triangleright \text{purchase}(\text{string}).
\end{align*}
\]

the same interaction can be type-abstracted from the viewpoint of Buyer:

\[
\begin{align*}
& s_1 \triangleleft \text{quoteReq}(\text{string}). \\
& s_2 \triangleright \text{quoteRep}(\text{int}). \\
& s_1 \triangleleft \text{purchase}(\text{string}).
\end{align*}
\]

which is the co-type of (2).

**Example 2.** (Session Type: parallel composition) In our types, session channels occur free in session types: this is a generalisation of session types in the literature. For faithfully capturing use cases of web services which exchange different data simultaneously, this extension is necessary to allow multiple session channels to be used in parallel in a single session. Let us show a simple example:

\[
\begin{align*}
& s \triangleleft \text{quote}(\text{int}). \text{end} \mid s' \triangleleft \text{extra}(\text{string}). \text{end}
\end{align*}
\]

Here a participant is sending a quote (integer) at \(s\) and extra information about the product at \(s'\) in a single session: without using distinct session channels, two communications can get confused and result in a type error.

**Example 3.** (Session Type: branching) Let us refine (1) with branching.

\[
\begin{align*}
\text{Buyer} \rightarrow \text{Seller}: & \quad s_1(\text{quoteReq, prd, prd}). \\
\text{Seller} \rightarrow \text{Buyer}: & \quad s_2(\text{quoteRep, 100, y}). \\
& \quad \left( \begin{array}{l}
\text{Buyer} \rightarrow \text{Seller}: s_1(\text{purchase, adr, adr}).0 \\
+ \\
\text{Buyer} \rightarrow \text{Seller}: s_1(\text{nothanks}).0
\end{array} \right)
\end{align*}
\]

\[
\begin{align*}
\text{Buyer} \rightarrow \text{Seller}: s_1(\text{quoteReq, prd, prd}). \\
\text{Seller} \rightarrow \text{Buyer}: s_2(\text{quoteRep, 100, y}). \\
\text{Buyer} \rightarrow \text{Seller}: s_1(\text{purchase, adr, adr}).0 \\
\text{Buyer} \rightarrow \text{Seller}: s_1(\text{nothanks}).0
\end{align*}
\]
This can be abstracted, from the viewpoint of Seller:

\[
\begin{align*}
&\text{s}_1 \rightarrow \text{quoteReq(string). } s_2 \leftarrow \text{quoteRep(int).} \\
&\quad \text{s}_1 \rightarrow \text{(purchase(string). end + nothanks().end)} \quad (5)
\end{align*}
\]

Note the sum + in (5) means the receiving party (here Seller) waits with two options, purchase and nothanks: on the other hand, the co-type of (5) (seen from Buyer’s side) becomes:

\[
\begin{align*}
&\text{s}_1 \leftarrow \text{quoteReq(string). } s_2 \rightarrow \text{quoteRep(int).} \\
&\quad \text{s}_1 \leftarrow \text{(purchase(string). end + nothanks().end)} \quad (6)
\end{align*}
\]

in which the sum + in (5) means that the sending party (here Buyer) may select one of purchase and nothanks from the two options.

Now we go back to the Buyer-Seller interaction in Figure 1 (a). Service channel quoteCh is assigned with the following branching type:

\[
\begin{align*}
&\text{s} \leftarrow \text{quote(integer). } s \rightarrow (\text{accept(null). } s \leftarrow \text{details(string). end +} \\
&\quad \text{reject(null). end)} \quad (7)
\end{align*}
\]

Service channel deliveryCh has type \text{t} \leftarrow \text{details(string). end}.

**Example 4.** (Session Type: recursion) Consider the following recursion.

\[
\text{rec } X.
\begin{align*}
\quad \text{ Buyer} \rightarrow \text{Seller : } s_1(\text{quoteReq, prd, prd).} \\
\quad \text{ Seller} \rightarrow \text{Buyer : } s_2(\text{quoteRep, 100, y).}
\end{align*}
\]

\[
\begin{align*}
&\quad \text{ Buyer} \rightarrow \text{Seller : } s_1(\text{purchase, adr, adr}. \text{0). } + \\
&\quad \text{ Buyer} \rightarrow \text{Seller : } s_2(\text{nothanks). X}
\end{align*}
\]

This behaviour, seen from the viewpoint of Seller, can be abstracted as the following session type:

\[
\begin{align*}
&\text{ rec t. } s_1 \rightarrow \text{quoteReq(string).} \\
&\quad s_2 \leftarrow \text{quoteRep(int).} \\
&\quad s_1 \rightarrow ( \text{purchase(string).end} \\
&\quad \quad + \text{nothanks().t)} \quad (10)
\end{align*}
\]

This type says that it first waits for a quoteReq message at \text{s}_1, to which it replies with quoteRep via \text{s}_2, then it waits for two options purchase and nothanks at \text{s}_1: in the former case it finishes this session while in the latter it recurs to the initial state, waiting for another quoteReq message.

It is notable that we can type the interaction written by the conditional in Figure 1 (b) by a similar recursive type. The typing of the service channel quoteCh will differ in the "rejection" branch, given as:

\[
\begin{align*}
&\text{ rec t. } s \leftarrow \text{quote(integer). } s \rightarrow (\ldots + \text{reject(null).t)}
\end{align*}
\]
4 End-Point Projection

4.1 Basic Ideas

A global description of communication behaviour is useful since it offers a clear view of its dynamic structure. Real execution of the description, however, is always through communication among distributed end-points which, as the notion of choreography dictates (cf. Section 2), may as well involve no centralised control. Thus we ask:

*How can we project a global description to end-point processes so that their interactions precisely realise the original global description?*

To think about this problem is the same thing as relating the global calculus to the $\pi$-calculus: terms in the $\pi$-calculus can be considered as a representation of end-point processes, and we map descriptions in the global calculus to those processes preserving the original semantics. This mapping, called end-point projection (EPP), is non-trivial due to the different nature of descriptions: a global calculus directly describes interactions among multiple participants involving sequencing, branching and recursion, which differs from the end-point description given in the $\pi$-calculus.

Through our investigation of this process, we identified the following three principles for global descriptions under which we can define a correct EPP, *connectedness* (a basic local causality principle), *well-threadedness* (a stronger locality principle based on session types) and *coherence* (a consistency principle for description of each participant in a global description). These notions enunciate, on the basis of session types, general disciplines for well-structured global description to which many business protocols we know from [59] follow. Schematically, the EPP mapping has the following shape:

$$I \mapsto A[P] | B[Q] | C[R] | \cdots$$

where $I$ is a global description, $A$, $B$ and $C$ are *participants* of the protocol and $P$, $Q$ and $R$ are projections of $I$ onto $A$, $B$ and $C$ respectively. We shall show that, when applied to well-typed interactions following the three principles, the EPP mapping thus defined satisfies:

- *type preservation* (the typing is preserved through EPP),
- *soundness* (nothing but behaviours in $I$ are in the image of its EPP) and
- *completeness* (all behaviours in $I$ are in the image).
Thus the resulting processes never have a type error, and they realise all and only interactions prescribed in the original global description. We leave the details of the EPP theory and its three disciplines to [16]. Instead, we shall explain how EPP works by using our running examples, after a brief note on the usage of an EPP theory.

4.2 Usage of End-Point Projection.

The EPP theory maps terms in the global calculus to those in the $\pi$-calculus. By establishing such a link on a formal basis, engineering concerns in global descriptions can be translated into technical questions in the $\pi$-calculus, the latter offering analytical techniques based on a rich accumulation of theoretical results. Similarly, the insight and results in the $\pi$-calculus can be reflected as the engineering ideas in global descriptions.

In addition to all this, having a mathematically well-founded link between two forms of descriptions has essential merits in the engineering of Web Service programming. Some of such engineering scenes include:

- **Interoperability.** Without a general notion of EPP, we cannot relate a global description and end-point behaviour uniformly, losing the essential nature of web services, interoperability.

- **Code generation.** We can create a (perhaps multi-language) complete distributed application by projecting a detailed global description to each of its end-points.

- **Prototype generation.** Projection can also be used for generating a skeleton code for each end-point which only contains basic communication behaviour, to be elaborated to full code. This is already used in [46].

- **Use of conformance.** A team of programmers initially agree on a shared global specification for communications among end-points: during/after programming, each programmer can check if her/his code conforms to the specification by conformance checking against projection. This scheme also applies to conformance of existing services/libraries to a given global scenario.

- **Runtime monitoring, testing and debugging.** At runtime, each end-point can check whether ongoing communications at his/her site conform to the global description by checking them against the projection of the description to that end-point. The monitoring can also be used for debugging and testing existing code.
Networks are parallel composition of participants. The latter is represented by $e$. That and the following two are for communication. This is in the style of [31], except the first two productions for processes describe terms meant for session initiation!

As is well known [42], we can easily encode the primitive into the $\pi$-calculus.

Further, various static and runtime analyses would become available for a global description from their well-developed end-point counterpart.

### 4.3 An End-Point Calculus

We briefly explain the end-point calculus which is a variant of the $\pi$-calculus [42] with locations. Below $P, Q, \ldots$ range over processes and $M, N, \ldots$ range over networks.

$$P ::= \! \text{ch}(\bar{s}).P \mid \bar{c}h(\bar{v}\bar{s}).P \mid s \triangleright \Sigma.\text{op}(y_i) . P_i \mid \bar{s} \triangleleft \text{op}(e).P \mid x := e . P \mid \text{if } e \text{ then } P_1 \text{ else } P_2 \mid P_1 \oplus P_2 \mid P_1 \mid P_2 \mid (vs)P \mid X \mid \text{rec } X . P \mid 0$$

$$N ::= A[P], N_1 | N_2 | (vs)N \mid \epsilon$$

The first two productions for processes describe terms meant for session initiation and the following two are for communication. This is in the style of [31], except that $y_i$ in the second construct (branching input) is not bound in $P_i$. The term $x := e . P$ assigns a value $v$ to $x$ in its store and then continues as $P$. The rest is standard. Networks are parallel composition of participants. The latter is represented by $A[P]$, which indicates a participant $A$ whose behaviour is given by $P$ and whose local state is $\sigma$. We often omit $\sigma$ when irrelevant.

The reduction semantics for the end-point calculus follows the $\pi$-calculus. We list the three key rules in Table 2. Other rules are standard. (E-Inrr) defines the session initiation: two participants $A$ and $B$ will synchronise to start a session, $\! \text{ch}(\bar{s}).P$ denoting a service and $\bar{c}h(\bar{v}\bar{s}).Q$ a request. The synchronisation will result in sharing fresh session names $\bar{s}$ local to $A$ and $B$. These session names are then used in (E-Com) for communication. In (E-Com), communicated values are assigned to local variables, rather than get substituted, for having the correspondence with the global calculus. (E-Asgn) updates a local store.

---

**Table 2** Semantics of the End-Point Calculus

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>(E-Inrr) $A[! \text{ch}(\bar{s}).P</td>
<td>P' \parallel \rho</td>
</tr>
<tr>
<td>(E-Com) $A[s \triangleright \Sigma.\text{op}(x_i) . P_i</td>
<td>P' \parallel \rho</td>
</tr>
<tr>
<td>(E-Asgn) $A[x := e . P</td>
<td>P' \parallel \rho]$</td>
</tr>
</tbody>
</table>

---

2For simplicity of the end-point projection, we added the assignment primitive to the calculus. As is well known [42], we can easily encode the primitive into the $\pi$-calculus.
Examples of the End-Point Projection. Let us recall our running example, Figure 1 (a) in § 3.1. An end-point representation of this example for Buyer may be written:

\[
\text{Buyer}\left[\right.\text{quoteCh}\left(\nu s\right), s \triangleright quote\left(x\right) . (\overline{s} \triangleright accept . s \triangleright details\left(y\right) . 0 \oplus \overline{s} \triangleright reject . 0) \left.]\right. \\
\]

Above Buyer[\(P\)] indicates a participant (a named agent) whose behaviour is given by the process \(P\). The Seller’s code is given as:

\[
\text{Seller}\left[!\text{quoteCh}\left(s\right), s \triangleright \langle 300 \rangle . s \triangleright \langle \text{accept} . \text{deliveryCh}\left(\nu t\right) . t \triangleright \text{delivery}\left(x\right) . \overline{s} \triangleright \text{delivery}\left(x\right) . 0 + \text{reject} . 0 \rangle] \right. \\
\]

The end-point representation for Shipper can be given similarly. Such an end-point description precisely depicts each local communication behaviour. The two service channels \text{quoteCh} and \text{deliveryCh} are replicated and ready to receive invocations, following (SCP). In the end-point processes, the session types explicitly show the direction of interaction on channels. Let \(\alpha_{1}\) for the type of \text{quoteCh} and \(\alpha_{2}\) for \text{deliveryCh} which is appeared in Example 3. Then the type of the seller above has type \(\alpha_{1}\) for \text{quoteCh}, but \(\alpha_{2}\) for \text{deliveryCh} indicating the direction of interactions: this is because the output channel is located at the shipper’s. In the global calculus, a channel is always used for both input and output, so there is no such need.

Similarly we may type the end-point processes for Buyer and Seller with recursion, corresponding to Figure 1 (b), as:

\[
\text{Buyer}\left[\right.\text{rec } X . \text{quoteCh}\left(\nu s\right), s \triangleright quote\left(x\right) . \\
\text{if reasonable}\left(x\right) \text{then } \overline{s} \triangleright accept . s \triangleright details\left(y\right) . 0 \text{ else } \overline{s} \triangleright reject . X \left.]\right. \\
\text{Seller}\left[!\text{quoteCh}\left(s\right), s \triangleright \langle 300 \rangle . \text{rec } X . \overline{s} \triangleright \text{quote}\left(\nu t\right) . t \triangleright \text{delivery}\left(x\right) . \overline{s} \triangleright \text{delivery}\left(x\right) . 0 + \text{reject} . X\rangle] \right. \\
\]

Note both in its term and in its typing, the end-point process for Shipper in Figure 1 (b) does not involve recursion since its session is self-contained inside a recursion.

A significant property of typed end-point processes is the lack of communication error in the sense that typed processes never invoke missing operations and communicate ill-typed values. This is fundamental at end-point level since it describes inputs and outputs separately. Via the EPP, we can guarantee that a global description which conforms the three disciplines does not have communication error of its end-point implementation.
5 Related Work and Future Topics

Global Descriptions of Communication Behaviour. Global methods for describing communication behaviour have been practiced in several different engineering scenes in addition to WS-CDL (for which this work is intended to serve as its theoretical underpinning). Representative examples include the standard notation for cryptographic protocols [43], message sequence charts (MSC) [35, 29, 7], and UML sequence diagrams [45]. These notations are intended to offer a useful aid at the design/specification stage, and do not offer full-fledged programming language, lacking, or example standard control structures and/or value passing.

DiCons (which stands for “Distributed Consensus”), which is independently conceived and predates WS-CDL, is a notation for global description and programming of Internet applications introduced and studied by Baeton and others [8], and would be the first fully expressive language for representing interactions based on a global method. DiCons uses programming primitives close to user’s experience in the web, such as web server invocation, email, and web form filing, rather than general communication primitives. Its semantics is given by either MSCs or direct operational semantics. DiCons does not use session types or other channel-based typing. An analogue of the theory of end-point projection has not been developed in the context of DiCons.

Petri Nets may also be viewed as offering a global description, though again they are more useful as a specification/analytical tool. As an example, a study by van der Aalst [56] presents an analysis of a business protocol showing how a description of an inter-participant business protocol can be implemented inside each participant without losing causal constraint, all represented in Petri Nets. While quite different in the formal apparatus and motivations, it shares a technical interest with our analysis in Section 4 as a causality analysis of interactions. The current lack of notions of types in Petri Nets may make it hard to carry out the analogue of the full constructions as done in our work.

As we noted, global notations are often used for representing security protocols. Strand Space [28] is a structure for analysing properties of cryptographic protocols. It models protocols as causal chains of interactions, and is often presented in a global notation similar to UML sequence diagrams. Strand space does not by itself offer a fully expressive description language with general control constructs. Their methods for security analysis may be applicable to our global calculus. Briais and Nestmann [13] present a global notation for representing protocol narrations and relate it to the π-calculus. Since their sole focus is on cryptographic aspects, their global formalism is not intended as a fully expressive language for describing interactions, lacking for example, channels, conditional and loops, as well as type disciplines for interactions.
The Bulletin of the EATCS

Theories of Types for the π-Calculus. Many theories of types for the π-calculus are studied. In this work, we used session types; session types provide a flexible programming style for structural interaction, and are used to statically check the safe and consistent composition of protocols in communication-centric distributed software. The session types also play a key role of EPP theory. In the context of session types, the present work extends the session structure with multiple session names which is useful for having parallel communications inside a session. In addition to the study of session types, these include input/output types [40, 47], linear types [37, 30], various kinds of behavioural types which include causality [34, 61, 60] and combination of behavioural types and model checking for advanced behavioural analysis [18, 49], to name a few. Behavioural types offer an advanced analyses for such phenomena as deadlock freedom. We are currently studying how these advanced type-based validation techniques on the basis of the present simple session type discipline will lead to effective validation techniques.

Type Disciplines for Concurrent Programming Languages. Our work shares with many recent works its direction towards well-structured communication-centred programming using types. Pict [48] is the programming language based on the π-calculus, with rich type disciplines including linear and polymorphic types. Polyphonic C♯ [9] uses a type discipline for safe and sophisticated object synchronisation. Gay, Vasconcelos, Dezani, Drossopoulou and others have studied interplay of session type disciplines with different programming constructs and program properties [54, 31, 24, 20, 57, 12]. More recently, a similar approach to the session types has been studied in the Singularity OS [22]; behaviours in this system are defined in the form of a state machine which specifies desired message exchange patterns. Messages encapsulate asynchronous method invocation, and consist of information on which method should be invoked, along with the actual arguments to use when the message received. Our end-point calculus is based on synchronous communication following the previous works on the session types; we believe all the technical results can be adapted to the asynchrony.

The EPP theory offers a passage through which these studies (all based on end-point languages and calculi) can be reflected onto global descriptions, as we have demonstrated for session types in the present work.

Process Calculus-Based Analysis of Web Service. Foster, Magee, Kramer and others studied a transition-based approach [1] for checking the correctness of the protocols written in CDL including absence of deadlock and livelock. Their approach is limited in untyped (data and message are abstracted and represented simply as base types) and communication without name passing. Extending their model abstraction to support data variable expression and message correlation
(state, types and channels) based on the syntax and semantics of the π-calculus is an interesting future topic. In general, pre-type checking would provide opportunity to reducing the size of static state machines in model checking.

Gordon, Fournet, Bhargavan, Pucella and Corin studied security-related aspects of web services in their series of works (whose origin lies in the the spi-calculus [6] and the applied π-calculus [5]). Their initial work [26] may be the first to apply the π-calculus to Web Services, focussing on security concern. In a more recent work [11], the authors have implemented part of WS-Security libraries using a dialect of ML, and have shown how annotated application-level usage of these security libraries in web services can be analysed with respect to their security properties by translation into the applied π-calculus. The benefits of such a tool can be reflected onto the global descriptions through the theory of EPP, by applying the tool to projections.

Laneve and Padovani [38] give a model of orchestrations of web services using an extensions of π-calculus to join patterns. They propose a typing system for guaranteeing a notion of smoothness i.e. a constraint on input join patterns such that their subjects (channels) are co-located in order to avoid a classical global consensus problem during communication. Reflecting the centralised nature of orchestration, neither a global calculus nor end-point projection is considered.

A bisimulation-based correspondence between choreography and orchestration in the context of web services has been studied in [14] by Busi and others, where a notion of state variables is used in the semantics of the orchestration model. They operationally relate choreography to orchestration. In [27], the same authors introduce SOCK, a calculus for web services based on end-point descriptions. This formalism models networks of participants and their local behaviour enhanced with a mechanism for controlling communication at run-time based on logical conditions on the participant’s store. In our work, communication structures are represented by session types, hence interaction is statically controlled and its error freedom is guaranteed by a sound type system. Their dynamic control allows concrete reasoning on the communicated values, but requires heavy run-time mechanisms. It is an interesting topic to integrate static and dynamic methods for efficient and flexible control of interactions. Neither strong type systems for communication nor descriptive principles for end-point projection are studied in their work.

Providing a formal foundation of WS-CDL based on process calculi has been suggested since the inception of WS-CDL, which led to invitation of π-calculus experts to the WS-CDL WG. Kavantzas [36] suggested the use of a calculus of mobile processes for modelling WS-CDL. Ross-Talbot and Brown are instrumental in the use of the π-calculus as a formal basis of WS-CDL. While we could have used any name passing calculi with enough expressive power, as formalisms for end-point processes, our choice of an imperative extension of the π-calculus
The Bulletin of the EATCS

has the merit in that we can have a direct, transparent correspondence between
the presented global formalism and the end-point formalism. A semantic preserv-
ing encoding of this applied version of the $\pi$-calculi into the pure $\pi$-calculus (or
other pure name passing calculi, including fusion calculi) is an interesting topic
for future study.

Extensions and Future Topics. Global descriptions have been practiced in var-
ious engineering contexts for a long time. The present work is a trial to realise
its potential as a mathematically well-founded programming method, centring on
type structures for communication. For practical applications, the EPP theory de-
mands further exploration, capturing all basic concurrent programming primitives.
In particular, channel passing is a useful extension for business protocols, for ex-
ample in the scenarios where participants need to send links to other participants.
A possible extension of the present EPP theory to channel passing is discussed
in [17, §17]. Other interesting extensions include broadcasting, various notions
of partial failure, richer primitives for internal calculation (such as a control of
shared resources by locking), exceptions, timeout, code passing, and integration
with various meta-level operators.

Acknowledgements. We thank Robin Milner for organising the current sci-
entific advisory team for W3C WS-CDL WG; the WG members, in particular
Gary Brown, Nickolas Kavantzas and Steve Ross-Talbot, for collaborations; Ross-
Talbot for his text of the history of WS-CDL; and Andrew Gordon for his helpful
comments on an early version of this tutorial. This work is supported by EPSRC
GR/T04236, GR/S55545, GR/S55538, GR/T04724, GR/T03208, GR/T03258 and
IST2005-015905 MOBIUS.

References
POPL ’01: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Princi-
Press.


The Bulletin of the EATCS


The Bulletin of the EATCS


TECHNICAL CONTRIBUTIONS
The Freudenthal problem and its ramifications (Part II)

Axel Born ♠ Cor A.J. Hurkens † Gerhard J. Woeginger †

I don’t know, I don’t know.
—George Harrison, “Something”

And I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know, I know.
—Bill Withers, “Ain’t No Sunshine”

This is the second article (in a series of three) dedicated to the many variants and variations of the Freudenthal problem. The common crux in all these variants is how to draw the right conclusions from a strange conversation that mainly consists of “I-do-not-know” and “Now-I-do-know” statements. The Freudenthal problem dates back to the year 1969, when Hans Freudenthal posed it in the Nieuw Archief voor Wiskunde [5]. Nowadays the problem is famous all over the world. It regularly shows up in mathematical puzzle corners, and analyzing the flow of information in the underlying conversation has become a standard exercise for computer science students. In a predecessor paper [2], we have surveyed the Freudenthal problem in its classical form and some of its most basic variations. In the current paper, we continue our survey by discussing some of its more remote relatives. Each of the following seven sections introduces one relative or one class of relatives: We give a complete formulation of the respective problem. We state the respective solution. And we try to present a more or less complete derivation of the solution.

♠Oberstufen-Realgymnasium Ursulinen, Leonhardstrasse 62, 8010 Graz, Austria.
†Email: {wscor|gwoegi}@win.tue.nl. Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
We admit that, nonetheless, some of our derivations are not easily verified without a computer.

Before we begin, some remarks on the conversations in this paper: We assume that every speaker is truthful, and has very fast and perfect reasoning skills; furthermore, everybody knows that this holds for all the other speakers, too. We assume that in every statement, the speaker reveals something new about the state of his knowledge; we sometimes stress this by putting an ‘Aha’ at the beginning of a statement; this also means that the corresponding information is revealed at the earliest possible moment in time.

1 Ferguson’s variant

Thomas Ferguson [3, 4] constructed a beautiful Freudenthal variant around the sum of two integers, and the sum of their squares. Note that his problem does not state any upper bounds on the involved numbers!

The teacher tells Sam and Quinn: I have secretly chosen two integers $x$ and $y$ with $1 \leq x \leq y$. I have told their sum $s = x + y$ to Sam and the sum of their squares $q = x^2 + y^2$ to Quinn.

1. Quinn says: I don’t know the numbers.
2. Sam says: I don’t know the numbers.
3. Quinn says: I don’t know the numbers.
4. Sam says: I don’t know the numbers.
5. Quinn says: I don’t know the numbers.
6. Sam says: I don’t know the numbers.
7. Quinn says: Aha. Then I do know the numbers.

Determine $x$ and $y$!

We say that a sum $s$ and a sum of squares $q$ are compatible, if there exist two positive integers $x$ and $y$ with $x + y = s$ and $x^2 + y^2 = q$. Let $Q_1$ denote the set of all integers $q$ that are compatible with two distinct sums (or equivalently: the set of all integers that can be written in two different ways as the sum of two squares). For instance $1^2 + 7^2 = 5^2 + 5^2$ demonstrates that $50 \in Q_1$, and $1^2 + 8^2 = 4^2 + 7^2$ demonstrates that $65 \in Q_1$.

Table 1 lists for each sum $s \leq 21$ all the compatible sums of squares $q \in Q_1$: If the entry at the crossing of row $q$ and column $s$ is in $\{+, 2, 3, 4, 5, 6\}$, then $s$ and $q$ are compatible. If this entry is $-$, then they are not compatible. The table
The Bulletin of the EATCS

Table 1: Compatibility between sums $s \leq 21$ and values $q \in Q_1$.

<table>
<thead>
<tr>
<th>$q \backslash s$</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>$\geq 22$</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>65</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>85</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>125</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>130</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>145</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>170</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>185</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>200</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>205</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>221</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>250</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>260</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>265</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>290</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>305</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>325</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>365</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

$suppresses the columns $s \leq 21$ that are not compatible with any $q \in Q_1$. The last column in the table states whether a row $q \in Q_1$ is compatible with any of the sums $s \geq 22$. Here is a lemma that provides more information on these sums $s \geq 22$.

**Lemma 1.** Every sum $s \geq 22$ is compatible with two distinct $q_1, q_2 \in Q_1$.

**Proof.** For two positive integers $a, b$ with $a > 2b$, we define $q(a, b) := 5(a^2 + b^2)$. The identity $q(a, b) = (a + 2b)^2 + (2a - b)^2 = (a - 2b)^2 + (2a + b)^2$ demonstrates that every value $q(a, b)$ belongs to $Q_1$, since it is compatible with the two sums $3a + b$ and $3a - b$. Now consider an arbitrary sum $s \geq 22$. If $s = 3k$, then choose $q_1 = q(k - 1, 3)$ and $q_2 = q(k + 1, 3)$. If $s = 3k + 1$, choose $q_1 = q(k, 1)$ and $q_2 = q(k + 1, 2)$. If $s = 3k + 2$, choose $q_1 = q(k, 2)$ and $q_2 = q(k + 1, 1)$.

After all these preparations, Ferguson’s problem has lost its mystery: Sam and Quinn are essentially manipulating a huge, infinite, two-dimensional table where columns correspond to sums and rows correspond to sums of squares. If a row has only a single surviving compatible column, then it is killed by Quinn’s next I-don’t-know statement. And symmetrically, if a column has only a single surviving
compatible row, then the next I-don’t-know by Sam removes this column from the table. An integer entry $\ell$ in Table 1 means that statement #\ell collides with this entry and kills the corresponding row (if $\ell$ is odd) or column (if $\ell$ is even).

Here is how the plot develops: Quinn’s statement #1 implies that $q$ lies in $Q_1$, and it kills off all the remaining rows. Sam’s statement #2 yields that $s \neq 8$, $s \neq 9$, $s \neq 10$, and $s \neq 15$. Table 1 and Lemma 1 show that all the other columns stay alive, since they are compatible with at least two surviving rows. Quinn’s statement #3 implies $q \neq 65$ and $q \neq 125$. Sam’s statement #4 yields $s \neq 11$. Quinn’s statement #5 yields $q \neq 85$. Sam’s statement #6 implies $s \neq 13$. At the moment when Quinn makes his statement #7, the surviving row $q = 145$ is only compatible with the surviving column $s = 17$, whereas all other surviving rows are compatible with two or more surviving columns. Only in case Quinn’s number is $q = 145$, he is able to make his statement #7. Therefore $q = 145$ and $s = 17$, and the secret numbers in Ferguson’s problem are $x = 8$ and $y = 9$.

We conclude this section with four small observations. First: If Sam already knows the numbers in statement #6, then the solution would be $x = 1$ and $y = 12$ instead. Second: If Quinn already knows the numbers in his statement #5, then the solution would be $x = 6$ and $y = 7$ instead. Third: If Sam already knows the numbers in his statement #4, then the solution would be $x = 2$ and $y = 9$ instead. Fourth: If Sam or Quinn would already know the numbers in statement #1, #2, or #3, then the solution would not be unique any more.

2 Berloquin’s variant

The following Freudenthal variant is called Berloquin, since we have found it in the book [1] by Pierre Berloquin. We have slightly modified it, and removed Berloquin’s upper bound constraint $x, y \leq 99$. The Freudenthal variant Berloquin results from replacing the constraint $2 \leq x \leq y$ by the more restrictive $2 \leq x < y$.

The teacher says to Peter and Sam: I have secretly chosen two integers $x$ and $y$ with $2 \leq x \leq y$. I have told their sum $s = x + y$ to Sam and their product $p = xy$ to Peter.

1. Sam says: I don’t know the numbers.
2. Peter says: I don’t know the numbers.
3. Sam says: Aha. Then I do know the numbers.
4. Peter says: Aha. Then I also know them.

Determine $x$ and $y$!
This puzzle Berloquin\textsuperscript{*} has a short and simple solution: Sam’s statement #1 implies \( s \geq 6 \), and doesn’t imply anything else. Peter’s statement #2 implies that \( p \) has two distinct factorizations \( p = xy \) with \( 2 \leq x \leq y \) and \( x + y \geq 7 \). Let \( \mathcal{P}_2 \) denote the set of all products that agree with statement #2. Sam’s statement #3 shows that \( s \geq 8 \) cannot hold: For \( s \geq 8 \), the two products \( 4(s - 4) = 2(2s - 8) \) and \( 6(s - 6) = 2(3s - 18) = 3(2s - 12) \) are both in \( \mathcal{P}_2 \). Hence Sam could not distinguish between the situation \( x = 4, y = s - 4 \) and the situation \( x = 6, y = s - 6 \) in statement #3. We conclude that \( s = 6 \) or \( s = 7 \) holds. The case \( s = 6 \) is impossible, since \( 6 = 2 + 4 = 3 + 3 \) and \( 2 \cdot 4 \notin \mathcal{P}_2 \) and \( 3 \cdot 3 \notin \mathcal{P}_2 \). In the remaining case \( s = 7 = 2 + 5 = 3 + 4 \), we have \( 2 \cdot 5 \notin \mathcal{P}_2 \) and \( 3 \cdot 4 \notin \mathcal{P}_2 \). Therefore, the unique solution for Berloquin\textsuperscript{*} is given by \( x = 3 \) and \( y = 4 \).

The puzzle Berloquin\textsuperscript{*} needs more work: Sam’s statement #1 this time implies \( s \geq 7 \). Peter’s statement #2 implies that \( p \) has two distinct factorizations \( p = xy \) with \( 2 \leq x < y \) and \( x + y \geq 7 \), and we denote the corresponding set of products by \( \mathcal{P}_2 \). Sam’s statement #3 excludes most of the remaining values for \( s \):

- \( s = 9 \) is impossible: Sam would neither be able to exclude the partition \( 9 = 3 + 6 \) (with product \( 18 \in \mathcal{P}_2 \)), nor the partition \( 9 = 4 + 5 \) (with product \( 20 \in \mathcal{P}_2 \)).
- \( s = 11 \) is impossible: Sam would neither be able to exclude the partition \( 11 = 2 + 9 \) (with product \( 18 \in \mathcal{P}_2 \)), nor the partition \( 11 = 3 + 8 \) (with product \( 24 \notin \mathcal{P}_2 \)).
- \( s = 12 \) is impossible: Sam would neither be able to exclude the partition \( 12 = 2 + 10 \) (with product \( 20 \in \mathcal{P}_2 \)), nor the partition \( 12 = 4 + 8 \) (with product \( 32 \in \mathcal{P}_2 \)).
- All cases \( s \geq 13 \) are impossible: The products \( 4(s - 4) = 2(2s - 8) \) and \( 6(s - 6) = 3(2s - 12) \) are both in \( \mathcal{P}_2 \). Hence, Sam would neither be able to exclude the case \( x = 4, y = s - 4 \), nor the case \( x = 6, y = s - 6 \).

We denote by \( S_1 \) the set of all sums \( s \) that are compatible with the first three statements. The above case analysis implies \( S_1 \subseteq \{7, 8, 10\} \). The following case analysis shows that \( S_1 = \{7, 8, 10\} \):

- For \( s = 7 \) the possible partitions are \( 7 = 2 + 5 \) and \( 7 = 3 + 4 \). Since \( 10 \notin \mathcal{P}_2 \) and \( 12 \in \mathcal{P}_2 \), this case would imply \( x = 3, y = 4 \) and \( p = 12 \).
- For \( s = 8 \) the possible partitions are \( 8 = 2 + 6 \) and \( 8 = 3 + 5 \). Since \( 12 \in \mathcal{P}_2 \) and \( 15 \notin \mathcal{P}_2 \), this case would imply \( x = 2, y = 6 \) and \( p = 12 \).
- For \( s = 10 \) the possible partitions are \( 10 = 2 + 8 \), \( 10 = 3 + 7 \), and \( 10 = 4 + 6 \). Since \( 16 \notin \mathcal{P}_2 \), \( 21 \notin \mathcal{P}_2 \) and \( 24 \in \mathcal{P}_2 \), this case would imply \( x = 4, y = 6 \) and \( p = 24 \).
Peter’s statement #4 indicates that he is able to pick the right sum $s$ from $S_3$. This yields $p = 24$, $s = 10$ and $x = 4$, $y = 6$ as the unique solution to problem Berloquin$^*$.

We observe that Berloquin$^*$ and Berloquin$^{**}$ are of exceptional beauty and elegance. Their statement is crisp and concise, and does not introduce any artificial constraints nor upper bounds on the involved integers. The reader should contrast them with the following (less appealing and somewhat boring) variant that explicitly introduces $x, y \leq 8$ and furthermore reveals $s = 8$.

The teacher says to Peter and Sam: I have secretly chosen two integers $x$ and $y$ with $2 \leq x \leq y \leq 8$. I have told their sum $s = x + y$ to Sam and their product $p = xy$ to Peter.

1. Peter says: I don’t know the numbers.
2. Sam says: I don’t know the numbers.
3. Peter says: Aha. Then I know that $s = 8$.
4. Sam says: Aha. Then I know the numbers.

Determine $x$ and $y$!

Statement #1 yields that the product lies in $\mathcal{P}_1 = \{12, 16, 24\}$, and statement #2 yields that the sum lies in $\mathcal{S}_2 = \{8, 10\}$. If $p = 16$, then Peter cannot make statement #3, since he cannot distinguish between $p = 2 \cdot 8$ and $p = 4 \cdot 4$. If $p = 24$, then Peter cannot claim the incompatible sum $s = 8$ in statement #3. Therefore $p = 12$ and $s = 8$ must hold, which yields $x = 2$ and $y = 6$.

3 Friedman’s variant

In this section we discuss another class of Freudenthal problems where the conversations mainly consist of statements of the type “I don’t know the numbers”. Their length can be formidable. For instance, Erich Friedman posed the following problem Friedman(1, 9, 9) in December 2000 at the MathPuzzle website www.mathpuzzle.com.

The teacher says to Peter and Sam: I have secretly chosen two integers $x$ and $y$ with $1 \leq x \leq y \leq 9$. I have told their sum $s = x + y$ to Sam and their product $p = xy$ to Peter.

1. Peter says: I don’t know the numbers.
2. Sam says: I don’t know the numbers.
3. Peter says: I don’t know the numbers.
4. Sam says: I don’t know the numbers.
5. Peter says: I don’t know the numbers.
6. Sam says: I don’t know the numbers.
7. Peter says: I don’t know the numbers.
8. Sam says: I don’t know the numbers.
9. Peter says: Aha. Then I do know the numbers.

Determine \( x \) and \( y \)!

The parameters \( m, M, r \) of the general problem \( \text{FRIEDMAN}(m, M, r) \) satisfy \( 1 \leq m < M \) and \( r \geq 2 \). The above formulation changes as follows: The upper and lower bounds on \( x \) and \( y \) become \( m \leq x \leq y \leq M \). The conversation is started by Peter and consists of \( r \) rounds. In the first \( r - 1 \) rounds Peter and Sam alternately tell each other that they still do not know the numbers, and in the last round Peter (if \( r \) is odd) or Sam (if \( r \) is even) announces that he has finally identified the numbers.

Now let us work through \( \text{FRIEDMAN}(1, 9, 9) \). Peter’s statement \#1 implies that the product \( p \) does not have a unique factorization \( p = xy \) with \( 1 \leq x \leq y \leq 9 \). Hence, \( p \) is from the set \( P_1 = \{4, 6, 8, 9, 12, 16, 18, 24, 36\} \). Table 2 demonstrates that every element of \( P_1 \) in fact has precisely two such factorizations.

<table>
<thead>
<tr>
<th>( p ) ( \backslash ) ( s )</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2;2</td>
<td>1;4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>2;3</td>
<td>-</td>
<td>1;6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
<td>2;4</td>
<td>-</td>
<td>-</td>
<td>1;8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3;3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1;9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3;4</td>
<td>2;6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4;4</td>
<td>-</td>
<td>2;8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3;6</td>
<td>-</td>
<td>2;9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4;6</td>
<td>3;8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>36</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6;6</td>
<td>4;9</td>
</tr>
</tbody>
</table>

Table 2: The factorizations of products \( p \in P_1 \) and their sums.

Sam’s statement \#2 yields that the sum lies in \( S_2 = \{5, 6, 7, 8, 9, 10, 11\} \). In particular \( s = 4 \) is impossible (since Sam would deduce \( x = y = 2 \)), \( s = 12 \) is impossible (since Sam would deduce \( x = y = 6 \)), and also \( s = 13 \) is impossible (since Sam would deduce \( x = 4 \) and \( y = 9 \)). This kills the corresponding three columns in Table 2. Similarly, we see that Peter’s statement \#3 gives \( p \neq 4 \);
that Sam’s statement #4 yields \( s \neq 5 \); that Peter’s statement #5 implies \( p \neq 6 \); that Sam’s statement #6 implies \( s \neq 7 \); that Peter’s statement #7 yields \( p \neq 12 \); and finally that Sam’s statement #8 yields \( s \neq 8 \). At the moment just before statement #9, the table has boiled down to \( p \in P_7 = \{8, 9, 16, 18, 24\} \) and \( s \in S_8 = \{6, 9, 10, 11\} \). Only \( p = 16 \) allows Peter to make his statement #9. This yields \( x = 2 \) and \( y = 8 \) as unique solution of problem Friedman(1,9,9).

Apparently, there are gazillions of puzzles Friedman\((m,M,r)\) that possess a unique solution; some of them are listed in Table 3. Note that as soon as we fix the two values \( m \) and \( M \), we also bound the number \( r \) of statements (in any meaningful conversation) from above. For instance for \( m = 1 \) and \( M = 9 \), another I-don’t-know by Peter in statement #9 would make the conversation stall and go on forever; therefore all problems Friedman\((1,9,r)\) with \( r \geq 10 \) are ill-posed. The following question is open, and might be very difficult to settle.

**Open problem 2.** Prove or disprove: For every \( R \geq 1 \), there exist integers \( m, M, r \) with \( 1 \leq m < M \) and \( r \geq R \), such that Friedman\((m,M,r)\) has a unique solution.

<table>
<thead>
<tr>
<th>( r )</th>
<th>( m )</th>
<th>( M )</th>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>99</td>
<td>77</td>
<td>84</td>
</tr>
<tr>
<td>15</td>
<td>66</td>
<td>99</td>
<td>77</td>
<td>84</td>
</tr>
<tr>
<td>15</td>
<td>301</td>
<td>369</td>
<td>320</td>
<td>342</td>
</tr>
<tr>
<td>17</td>
<td>286</td>
<td>343</td>
<td>288</td>
<td>343</td>
</tr>
<tr>
<td>20</td>
<td>177</td>
<td>234</td>
<td>198</td>
<td>210</td>
</tr>
<tr>
<td>21</td>
<td>177</td>
<td>233</td>
<td>189</td>
<td>220</td>
</tr>
</tbody>
</table>

Table 3: Some puzzles Friedman\((m,M,r)\) with unique solutions.

## 4 Kraus’ variant with the four digits

The MathPuzzle website [www.mathpuzzle.com](http://www.mathpuzzle.com) mentions a nice and very elaborate Freudenthal variant that has been designed by Robert Kraus. It is centered around three persons and four secret digits. Whenever in this section some statement is made by two or more persons, we assume that they all speak simultaneously. Hence, the information revealed by one of the speakers will not affect the statements of the other simultaneous speakers.

The teacher tells Peter, Quinn, and Vince: I have secretly chosen four digits \( a, b, c, d \) with \( 1 \leq a < b < c < d \leq 9 \). I have told their product \( p = abcd \) to Peter, the sum of their squares \( q = a^2 + b^2 + c^2 + d^2 \) to Quinn, and the value \( v = ab + cd \) to Vince.
The Bulletin of the EATCS

1. Peter, Quinn, and Vince say: I don’t know the digits.
2. Peter, Quinn, and Vince say: I still don’t know the digits.
3. Peter, Quinn, and Vince say: I still don’t know the digits.
4. Peter, Quinn, and Vince say: Aha. Now I do know the digits.

Determine \( a, \ b, \ c, \text{ and } d! \)

Solving this problem without the help of a computer is very time-consuming. Here is the sketch of a solution: Let \( Q_0 \) denote the set of all 126 quadruples \((a, b, c, d)\) that agree with the initial statement of the teacher. The quadruples in \( Q_0 \) take 79 different \( p \)-values, 93 different \( q \)-values, and 69 different \( v \)-values. But 41 of these \( p \)-values show up for only a single quadruple, and hence would allow Peter to identify the numbers already in statement \#1. Similarly, 67 of the \( q \)-values and 32 of the \( v \)-values show up for a unique quadruple in \( Q_0 \), and thus also collide with statement \#1. Generally speaking, statement \#\( \ell \) (where \( \ell = 1, 2, 3 \)) eliminates from set \( Q_{\ell - 1} \) all the quadruples that have a unique \( p \)-value or \( q \)-value or \( v \)-value; this yields a new set \( Q_\ell \) of surviving quadruples. It can be seen that statement \#1 altogether eliminates 83 quadruples from \( Q_0 \), that statement \#2 eliminates 21 quadruples, and that statement \#3 eliminates 15 quadruples.

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( d )</th>
<th>( p )</th>
<th>( q )</th>
<th>( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>120</td>
<td>78</td>
<td>34</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>168</td>
<td>102</td>
<td>46</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>336</td>
<td>150</td>
<td>62</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>168</td>
<td>78</td>
<td>34</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>240</td>
<td>102</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>336</td>
<td>126</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>756</td>
<td>155</td>
<td>75</td>
</tr>
</tbody>
</table>

Table 4: The set \( Q_3 \) in the four-digit puzzle by Robert Kraus.

The resulting set \( Q_3 \) consists of the 7 quadruples listed in Table 4. Note that 34, 46, and 62 each occur twice among the \( v \)-values, whereas 75 occurs only once. Statement \#4 by Vince implies \( v = 75 \). Since also Peter’s corresponding product \( p = 756 \) and Quinn’s corresponding sum of squares \( q = 155 \) are unique within \( Q_3 \), the solution of this puzzle by Robert Kraus is given by \((a, b, c, d) = (3, 4, 7, 9)\).

We have made the above puzzle even more elaborate, and generalized it to four persons and five secret numbers. In the new puzzle stated below, there are 252 quintuples \((a, b, c, d, e)\) that agree with the initial statement of the teacher. A computer program demonstrates that, statement by statement, the number of
surviving quintuples drops down to 140, 100, 85, 73, 64, 62, 60, 57, 54, 50, 47, 44, 40, 36, 33, 31, 28, 24, 19, 13, 8, 4, and finally to 1. After statement #23 the quintuple (2, 5, 6, 7, 8) is the only survivor. Peter, Quinn, Sam, and Vince recognize this, and the game is over:

The teacher tells Peter, Quinn, Sam, and Vince: I have secretly chosen five integers \(a, b, c, d, e\) with \(1 \leq a < b < c < d < e \leq 10\). I have told their product \(p = abcd\) to Peter, the sum of their squares \(q = a^2 + b^2 + c^2 + d^2 + e^2\) to Quinn, their sum \(s = a + b + c + d + e\) to Sam, and the value \(v = (a + b + c)(d + e)\) to Vince.

1. Peter, Quinn, Sam, and Vince say: I don’t know the numbers.
2. Peter, Quinn, Sam, and Vince say: I don’t know the numbers.
3. Peter, Quinn, Sam, and Vince say: I don’t know the numbers.
4. Peter, Quinn, Sam, and Vince say: I don’t know the numbers.
   ...
   ...
   ...
22. Peter, Quinn, Sam, and Vince say: I don’t know the numbers.
23. Peter, Quinn, Sam, and Vince say: I don’t know the numbers.

Determine \(a, b, c, d\) and \(e\)!

A final remark: If the teacher tightens his constraint from \(1 \leq a < b < c < d < e \leq 10\) to \(1 \leq a < b < c < d < e \leq 8\), and if Peter, Quinn, Sam, and Vince state only twice (instead of 23 times) that they do not know the numbers, then \((1, 3, 4, 5, 8)\) becomes the new unique solution.

\section{A variant with two digits and many divisors}

For an integer \(n\), we denote by \(\tau(n)\) the number of its positive integer divisors, including the divisors 1 and \(n\). When we talk about the digits of \(n\), we of course mean the digits in the decimal representation of \(n\). Here is a cute Freudenthal puzzle.

The teacher says to Sam and Ted: I have secretly chosen a 2-digit integer \(n\) from the range \(10 \leq n \leq 99\). I have told Sam the sum \(s\) of the two digits in \(n\), and I have told Ted the number \(t = \tau(n)\) of divisors of \(n\).

1. Ted says: I don’t know \(n\), but I know whether it’s odd or even.
The Bulletin of the EATCS

2. Sam says: Aha. Then I do know \( n \).
3. Ted says: Aha. Then I also know \( n \).

Determine the number \( n \)!

Any 2-digit integer \( n \) satisfies \( 2 \leq \tau(n) \leq 12 \). Ted’s statement #1 contains a lot of additional information on \( t \), and allows us to exclude the following values:

- \( t = 4 \) is impossible, since \( \tau(10) = \tau(15) = 4 \).
- \( t = 5 \) is impossible, since \( \tau(16) = \tau(81) = 5 \).
- \( t = 6 \) is impossible, since \( \tau(12) = \tau(45) = 6 \).
- \( t = 7 \) is impossible, since only \( n = 64 \) satisfies \( \tau(n) = 7 \).
- \( t = 9 \) is impossible, since only \( n = 36 \) satisfies \( \tau(n) = 9 \).
- \( t = 11 \) is impossible, since there is no 2-digit \( n \) with \( \tau(n) = 11 \).

The remaining candidate values for \( t \) are 2, 3, 8, 10, and 12. These five numbers are indeed compatible with Ted’s statement #1:

- \( t = 2 \): All 2-digit integers with \( \tau(n) = 2 \) are primes, and hence odd: 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.
- \( t = 3 \): All 2-digit integers with \( \tau(n) = 3 \) are squares of primes, and hence odd: 25, 49.
- \( t = 8 \): All 2-digit integers with \( \tau(n) = 8 \) are even: 24, 30, 40, 42, 54, 56, 60, 70, 78, 88.
- \( t = 10 \): All 2-digit integers with \( \tau(n) = 10 \) are even: 48, 80.
- \( t = 12 \): All 2-digit integers with \( \tau(n) = 12 \) are even: 60, 72, 84, 90, 96.

Now let us turn to Sam’s statement #2. It is impossible that \( s = 4 \), since Sam could neither exclude \( n = 13 \) nor \( n = 31 \) (which both are compatible with statement #1). It is impossible that \( s = 5 \), since Sam could neither exclude \( n = 23 \) nor \( n = 41 \) (which both are compatible with statement #1). By analogous reasoning, we see that the digit sum \( s \) cannot take any value from the set \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16\}, since these values would not allow Sam to uniquely identify \( n \).

Hence, the only cases that agree with Sam’s statement #2 are the four cases \( s = 2, s = 3, s = 14, s = 17 \), from which Sam would respectively deduce that \( n = 11, n = 30, n = 59, n = 89 \). But the three cases \( n = 11, n = 59, \) and \( n = 89 \) all have \( t = 2 \), and thus collide with Ted’s statement #3. Therefore \( n = 30 \) with \( s = 3, t = 8 \) is the unique solution to this problem.
6 Kraus’ variant with ratio and remainder

Robert Kraus (whom we have met before in Section 4) has also created the following Freudenthal variant that is based on the division of two square numbers.

The teacher says to Xavier, Yvo, Zeno, and Rick: I have secretly chosen four integers \( x, y, z, r \), where \( x \) is a 4-digit square and \( y \) is a 3-digit square. If I divide \( x \) by \( y \), then the integer part of the resulting ratio equals \( z = \lfloor x/y \rfloor \), and the remainder in this division equals \( r \). Note that this implies \( x = yz + r \). I have told \( x \) to Xavier, \( y \) to Yvo, \( z \) to Zeno, and \( r \) to Rick.

1. One of the four guys says: I do know the numbers.
2. Another guy says: Aha. Now I know the numbers.
3. The third guy says: Aha. Now I know the numbers.
4. Finally, the last guy says: Aha. Now I know the numbers.

If we had revealed the identity of the last guy, the reader would be able to uniquely determine \( x, y, z, r \). Now find the values of \( x, y, z, r \)!

Let \( Q \) denote the set of all 1496 quadruples \((x, y, z, r)\) that agree with the announcement of the teacher. When the first guy says that he knows the numbers, the coordinate corresponding to this guy must be unique in \( Q \). And the coordinates corresponding to the quiet guys must all be non-unique in \( Q \). Based on these observations, the puzzle can easily be attacked by computer: We check all 24 orderings of the four guys. Once an ordering is fixed, we follow the conversation step by step and eliminate all quadruples with the appropriate unique respectively non-unique coordinates. The following four orderings lead to a unique solution:

- **S1**: Zeno, Xavier, Rick, Yvo leads to \((9801, 100, 98, 1)\)
- **S2**: Zeno, Rick, Xavier, Yvo leads to \((9409, 100, 94, 9)\)
- **S3**: Rick, Xavier, Yvo, Zeno leads to \((5929, 529, 11, 110)\)
- **S4**: Rick, Xavier, Zeno, Yvo leads to \((4900, 841, 5, 695)\)

Since the identity of the last guy makes the solution unique, we conclude that Rick speaks first, Xavier speaks second, Yvo speaks third, Zeno speaks last, and that \((x, y, z, r) = (5929, 529, 11, 110)\).

In the remainder of this section, we will explain how and why we end up with the four solutions S1–S4. The set \( Q \) contains 18 quadruples with unique \( z \)-coordinate, 222 quadruples with unique \( r \)-coordinate, and no quadruples with
The Bulletin of the EATCS

Table 5: The situation where Zeno makes statement #1.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>r</th>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>6724</td>
<td>100</td>
<td>67</td>
<td>24</td>
<td>q₁₀</td>
<td>8836</td>
<td>100</td>
<td>88</td>
<td>36</td>
</tr>
<tr>
<td>q₂</td>
<td>7056</td>
<td>100</td>
<td>70</td>
<td>56</td>
<td>q₁₁</td>
<td>9025</td>
<td>100</td>
<td>90</td>
<td>25</td>
</tr>
<tr>
<td>q₃</td>
<td>7225</td>
<td>100</td>
<td>72</td>
<td>25</td>
<td>q₁₂</td>
<td>9216</td>
<td>100</td>
<td>92</td>
<td>16</td>
</tr>
<tr>
<td>q₄</td>
<td>7569</td>
<td>100</td>
<td>75</td>
<td>69</td>
<td>q₁₃</td>
<td>9216</td>
<td>121</td>
<td>76</td>
<td>20</td>
</tr>
<tr>
<td>q₅</td>
<td>8281</td>
<td>100</td>
<td>82</td>
<td>81</td>
<td>q₁₄</td>
<td>9409</td>
<td>100</td>
<td>94</td>
<td>9</td>
</tr>
<tr>
<td>q₆</td>
<td>8464</td>
<td>100</td>
<td>84</td>
<td>64</td>
<td>q₁₅</td>
<td>9409</td>
<td>196</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>q₇</td>
<td>8464</td>
<td>121</td>
<td>69</td>
<td>115</td>
<td>q₁₆</td>
<td>9604</td>
<td>100</td>
<td>96</td>
<td>4</td>
</tr>
<tr>
<td>q₈</td>
<td>8649</td>
<td>100</td>
<td>86</td>
<td>49</td>
<td>q₁₇</td>
<td>9801</td>
<td>100</td>
<td>98</td>
<td>1</td>
</tr>
<tr>
<td>q₉</td>
<td>8649</td>
<td>121</td>
<td>71</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: The situation where Zeno makes statement #1.

unique \(x\)-coordinate or \(y\)-coordinate. The quadruple \((9025, 121, 74, 71)\) in \(Q\) has both, a unique \(z\)-coordinate and a unique \(r\)-coordinate. Neither Xavier nor Yvo makes statement #1, and hence we only have to distinguish two (major) cases.

In the first case Zeno makes statement #1. Table 5 lists the 17 quadruples surviving from \(Q\) (that have unique \(z\)-coordinate and non-unique \(x\)-coordinate, \(y\)-coordinate, and \(z\)-coordinate in \(Q\)). Who could make statement #2 in this case?

- If it is Xavier, then Yvo and Rick remain quiet. This leaves \(q₃\), \(q₁₁\), and \(q₁₇\) alive, which are indistinguishable to Yvo. Hence Rick makes statement #3, and Yvo makes statement #4. This yields solution S₁.
- If it is Yvo, then he sees the unique \(y\)-coordinate of quadruple \(q₁₅\). But then Xavier and Rick both identify the numbers in statement #3. A contradiction.
- If it is Rick, then Xavier and Yvo remain quiet. This leaves the seven quadruples \(q₆\), \(q₇\), \(q₈\), \(q₉\), \(q₁₂\), \(q₁₃\), and \(q₁₄\) alive. Xavier makes statement #3, Yvo makes statement #4, and we have reached solution S₂.

In the second case Rick makes statement #1, and 221 quadruples from \(Q\) survive his statement. Who makes statement #2 in this case?

- If Xavier makes statement #2 and Yvo and Zeno remain quiet, then only the seven quadruples \(q'_₁, \ldots, q'_₇\) in Table 6 survive statement #2. (a) If Yvo makes statement #3 and Zeno remains quiet, this kills all quadruples except \(q'_₄, q'_₅\). Then Zeno makes statement #4, and we reach solution S₃. (b) If Zeno makes statement #3 and Yvo remains quiet, this only leaves quadruple \(q'_₂\). We reach solution S₄.
- If Yvo makes statement #2, whereas Xavier and Zeno remain quiet, there only remain two quadruples: \((3136, 225, 13, 211)\) and \((4761, 400, 11, 361)\).
Then Xavier and Zeno both identify the numbers in statement #3. A contradiction.

- If Zeno makes statement #2, whereas Xavier and Yvo remain quiet, only the six quadruples $q''_1, \ldots, q''_6$ in Table 6 survive. Since all $x$-coordinates are unique, Xavier makes statement #3 while Yvo is quiet. This leaves $q''_2, q''_3, q''_4, q''_5, q''_6$, which are indistinguishable for Yvo. Another contradiction.

We have handled all cases, and our analysis is complete.

7 The Po Leung Kuk variant

Po Leung Kuk is a renowned welfare organization and school-sponsoring body in Hong Kong. Each year, Po Leung Kuk sponsors the Primary Mathematics World Contest (PMWC) in Hong Kong that attracts many participants (of age 13 or under) from around the world. Here is an adapted version of the twelfth problem posed at the individual round of PMWC’2001:

The teacher says to Xavier, Yvo, and Zeno: I have secretly chosen three positive integers $x, y, z$ with $x + y + z = 14$. I have told $x$ to Xavier, $y$ to Yvo, and $z$ to Zeno.

1. Xavier says: I know that Yvo and Zeno have different numbers.
2. Yvo says: I already knew that all our numbers are different.

Determine the values of $x$, $y$, and $z$!
If we replace the value 14 in the conversation by an arbitrary positive integer \( s \), then we arrive at the generalized Po Leung Kuk problem \( K(s) \). In this section, we will show that \( K(s) \) has a unique solution for all \( s \geq 6 \). The analysis depends on the remainder of \( s \) modulo 4.

Let us consider the case where the sum is of the form \( s = 4k + 2 \). Then Xavier’s statement #1 implies that \( x \) is odd: If \( x \) was even, then Xavier could not exclude the possibility that \( y = z = \frac{1}{2}(s - x) \). But if \( x \) is odd, then Xavier knows that also the sum \( y + z \) is odd; hence \( y \) and \( z \) must be different. Statement #2 by Yvo tells us that \( y \) is odd and that \( y \geq 2k + 1 \): If \( y \leq 2k - 1 \), then Yvo could never be sure that \( x \neq y \). We summarize: Since \( x \) and \( y \) are odd, we know that \( z \) must be even.

Since \( x \geq 1 \) and \( y \geq 2k + 1 \), we know that \( z \leq 2k + 1 \). Zeno is aware of all this, just before he makes his statement #3. If \( z = 2k - 2\ell \) with an integer \( \ell \geq 1 \), then Zeno could not distinguish between \( x = 1, y = 2k + 2\ell + 1, z = 2k - 2\ell \) and \( x = 1 + 2\ell, y = 2k + 1, z = 2k - 2\ell \). Hence, the only possibility is \( z = 2k \), which fixes \( x = 1 \) and \( y = 2k + 1 \) at their respective lower bounds.

This settles the case \( s = 4k + 2 \). The corresponding answers for the remaining three cases \( s = 4k, s = 4k + 1 \) and \( s = 4k + 3 \) are summarized in Table 7. The corresponding arguments run in parallel to the above argument.

<table>
<thead>
<tr>
<th></th>
<th>( Kuk(s) )</th>
<th></th>
<th>( Kuk^+(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( x )</td>
<td>( y )</td>
<td>( z )</td>
</tr>
<tr>
<td>( s = 4k )</td>
<td>1</td>
<td>2( k + 1 )</td>
<td>2( k - 2 )</td>
</tr>
<tr>
<td>( s = 4k + 1 )</td>
<td>2</td>
<td>2( k + 2 )</td>
<td>2( k - 3 )</td>
</tr>
<tr>
<td>( s = 4k + 2 )</td>
<td>1</td>
<td>2( k + 1 )</td>
<td>2( k )</td>
</tr>
<tr>
<td>( s = 4k + 3 )</td>
<td>2</td>
<td>2( k + 2 )</td>
<td>2( k - 1 )</td>
</tr>
</tbody>
</table>

Table 7: The solutions for \( Kuk(s) \) and \( Kuk^+(s) \).

Now let us turn to another variant of the Po Leung Kuk problem, where a slightly different wording of statement #2 leads to a substantially different answer. This variant is called \( Kuk^+(s) \), where the parameter \( s \geq 6 \) is a positive integer:

The teacher says to Xavier, Yvo, and Zeno: I have secretly chosen three positive integers \( x, y, z \) with \( x + y + z = s \). I have told \( x \) to Xavier, \( y \) to Yvo, and \( z \) to Zeno.

1. Xavier says: I know that Yvo and Zeno have different numbers.
2. Yvo says: [Now I know] that all our numbers are different.

Determine the values of \( x, y, \) and \( z \)!
Let us consider the case where the sum is of the form $s = 4k + 2$. Exactly as in problem $K_{uk}(4k + 2)$, we may deduce from Xavier’s statement #1 that $x$ is odd. In statement #2, Yvo informs us that he has learned something new from Xavier. Hence, it is impossible that $y$ is odd with $y \geq 2k + 1$; otherwise Yvo would have already known before that the three integers are different. It is also impossible that $y$ is odd with $y \leq 2k - 1$; otherwise Yvo could never be sure that $x \neq y$. And it is also impossible that $y = 4\ell$ holds; otherwise $x + z = 4(k - \ell) + 2$, and Yvo could not be sure that $x \neq z$. We conclude that $y$ must be of the form $y = 4\ell + 2$. Furthermore, $z$ must be odd. If $z \leq 4k - 5$ then Zeno could not distinguish between the situation where $x = 4k - z$ and $y = 2$, and the situation where $x = 4k - 4 - z$ and $y = 6$. If $z = 4k - 1$ then Zeno would have known right after Xavier’s statement that $x = 3$ and $y = 2$; this contradicts the ‘aha’ in statement #3. The only remaining possibility is $z = 4k - 3$ with $x + y = 5$. Then Zeno deduces from statement #2 that $x = 3$ and $y = 2$. The unique solution of $K_{uk}^*(4k + 2)$ is $x = 3$, $y = 2$, and $z = 4k - 3$.

This settles the case $s = 4k + 2$. The case $s = 4k + 3$ can be handled similarly; the unique solution is given in Table 7. The two cases $s = 4k$ and $s = 4k + 1$ are not uniquely solvable; both possess two possible solutions that are summarized in Table 7. We leave all details to the reader.

Finally, let us return to the original Po Leung Kuk problem. The solution to $K_{uk}(14)$ is $(x, y, z) = (1, 7, 6)$, and the slightly different wording of $K_{uk}^*(14)$ yields $(x, y, z) = (3, 2, 9)$. The difference between the two problem formulations might be even more striking for $s = 6$: The solution of $K_{uk}(6)$ is $(1, 3, 2)$, whereas the solution for $K_{uk}^*(6)$ is the permutation $(3, 2, 1)$.

References


**ON MAXIMAL PREFIX CODES**

Ludwig Staiger  
Martin-Luther-Universität Halle-Wittenberg  
Institut für Informatik  
von-Seckendorff-Platz 1  
D–06099 Halle (Saale), Germany  
staiger@informatik.uni-halle.de

Abstract

Kraft’s inequality is a classical theorem in Information Theory which establishes the existence of prefix codes for certain (admissible) length distributions. We prove the following generalisation of Kraft’s theorem: For every admissible infinite length distribution one can construct a maximal prefix codes whose codewords satisfy this length distribution.

Prefix codes are widely used in data transmission or in (algorithmic) information theory (see [3, 4]). A set of nonempty words \( C \subseteq X^* \) over an alphabet \( X \) is called a prefix code provided \( w \in C \) is not a prefix of \( v \in C \), for every pair of distinct words \( w, v \in C \).

A classical theorem about the existence prefix codes is called Kraft’s inequality [2].

**Theorem 1 (Kraft’s inequality).** Let \( X \) be a finite alphabet, \( I \subseteq \mathbb{N} \) and let \( f : I \to \mathbb{N} \) be a non-decreasing function such that \( \sum_{n \in I} |X|^{-f(n)} \leq 1 \). Then there is a prefix code \( C = \{v_n : n \in I\} \subseteq X^* \) such that \( |v_n| = f(n) \).

Here \( |X| \) denotes the cardinality of the set \( X \), and \( |v| \) denotes the length of the word \( v \) and \( \sum_{n \in I} |X|^{-f(n)} \leq 1 \) means that the length distribution \( f : I \to \mathbb{N} \) is admissible.

The aim of this note is to show that a simple modification of Kraft’s construction (see e.g. [4]) is suitable for the construction of infinite maximal prefix codes \( C \subseteq X^* \) whenever \( \sum_{v \in C} |X|^{-|v|} \leq 1 \).

Here a code \( C \subseteq X^* \) is referred to as maximal prefix if \( C \) is a prefix code and for every prefix code \( C' \supseteq C \) implies \( C' = C \). It is known that a maximal prefix code need not be maximal as a code (see e.g. [1, II. Example 3.1]). For finite codes \( C \subseteq X^* \), however, a maximal prefix code satisfies \( \sum_{v \in C} |X|^{-|v|} = 1 \) and is also maximal as a code.
Theorem 2. Let \( f : \mathbb{N} \to \mathbb{N} \) be a non-decreasing function such that \( \sum_{n \in \mathbb{N}} |X|^{-f(n)} \leq 1 \). Then there is a maximal prefix code \( C = \{v_n : n \in \mathbb{N}\} \subseteq X^* \) such that \( |v_n| = f(n) \).

We use the following characterisation of maximal prefix codes whose proof is given here for the sake of completeness.

Lemma 3. Let \( M \) be an infinite subset of \( \mathbb{N} \). A code \( C \subseteq X^* \) is maximal prefix if and only if for all \( w \in \{v : v \in X^* \wedge |v| \in M\} \) there is a \( v \in C \) such that \( w \subseteq v \) or \( v \subseteq w \).

Proof. If \( C \) is not maximal prefix then there is a \( w \notin C \) such that \( C \cup \{w\} \) is a prefix code. Consider \( wu \in X^* \) where \( wu \in M \). Since \( w \notin v \) and \( v \notin w \) for every \( v \in C \), the same holds true for the word \( wu \).

Conversely, if for some \( w \in \{v : v \in X^* \wedge |v| \in M\} \) there is no \( v \in C \) such that \( w \subseteq v \) or \( v \subseteq w \) then \( C \cup \{w\} \) is a prefix code properly containing \( C \). \( \square \)

Now, using this lemma we construct a prefix code which satisfies the condition of Lemma 3 for some infinite set \( M \subseteq \{f(n) : n \in \mathbb{N}\} \). This is done by the following algorithm MaxKraft.

Algorithm MaxKraft
\[
\begin{align*}
0 & \quad n := 0; \quad l := 0; \quad C := \emptyset; \quad M := \emptyset \\
1 & \quad \text{For } i = 1 \text{ to } \infty \text{ do} \\
2 & \quad l := f(n); \quad W := X^i \setminus C \cdot X^*; \quad M := M \cup \{l\} \\
3 & \quad \text{Let } W = \{w_1, \ldots, w_{|W|}\} \\
4 & \quad \text{For } j = 0 \text{ to } |W| - 1 \text{ do} \\
5 & \quad \quad C := C \cup \{w_{j+1} \cdot 0^{(n+j)-l}\} \\
6 & \quad \text{Endfor} \\
7 & \quad n := n + |W| \\
8 & \quad \text{Endfor}
\end{align*}
\]

Here the set \( M \) is included just to have a reference to Lemma 3.

At stage \( i + 1 \) our parameters before constructing the new approximation \( C_{i+1} \) are \( C_i \), \( n_i \) and \( l_{i+1} = f(n_i) \) where \( f(n_i - 1) = \sup\{|w| : w \in C_i\} \).

Then the set \( W_{i+1} = X^{l_{i+1}} \setminus C_i \cdot X^* \) is the set of words which have no prefix in \( C_i \). For each of the words \( \{w_1, \ldots, w_{|W_{i+1}|}\} \), the body of the For-loop (lines 4 to 6) adds the word \( w_{j+1} \cdot 0^{(n_i+j)-l_{i+1}} \) of length \( f(n_i + j) \) to \( C_i \). Thus \( f(j) \) is the length of the \( j \)th word in \( C_{i+1} \) if \( j \leq |C_{i+1}| \), in particular \( f(n_{i+1} - 1) = \sup\{|w| : w \in C_{i+1}\} \).

As in the proof of Kraft’s inequality, we obtain that
\[
|W_{i+1}| = \sum_{v \in C_i} |X|^{-f(n_i)} = |X|^{f(n_i)} \cdot \sum_{j=1}^{|C_i|} |X|^{-f(j)} < |X|^{f(n_i)}.
\]
The Bulletin of the EATCS

Consequently, the algorithm does not stop, that is, $C_i \subset C_{i+1}$, and returns an infinite set $C = \bigcup_{i=1}^\infty C_i$ in which the word constructed in step $j$ has length $f(j)$.

Clearly, the resulting $C_{i+1}$ is a prefix-code, if $C_i$ is a prefix-code, and by the steps in lines 4 and 5 every word of length $l_{i+1}$ has a prefix in $C_i \subseteq C_{i+1}$ or is a prefix of some word in $C_{i+1}$.

At the next stage this process is repeated for the new (greater) length $l_{i+2} := f(n_i + |W_{i+1}|)$. So, by induction, it is seen that $C = \bigcup_{i=1}^\infty C_i$ is a prefix code for which the infinite set $M = \{l_i : i = 1, \ldots\}$ is a witness for its prefix maximality.

The algorithm depends on the monotonicity of the function $f : \mathbb{N} \to \mathbb{N}$. The monotonicity guarantees that, when, at some stage $i$, the finite approximation $C_i$ of the code $C$ is constructed, all words $w \in C \setminus C_i$ will have length $|w| \geq f(n_i - 1)$.

References

RELABELING AND THE INDEPENDENCE THEOREM IN
THE DOUBLE-PUSHOUT APPROACH TO
GRAPH TRANSFORMATIONS

Hans J. Schneider
Universität Erlangen-Nürnberg – Institut für Informatik
(Lehrstuhl 2)
Martensstraße 3, 91058 Erlangen (Germany)
schneider@informatik.uni-erlangen.de

Abstract
Studying parallelism and concurrency in the double-pushout approach to
graph transformations is mainly based on the so-called triple-pushout con-
dition. The categories of sets and of graphs satisfy this condition. If we,
however, consider graph morphisms that may change the labels in a well-
defined manner (structurally labeled graphs), the triple-pushout condition
does no longer hold true. In this paper, we propose a restriction on the struc-
ture of the alphabet such that on the one hand, the triple-pushout condition
can be proved, and on the other hand, relevant applications are covered, e.g.,
term graph rewriting.

1 Introduction

Graph structures are ubiquitous in computer science as well as in many application
areas. They are a very natural way to explain complex situations on an intuitive
level, and they are used to define the syntax of structures according to given rules
(graph grammars) as well as to describe dynamically deriving new situations from
given ones (graph transformations). In 1973, we have introduced the double-
pushout approach to formalize this process [9]. Using concepts from category
theory leads to elegant proofs, which can easily be applied to different categories
[1, 3, 17].

Derivation sequences consist of derivation steps that are applied one after the
other. If two productions are applicable to a given graph \( G \), the result may depend
on the order of application, or even the second production is no longer applicable.
after the first has been applied. Of course, the result does not depend on the order of application if the left-hand sides of both productions are mapped into disjoint parts of $G$. But this condition is too strong. The images of the left-hand sides may overlap. In this case, however, the first derivation step must transfer the elements that are needed to apply the second production from the left-hand side to the right-hand side by the interface graph. The notion of parallel independence and the parallel independence theorem characterize the interchangeability categorically. The survey paper by H. Ehrig [2] as well as the analysis of parallelism by H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce [4] and the fundamental summary by A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe [1] restrict discussion of parallel independence to graph productions the left-hand sides as well as the right-hand sides of which are injective graph morphisms. Even the most recent approach by H. Ehrig, A. Habel, J. Padberg, and U. Prange based on adhesive categories [5] assumes this restriction. Nevertheless, there are relevant applications that need productions with noninjective right-hand sides.\textsuperscript{1} An example is removing common subexpressions from a term graph. D. Plump has discussed this application in detail [13]; but that presentation is not based on the double-pushout approach. In addition, this application needs changing some node labels when replacing the left-hand side by the right-hand side of a production. A. Habel and D. Plump propose a solution to this problem using partially labeled interface graphs [11]: If the node of the interface graph is not labeled, the corresponding nodes on the left-hand and on the right-hand side may bear different labels. A special constraint on the productions that can not be formulated categorically ensures that derived graphs are totally labeled. A more general approach to relabeling (but restricted to injective morphisms) has been presented by F. Parisi-Presicce, H. Ehrig, and U. Montanari [12]. These authors also use a set-theoretic formulation to make the labels on the derived graph unambiguous. In [16], we have shown that all these cases can be also treated categorically, giving us the chance to apply the general concepts to relabeling, too.

A typical example of these concepts of interest is the theorem on parallel and sequential independence. The aim of the present paper is to formulate it in such a way that it can be applied to term graph rewriting and other areas that need relabeling. In Section 2, we review the basic notions without explicitly mentioning hypergraphs, although terms are often modeled by hypergraphs. (It is straightforward to apply our arguments to this case.) Considering the original proof of the parallel independence theorem by H. Ehrig and H.-J. Kreowski [8] allows us to characterize the weakest conditions the proof needs (PIT-categories). Especially,
the restriction to injective right-hand sides is not necessary. Unfortunately, removing this restriction makes the proof that \( \text{Set} \) satisfies the triple-pushout condition a little bit tortuous, since the standard proof does no longer work. In the main section, we consider the case of structurally labeled graphs introduced by F. Parisi-Presicce, H. Ehrig, and U. Montanari [12]. It is well-known that in general, the category \( \text{SLgraph} \) does not satisfy the triple-pushout condition [6], but we can restrict the structure of the alphabet such that on one hand, the condition becomes valid, and on the other hand, some relevant applications are covered.

2 Fundamentals and Previous Results

We briefly recapitulate the notion of derivability as used in the categorical approach.

Definition 2.1 (Production). A production is a pair of morphisms \((p_l', p_r')\) with common domain: \( p = (B_l' \leftarrow I \rightarrow B_r') \).

We call \( B_l' \) the left-hand object of the production and \( B_r' \) its right-hand object, whereas the terms left-hand side and right-hand side refer to \( p_l' \) and \( p_r' \), respectively. In [1] as well as in [3], both sides are assumed to be injective. In [17], we consider the general case. The interface object \( I \) is used to define the transformation:

Definition 2.2 (Derivability). An object \( G_l' \) is said to be derivable from an object \( G_l \) by the production \( p = (B_l' \leftarrow I \rightarrow B_r') \), written as \( G_l' \overset{p}{\longrightarrow} G_r' \), if there is a context object \( C \) together with a morphism \( g : I \rightarrow C \), called the embedding, such that \( G_l \) and \( G_r \) are the pushout objects in the following diagram:

\[
\begin{array}{ccc}
B_l' & \overset{p_l'}{\longrightarrow} & I & \overset{p_r'}{\longrightarrow} & B_r' \\
\downarrow g & & \downarrow g & & \downarrow g \\
G_l' & \underset{p_l}{\longleftarrow} & C & \underset{p_r}{\longrightarrow} & G_r' \\
\end{array}
\]

\( g_l' \) is called the match (or handle), and \( g_r' \) is the co-match (or co-handle).

\( G_l' \) and \( G_r' \) are unambiguously defined if we know the production \( p = (p_l', p_r') \) and the embedding \( g : I \rightarrow C \) of the interface object \( I \) into the context object \( C \). It is well-known from category theory that an injective embedding results

\[2\text{A. Habel, J. Müller, and D. Plump have studied the four cases resulting from requiring the match and/or the right-hand side to be injective or not, in great detail [10].} \]
in injective match and injective co-match. In this paper, we require neither the embedding nor the match to be injective.

As usual, a graph is a quadruple \( G = (E, V, s, t) \) with \( E \) and \( V \) being the sets of edges and nodes, respectively, and two set morphisms \( s \) and \( t \) assigning a source node and a target node to each edge. A graph morphism \( f : G \to H \) is a pair \((f_E : E_G \to E_H, f_V : V_G \to V_H)\) such that \( f_V \cdot s_G = s_H \cdot f_E \) and \( f_V \cdot t_G = t_H \cdot f_E \). The category \( \mathcal{L} \text{graph} \) of labeled graphs with label preserving morphisms does not add any new difficulties to constructing pushouts. In many areas of computer science as well as in application areas, however, it is necessary to change labels. With data base applications in mind, F. Parisi-Presicce, H. Ehrig, and U. Montanari [12] define a structure on the labeling alphabet that allows the user to specify which changes are allowed and which are not:

**Definition 2.3 (SLgraph).** Let \( L_E \) and \( L_V \) be two alphabets on which reflexive and transitive relations \( \sqsubseteq_{L_E} \) and \( \sqsubseteq_{L_V} \) are defined.\(^3\) A structurally labeled graph (SL-graph) is a labeled graph \( G = (E, V, s, t, l_E, l_V) \) with \( l_E : E \to L_E \) and \( l_V : V \to L_V \). An SL-graph morphism \( f : G \to H \) is a graph morphism \( f = (f_E : E_G \to E_H, f_V : V_G \to V_H) \) which additionally satisfies:\(^4\)

\[
\forall v \in V_G (l_{VG}(v) \sqsubseteq_{L_V} l_{VH}(f_V(v))) \land \forall e \in E_G (l_{EG}(e) \sqsubseteq_{L_E} l_{EH}(f_E(e))).
\]

The set of SL-graphs and the set of SL-graph morphisms together with componentwise composition constitute the category \( \mathcal{SL} \text{graph} \).

Constructing the pushout in \( \mathcal{SL} \text{graph} \), some elements are put together. We have to label the resulting element with the least upper bound of the labels of the original elements.\(^5\)

**Lemma 2.4 ([12]).** If in the structured alphabet, the least upper bound exists, then \( \mathcal{SL} \text{graph} \) has pushouts.

Parisi-Presicce et al. restrict discussion to injective left-hand sides and additionally assume the right-hand sides to be injective. Considering the underlying graphs, injective left-hand sides make the derivation steps unambiguous if it exists. It is easy to see that structured labeling adds ambiguity to constructing pushout complements even in the case of an injective \( p^l \), since different labels may lead to the same least upper bound. Parisi-Presicce et al. make the definition unambiguous by an explicit condition written in a set-theoretic style. It requires existence of a minimal label that is chosen to be the solution. In [16], we have treated this aspect categorically.

\(^3\)Contrary to a partial order, a structured alphabet need not be antisymmetric.

\(^4\)In the following, we omit the indices of the relations, since they can not be confused.

\(^5\)Parisi-Presicce et al. use the inverse relation \( \sqsupseteq \). Therefore, they need the greatest lower bound.
The work reported here has been motivated by term graph rewriting [13]. In this case, we can take advantage of the special structure of the alphabet:

**Definition 2.5 (Term graph alphabet).** A term graph alphabet consists of the following:

1. A set of (possibly typed) variables, constants, and operator symbols that are given by the application (basic labels),
2. A set of (possibly typed) metalabels that may occur in the productions with \( l \subseteq v, l \subseteq c, l \subseteq op \) if and only if the type of the metalabel \( l \) and the type of the variable \( v \), the constant \( c \), or the operator symbol \( op \), respectively, agree,
3. A special symbol \( \bot \) that may occur in the interface graph and that satisfies \( \bot \subseteq l \) for each metalabel \( l \),
4. A special symbol \( \top \) with \( v \subseteq \top, c \subseteq \top, op \subseteq \top \) for all variables \( v \), constants \( c \), and operator symbols \( op \), respectively.

If we do not have a typed alphabet, the second condition simply means that the arity of the operator is the same as that of the metalabel.

This structure of a term graph alphabet is illustrated in Fig. 1. In the graphs to be derived, we find only the labels of the second level, where \( v \) and \( c \) represent the variables and constants occurring in the application of interest. These two nodes are placeholders for a set of nodes with different identifiers. On the first level, we have metavariables that may occur in the productions. In order to simplify the figure, we have mentioned only one group of function symbols, namely the dyadic operators on the natural numbers. We have arcs from the metalabel \( op_{\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}} \) to all the operators of this type. The label \( \bot \) is allowed only in the interface graph. Finally, the \( \top \)-symbol is necessary to ensure existence of the least upper bound. If it occurs in a derived graph, the production can be applied formally, but from the application point of view, the result is not valid.

**Figure 1.** Structure of a term graph alphabet

This structure of a term graph alphabet is illustrated in Fig. 1. In the graphs to be derived, we find only the labels of the second level, where \( v \) and \( c \) represent the variables and constants occurring in the application of interest. These two nodes are placeholders for a set of nodes with different identifiers. On the first level, we have metavariables that may occur in the productions. In order to simplify the figure, we have mentioned only one group of function symbols, namely the dyadic operators on the natural numbers. We have arcs from the metalabel \( op_{\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}} \) to all the operators of this type. The label \( \bot \) is allowed only in the interface graph. Finally, the \( \top \)-symbol is necessary to ensure existence of the least upper bound. If it occurs in a derived graph, the production can be applied formally, but from the application point of view, the result is not valid.
Example 2.6. Let us consider a graph production removing common subexpressions from a term graph (Figure 2). The nodes of a term graph are labeled with operator symbols, constants, and variables. Constants and variables occur only at the leaves.

In the figures, the nodes are represented by their labels. The different nodes are distinguished by numbers that we write as exponents, and the mappings are often given by these numbers, e.g., \( p'_{1}(3) = 3 \). If a mapping is not injective, we use square brackets: \( p'_{1}(1) = p'_{1}(2) = [1, 2] \). Edge labels \( l, r \) indicate the left-hand and the right-hand operands, respectively.

Two derivations are given in Figure 3. This figure shows only the interesting parts of the term graphs explicitly, the rest is indicated by dots. In the first derivation, we apply the production \( p \) to the graph \( G_{0} \) using the match \( g'_{1} \) defined by:

\[
\begin{align*}
g'_{1}(1) &= 3 \\
g'_{1}(2) &= 4 \\
g'_{1}(3) &= g'_{1}(4) = [5, 6].
\end{align*}
\]

Afterwards, we apply the same production to \( G_{1} \) using

\[
\begin{align*}
g'_{2}(1) &= 1 \\
g'_{2}(2) &= 2 \\
g'_{2}(3) &= g'_{2}(4) = [3, 4].
\end{align*}
\]

In the second derivation, we start with matching the left-hand side of \( p \) into the upper part of the given graph \( G_{0} \): \( g''_{1}(1) = 3 \\
g''_{1}(2) = g''_{1}(3) = 3 \\
g''_{1}(4) = 4 \) and the second step is given by \( g''_{2}(1) = 3 \\
g''_{2}(2) = 4 \\
g''_{2}(3) = g''_{2}(4) = [5, 6]. \)

The order of applying the productions does not affect the result if the elements that are needed to apply the second production are either not used by the first production or are transferred from the left-hand side to the right-hand side via the interface graph:

**Definition 2.7 (Parallel independence [8]).** Two derivation steps \( G_{0} \xrightarrow{p_{1}} G_{1} \) and \( G_{0} \xrightarrow{p_{2}} G'_{1} \) are called parallel independent (or one derivation step is parallel independent of the other) if and only if

\[
(\exists \alpha_{1} : B'_{1} \rightarrow C_{2})(\exists \alpha_{2} : B'_{2} \rightarrow C_{1})(g'_{1} = \bar{p}_{1} \cdot \alpha_{1} \land g'_{2} = \bar{p}_{1} \cdot \alpha_{2})
\]

\[6\]In this case, the node mapping defines an unambiguous mapping of the edges.
with the notations as in the definition of derivability and the indices referring to the first and to the second derivation step, respectively.

The reason to factorize the handle \( g'_2 = \bar{p}_1 \cdot \alpha_2 \) is to make \( p_2 \) applicable to \( G_1 \) using the handle \( \bar{p}_1 \cdot \alpha_2 \). The Independence Theorem below shows that under suitable assumptions, the result does not depend on the order of applying the productions.

The definition of parallel independence adequately models the problems arising in studying concurrent processes. Alternatively, we may be interested in exchanging the steps of a given sequence in order to get a canonical form:

**Definition 2.8 (Sequential independence [8])**. Two derivation steps \( G_0 \stackrel{p_1}{\Rightarrow} G_1 \) and \( G_1 \stackrel{p_2}{\Rightarrow} G_2 \) are called sequentially independent (or the second step is sequentially independent of the first) if and only if

\[
(\exists \alpha_1 : B'_1 \rightarrow C_1)(\exists \alpha_2 : B'_2 \rightarrow C_1) (g'_1 = \bar{p}_1 \cdot \alpha_1 \land g'_2 = \bar{p}_2 \cdot \alpha_2)
\]

with the notations as in the definition of derivability and the indices referring to the first and to the second derivation step, respectively.

Both derivations of Figure 3 also satisfy this condition.

A closer look to the original proof by H. Ehrig and H.-J. Kreowski [8] reveals the properties a category must provide such that the Independence Theorem can be proved. In a previous paper [14], we have called such a category a PIT-category indicating by the prefix that the (parallel) independence theorem holds. Here, we weaken that assumption to allow noninjective right-hand sides, but we do not change the name:

**Definition 2.9 (PIT-category [14])**. We call a category \( C \) a PIT-category with respect to a class \( \mathcal{M} \) of distinguished morphisms if the following conditions are satisfied:

- (a) \( C \) has pushouts, and if Figure 4(a) is a pushout diagram with morphism \( A \rightarrow B \) being in \( \mathcal{M} \), then \( C \rightarrow D \) is in \( \mathcal{M} \), too.
- (b) If morphism \( B \rightarrow D \) is in \( \mathcal{M} \) and if \( C \rightarrow D \) is an arbitrary morphism of \( C \), there exists the pullback diagram 4(a) and the morphism \( A \rightarrow C \) is in \( \mathcal{M} \), too.
- (c) Triple-Pushout-Condition: If in the diagram of Figure 4(b), the composed diagrams (1 + 3) and (2 + 3) are pushout diagrams, and subdiagram (3) is a pullback diagram with \( C_2 \rightarrow G \) in \( \mathcal{M} \), then the subdiagrams (1) and (2) are pushout diagrams.

In condition (c), we only assume \( C_2 \rightarrow G \) to be in \( \mathcal{M} \), but \( C_0 \rightarrow C_1 \) is in \( \mathcal{M} \), too, because of condition (b). The definition of adhesive categories, as given
The Bulletin of the EATCS

by H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer in [3], assumes all the other morphisms to be in \( \mathcal{M} \), too.

An analogous concept is the HLR’-category discussed by H. Ehrig, A. Habbel, and F. Parisi-Presicce [7], but that concept also includes composing parallel productions, which we do not need here. A more important difference is that condition (c) is replaced by an assumption on the pushout-pullback-decomposition from that the triple-pushout-condition follows, but not vice versa.

Theorem 2.10 (Independence theorem [8]). If in a PIT-category, we have two productions \( p_1 \) and \( p_2 \) the left-hand sides of which are in the distinguished class \( \mathcal{M} \), then the following assertions hold:

1. If two derivation steps \( G_0 \xrightarrow{p_1} G_1 \) and \( G_0 \xrightarrow{p_2} G'_1 \) are parallel independent, then there exists a \( G_2 \) such that we get \( G_1 \xrightarrow{p_1} G_2 \) and \( G_1 \xrightarrow{p_2} G'_2 \). The constructed derivation sequences \( G_0 \xrightarrow{p_1} G_1 \xrightarrow{p_2} G_2 \) and \( G_0 \xrightarrow{p_2} G'_1 \xrightarrow{p_1} G'_2 \) are sequentially independent.

2. If two derivation steps \( G_0 \xrightarrow{p_1} G_1 \xrightarrow{p_2} G_2 \) are sequentially independent, then there exists a \( G'_1 \) such that \( G_0 \xrightarrow{p_2} G'_1 \) and \( G'_1 \xrightarrow{p_1} G_2 \). Furthermore, \( G_0 \xrightarrow{p_1} G_1 \) and \( G_0 \xrightarrow{p_2} G'_1 \) are parallel independent, and \( G_0 \xrightarrow{p_2} G'_1 \xrightarrow{p_1} G_2 \) is sequentially independent.

This theorem has been proved by H. Ehrig and H.-J. Kreowski [8]. They do not even assume that the left-hand sides of the productions are injective. Instead, they restrict discussion to “natural” derivation steps, i.e., \( \bar{p}_1 \) and \( \bar{p}_2 \) must be injective.\(^7\) This assumption is identical to our assumption that \( C_2 \rightarrow G \) must be in \( \mathcal{M} \).

The definitions of sequential independence and of parallel independence do not require any morphisms to be in the distinguished class \( \mathcal{M} \). We need this condition only to prove the independence theorem.

3 Independence Theorem in the Category of Structurally Labeled Graphs

In all categories of interest, existence of pushouts and pullbacks is not a restriction, at all. Since the construction of pushouts and pullbacks in \( \text{Graph} \) is uniquely determined by separately constructing them for nodes and edges in \( \text{Set} \), it is sufficient to consider the category of sets.

In \( \text{Set} \), the triple-pushout condition (Definition 2.9(c)) is usually based on the lemma on composing and decomposing pushouts and on the fact that a pullback

\(^7\)This is a special case of what we have called maximal derivation steps in [16].
diagram $\hat{f} \cdot g = \hat{g} \cdot f$ with injections $\hat{f}$ and $\hat{g}$ and jointly epimorphic $(\hat{f}, \hat{g})$ is a pushout diagram, too.\textsuperscript{8}

In each category with pushouts, the morphisms constructed in a pushout diagram are jointly epimorphic. Therefore, in the triple-pushout condition, the pair $(\hat{p}_1, \hat{p}_2)$ is jointly epimorphic, too. Proving the second part of the independence theorem in the case of noninjective right-hand sides, however, we have only one injective morphism. For that reason, the standard proof (see, e.g., [1, 3]) is not applicable.

Nevertheless, we can prove that $\mathcal{S}et$ satisfies the triple-pushout condition even under the weakened assumptions. We can show that in Fig. 4, subdiagrams (1) and (2) are pushout diagrams by taking advantage of the overlapping structure of the large diagrams, and then the pushout property of (3) follows by the lemma on decomposing pushouts. (You can find the details of this proof in the appendix of the version on the author’s homepage.) Therefore, we have:

**Fact 3.1.** The categories $\mathcal{S}et$ and $\mathcal{G}raph$ are PIT-categories with respect to the class of injections.

We now turn to the category $\mathcal{S}L\mathcal{G}raph$ of structurally labeled graphs. The pullback construction in $\mathcal{S}L\mathcal{G}raph$ is dual to the pushout construction:

**Lemma 3.2 (Pullback in $\mathcal{S}L\mathcal{G}raph$).** If in the structured alphabet, the greatest lower bound exists, then $\mathcal{S}L\mathcal{G}raph$ has pullbacks.

Unfortunately, the category $\mathcal{S}L\mathcal{G}raph$ of structurally labeled graphs is not a PIT-category, i.e., the independence theorem does not hold even in the case that all morphisms are injective.

**Example 3.3.** We refer to a counter-example given by H. Ehrig, A. Habel, and F. Parisi-Presicce [6]. We assume all morphisms in the diagram of Figure 4(b) to be injective. We consider two elements $i_1 \in I_1$ and $i_2 \in I_2$ mapped onto the same $g \in G$ by the morphisms $\hat{p}_2 \cdot \alpha_1 \cdot p_1$ and $\hat{p}_1 \cdot \alpha_2 \cdot p_2$, respectively. The labels of these elements satisfy the relations as given in Figure 5.\textsuperscript{9} If the structured alphabet contains a label $x$ related to the other labels as indicated in the subdiagram (1), we have a problem: $l(g)$ is the least upper bound of $l(h_1)$ and $l(c_1)$ as well as of $l(h_2)$ and $l(c_2)$ by the pushout property of the subdiagrams (1 + 3) and (2 + 3), respectively, and $l(c_0)$ is the greatest lower bound of $l(c_1)$ and $l(c_2)$ by the pullback property of (3). $l(c_2)$, however, is not the least upper bound of $l(c_0)$ and $l(h_1)$, but $x$ is. Therefore, subdiagram (1) is not the pushout diagram of $g_1'$ and $p_1$, although the diagram of the underlying unlabeled graphs is.

\textsuperscript{8} A pair $(f, g)$ of morphisms is called jointly epimorphic if and only if for all $h_1, h_2$ with $h_1 \cdot f = h_2 \cdot f \land h_1 \cdot g = h_2 \cdot g$, the equality $h_1 = h_2$ follows. In $\mathcal{S}et$, this means that $D = f[C] \cup g[B]$.

\textsuperscript{9} For reason of simplicity, we have omitted the indices of the $l$-functions.
The Bulletin of the EATCS

This example also shows that it is not promising to impose an additional restriction to the set of distinguished morphisms \( \mathcal{M} \), since the problem arises from the morphisms \( g'_2 \) and \( \alpha_1 \) that are not assumed to be in \( \mathcal{M} \). Instead, we must impose a constraint to the structured alphabet that excludes existence of such an element \( x \). If we slightly vary the lower part of Figure 5, we get the diagram of Figure 6 suggesting the following theorem:

**Theorem 3.4.** The category SL\( \text{graph} \) is a PIT-category with respect to \( \mathcal{M} \) being the class of all morphisms the underlying graph morphisms of which are injective,

(a) if in the alphabet, the least upper bound and the greatest lower bound exist for each pair of labels, and

(b) if the structure of the alphabet satisfies

\[
\text{glb}(l_1, l_2) = \text{glb}(l_1, l'_{2}) \land l_2 \subseteq l'_{2} \implies l_2 = l'_{2}
\]

The proof is based on the fact that \( \text{Graph} \) is a PIT-category with respect to injective graph morphisms. Condition (a) ensures that pushout and pullback constructions exist in the category SL\( \text{graph} \) under consideration. We have only to show that in the triple-pushout condition (Figure 4b), the subdiagrams (1) and (2) satisfy the pushout condition with respect to the labeling. Let us consider subdiagram (3). The underlying diagram in \( \text{Graph} \) is a pushout diagram. If it were not the pushout diagram in SL\( \text{graph} \), there must be at least one element \( c_2 \) of \( C_2 \) the label of which is not

\[
x = \text{lub}\{l_{C_{1}}([c_{0} \mid g''_{2}(c_{0}) = c_{2}]) \cup l_{b_{1}}([b_{1} \mid \alpha_{1}(b_{1}) = c_{2}])\}
\]

since this is the label of \( C_{2} \) in the case that (1) is the pushout. Trivially, we have \( l_{C_{2}}(c_{0}) \subseteq x \) for all \( c_{0} \) with \( g''_{2}(c_{0}) = c_{2} \) and \( l_{b_{1}}(b_{1}) \subseteq x \) for all \( b_{1} \) with \( \alpha_{1}(b_{1}) = c_{2} \). On the other hand, we have \( x \subseteq l_{C_{2}}(g''_{2}(c_{0})) \) and \( x \subseteq l_{C_{2}}(\alpha_{1}(b_{1})) \) by the universal property of the assumed pushout. Furthermore, \( C_{0} \) is the pullback object in subdiagram (3), and therefore,

\[
l_{C_{2}}(c_{0}) = \text{glb}(l_{C_{1}}(g''_{1}(c_{0})), l_{C_{2}}(g''_{2}(c_{0}))).
\]

Now, we make use of the fact that the greatest lower bound \( \text{glb}(a, b) \) does not change if we replace \( b \) by an element \( c \) with \( \text{glb}(a, b) \subseteq c \subseteq b \), i.e., \( \text{glb}(a, b) = \text{glb}(a, c) \), and we get

\[
l_{C_{2}}(c_{0}) = \text{glb}(l_{C_{1}}(g''_{1}(c_{0})), \underbrace{x}_{}).
\]

Condition 3.4(b) yields \( x = l_{C_{2}}(g''_{2}(c_{0})) = l_{C_{2}}(c_{2}) \). Since we have not used injectivity of the underlying graph morphism \( \bar{p}_{2} \) in this part of the proof, it can be analogously applied to subdiagram (2).

It is easy to see that a term graph alphabet satisfies the criteria of Theorem 3.4:

**Theorem 3.5.** The category SL\( \text{graph} \) using a term graph alphabet is a PIT-category with respect to \( \mathcal{M} \) being the class of all morphisms the underlying graph morphisms of which are injective.
In the alphabet, the symbol $\bot$ allows relabeling as we have seen in our example. Furthermore, it ensures existence of pullbacks. On the other hand, the symbol $\top$ ensures existence of pushouts. The formal construction of a derivation step, however, can lead to a $\top$-label on a node in the derived graph contrary to our idea of what a term graph is. Therefore, we must exclude such derivation steps:

**Definition 3.6 (Term graph grammar).** A term graph grammar is a graph grammar labeled with a term graph alphabet:

(a) The start graph is labeled with basic labels.
(b) The graphs in the productions are allowed to bear basic labels and metalabels. In addition, $\bot$ is allowed on the interface graphs.
(c) A derivation step is valid if and only if the derived graph is labeled only with basic labels.

Conditions (a) and (c) ensure that we can derive only graphs that are correctly labeled.

**Corollary 3.7.** In a term graph grammar, the assertions of the independence theorem hold true for valid derivation sequences.

It may be the reader’s impression that we have designed the criterion to especially solve the term graph application. Therefore, we also consider the example used by F. Parisi-Presicce, H. Ehrig, and U. Montanari [12] introducing the structured labeling:

**Example 3.8.** We start with the graph of Figure 7 depicting a family’s mother-of relation. We want to insert sister-of edges by applying a suitable production that is shown in Figure 8. Its left-hand object must ensure not only that both persons have the same mother, but also that the person at the source of the new edge is female. We ensure this by defining a transitive relation on the alphabet generated by:

$$\begin{align*}
  f &\subseteq \text{mary, joan, jane, dora} \\
  m &\subseteq \text{david} \\
  \bot &\subseteq p \subseteq f, m
\end{align*}$$

The structure of this alphabet is illustrated by Figure 9(a). Indeed, this structure does not satisfy the condition (b) of Theorem 3.4:

$$\text{glb(jane, david)} = \text{glb(female, david)} \land \text{female} \subseteq \text{jane}$$

We can, however, change the structure such that the condition is satisfied: We lift the metalabel $p$ to the level of $m$ and $f$. Of course, the relevant information is not changed if we directly connect $p$ to the labels $l$ with $f \subseteq l$ or $m \subseteq l$, respectively, as shown in Figure 9(b).

---

10 For reason of space, we have abbreviated the names to the first two letters.
4 Conclusion

The double-pushout approach leads to clearly structured results on graph transformations. Since it is based on category theory, the results are generic and can be applied to a wide variety of graph models. The independence theorem, which is the basis of a lot of results on parallelism and concurrency, has been proved even without reference to graphs. We could show that it also holds true in the case integrating structural transformations and operations on labels if we restrict the operations on the labels in a suitable way. This constraint does not exclude some relevant applications. Another application of interest is parallel logic programming. It is straightforward to show that Prolog rules can be modeled as graph transformation rules [15, Sect. 7.7] and satisfy our criterion.

Nevertheless, we have to analyze further applications. An example not covered by our criterion is an alphabet of sets with the subset relation as the structure. You can describe the dining philosophers in this way. (See, e.g., [15, Sect. 7.5].) Although this structure does not satisfy Theorem 3.4, we observe that the subsets of labels that can occur on a specific node do. This example, however, can not be easily generalized. Applications of this type deserve further investigations.

The author would like to thank H. Ehrig, A. Habel, and U. Prange for their comments and suggestions on previous versions of this paper.

References


[17] H.J. Schneider: Graph Transformations – An Introduction to the Categorical Approach, Preliminary version: www2.informatik.uni-erlangen.de/~schneide/gtbook/index.xml
Figure 3. Parallel independent derivation steps
Figure 4.

Figure 5.

Figure 6.

Figure 7.
Figure 8.

Figure 9. Structure of the alphabet used in Example 3.8
THE PUZZLE CORNER

BY

LAURENT ROSAZ

LRI, Orsay CNRS-Université de Paris Sud
Bât 490, 91405 Orsay France
Laurent.Rosaz@lri.fr

Readers are invited to send comments, and to send exercises, even if they don’t know the answer. Write to Laurent.Rosaz@lri.fr.

78 The ants

At both ends of a 1-meter line, I have made a hole. Some ants are walking on that line, some of them from left to right, some others from right to left. They walk at 1 meter per minute. When they reach an end, they fall in the hole. When two ants bump into each other, they instantly turn back and keep on in their new direction. Show that all ants will have fallen in 1 minute.

Thanks to Sebastien Tixeuil for that problem.

79 Other proofs of the undecidability of the halting problem

You know of course that the halting problem is undecidable, which is usually proved by the “diagonal” argument. Can you find two others proofs?

Hint for proof number one: introduce $K(n)$ which is the size of the smallest program which doesn’t read anything and produces $n$ as output (Kolmogorov complexity). Show that $K$ is not computable. (hint in the hint: introduce $f_K(m) =$ the smallest number $n$ such that $K(n) \geq m$).
Thanks to Sophie Laplante for learning me about that proof.

Hint for proof number two: introduce \( f(n) \) which is the largest computing time of halting programs without input, of length \( n \).

Recall that writing a number \( p \) requires about \( \ln p \) digits.

**Solutions to Previous Puzzles**

### 76 The 4 buoys

A geophysicist tells me he has dropped 4 buoys, floating on the sea, which are equidistant of each other. Should I call him a liar?

**Solution:** No. It is impossible that four different points on a plane are equidistant of each other, but it is possible in the 3-dimensional space: The 4 points are the vertices of a regular tetrahedron. Recall that earth is a SPHERE with two thirds of sea, so it should not be hard to find four points on the sea level drawing a regular tetrahedron (If you put the first boat next to the north pole, you have plenty of sea in the southern hemisphere to put the other three buoys).

### 77 Going far?

Your are on the road. At km 0 is the unique gas station. You have \( N \) cars (and drivers) and you can fill the \( N \) tanks at the gas station. It is forbidden to store gas elsewhere than in the tanks. It is allowed at any time to decant gas from one car tank to another. A full tank allows a car to drive for a constant distance \( K \). You have car number one (and that makes you the chief) and you are ready to abandon other cars on the way. How far can you go?

**Solution:** You can go about \( K \times \ln n \) kilometers. All cars get filled, they go \( K/n \) kilometers, then one car is abandoned, its remaining gas is decanted to other cars, so that the \( n-1 \) cars are filled. The \( n-1 \) cars go \( K/(n-1) \) kilometers, then one car is abandoned, and so on. You will manage to go \( K(1/n+1/(n-1)+\ldots+1/2+1/1) = K \times \sum_{i=0}^{n-1} 1/i \sim K \times \ln n \). It is optimal: Prove as a lemma that if \( S \) is a solution, and if \( S \) drops its first car before \( K/n \) (some gas will be wasted), or after \( K/n \) (at the first abandon, not all cars can be filled), then one can build a better solution where the first car is abandoned at \( K/n \). Then use the lemma and an induction to prove our solution is optimal.
Peter Ladislaw Hammer was born in Timisoara, Romania, on December 23, 1936. He earned his Ph.D. in mathematics under Academician Grigore C. Moisil at the University of Bucharest in 1966. He defected to Israel in 1967 where he became a professor at the Technion in Haifa. After moving to Canada, he taught at the University of Montréal (1969-72) and at the University of Waterloo (1972-83). In 1983, he moved to Rutgers University in the USA, where he founded RUTCOR - the Rutgers Center for Operations Research. He remained the director of RUTCOR until his untimely death in a tragic car accident, on December 27, 2006.

For more than 40 years, Peter Hammer has ranked among the most influential researchers in the fields of operations research and discrete mathematics. He has made numerous major contributions to these fields, launching several new research directions. His results have influenced hundreds of colleagues and have made a lasting impact on many areas of mathematics, computer science, and statistics.

Most of Peter Hammer’s scientific production has its roots in the work of George Boole on propositional logic. More than anyone else, Peter Hammer has used and extended Boole’s machina universalis to handle questions relating to decision making, analysis and synthesis as they arise in natural, economic and social sciences. Over the span of his scientific career, he has conducted eclectic forays into the interactions between Boolean methods, optimization, and combinatorial analysis, while adapting his investigations to the most recent advances of mathematical knowledge and of various fields of application. Among the main research topics which have received his attention, one finds an impressive array of methodological studies dealing with combinatorial optimization, some excursions into logistics and game theory, numerous contributions to graph theory, to the algorithmic aspects of propositional logic, to artificial intelligence and, more recently, to the development of innovative data mining techniques. His publications include
At the very onset of his career, as a researcher at the Institute of Mathematics of the Academia of Romania, Peter Hammer wrote several important articles on transportation problems, jointly with Egon Balas. At the same time his advisor, Grigore Moisil, directed him to the study of Boolean algebra. In a series of papers, Peter Hammer demonstrated that a large variety of relevant problems of operations research, combinatorics and computer science can be reduced to the optimization of a pseudo-Boolean function under constraints described by a system of pseudo-Boolean inequalities. A further main conceptual step in his work was the characterization of the set of feasible solutions of the above system as solutions of a single Boolean equation (or, equivalently, of a satisfiability problem). This led him, in joint work with Ivo Rosenberg and Sergiu Rudeanu, to the development of an original approach inspired from classical Boolean methods for the solution of a large variety of discrete optimization problems. This research project culminated in 1968 with the publication of the book *Boolean Methods in Operations Research and Related Areas* (Springer-Verlag, 1968), co-authored by Sergiu Rudeanu. This landmark monograph, which founded the field of pseudo-Boolean optimization, has influenced several generations of students and researchers, and is now considered a “classic” in operations research.

In a sense, Peter Hammer’s early work can be viewed as a forerunner of subsequent developments in the theory of computational complexity, since it was in effect demonstrating that a large class of combinatorial optimization problems is reducible to the solution of Boolean equations. However, this purely “reductionist” view of his work would be quite narrow. In fact, Peter Hammer has systematically used the “canonical” representation of various problems in terms of Boolean functions or Boolean equations to investigate the underlying structure, the “essence” of the problems themselves. More than often, this goal is met through a simplifying process based, once again, on the tools of Boolean algebra. This approach provides, for instance, a simple way to demonstrate that every system of linear inequalities in binary variables is equivalent to a set of inequalities involving only 0, 1, \(-1\) coefficients, as observed in a joint paper by Frieda Granot and Peter Hammer (1972). It also led Peter Hammer, Ellis Johnson and Uri Peled (1975) to early investigations into the facial structure of knapsack polyhedra.

In a related stream of research, Peter Hammer has established numerous fruitful links between graph theory and Boolean functions. In a famous joint paper with Vášek Chvátal on the aggregation of inequalities in integer programming (1977), he introduced and characterized the class of threshold graphs, inspired by threshold Boolean functions. Threshold graphs have subsequently been the subject of scores of articles and of a book by N.V.R. Mahadev and Uri Peled, two of Peter Hammer’s former doctoral students. Other links between graphs
and Boolean or pseudo-Boolean functions have been explored in joint work with Claude Benzaken, Dominique de Werra, Stephan Foldes, Toshihide Ibaraki, Alex Kelmans, Vadim Lozin, Frédéric Mafray, Bruno Simeone, etc.

Quadratic 0-1 optimization has been one of Peter Hammer’s main fields of investigation. The theory of roof-duality (1984), jointly developed with Pierre Hansen and Bruno Simeone, builds on concepts from linear programming (linear relaxations), Boolean theory (quadratic Boolean equations) and networks (maximum network flow problems) to compute best linear approximations of quadratic pseudo-Boolean functions and tight bounds on their maximum value. Further research along similar lines has been conducted by Peter Hammer in collaboration with Endre Boros, Jean-Marie Bourjolly, Yves Crama, David Rader, Gabriel Tavares, Xiaorong Sun, etc.

Peter Hammer has also shown interest for the application of Boolean models in artificial intelligence and related fields, as witnessed by numerous joint papers with Gabriela and Sorin Alexe, Martin Anthony, Tiberius Bonates, Endre Boros, Yves Crama, Oya Ekin, Toshihide Ibaraki, Alex Kogan, Miguel Lejeune, Irina Lozina, and other coworkers. His contributions bear on automatic theorem proving, compression of knowledge bases, algorithms for special classes of satisfiability problems, etc. About 20 years ago, he launched an innovative approach to data mining based on a blend of Boolean techniques and combinatorial optimization. The basic tenets of this approach were presented in a joint paper with Yves Crama and Toshihide Ibaraki (1988) and were subsequently developed by Peter Hammer and his coworkers into a new broad area of research, which he dubbed Logical Analysis of Data, or LAD for short. The effectiveness of the LAD methodology has been validated by many successful applications to data analysis problems. In particular, some front-of-the-line medical centers are increasingly using LAD in the practice of medical diagnosis for a variety of syndromes.

Many aspects of Peter Hammer’s contribution to the study of Boolean functions and their combinatorial structure are to be found in a forthcoming monograph entitled Boolean Functions: Theory, Algorithms, and Applications, co-authored by Yves Crama and several other close collaborators of Peter Hammer, to be published by Cambridge University Press in 2007.

Beside his scientific production, Peter Hammer will undoubtedly be remembered for his vigorous contribution to and promotion of discrete mathematics and operations research. He was the founder and editor-in-chief of several highly-rated professional journals, including Discrete Mathematics, Discrete Applied Mathematics, Discrete Optimization, Annals of Discrete Mathematics, Annals of Operations Research and the SIAM Monographs on Discrete Mathematics and Applications. At Rutgers University, Peter Hammer was the founding Director of the operations research programme, and he was largely responsible for developing RUTCOR into an internationally recognized center of excellence and an open
institute, where seminars, workshops, graduate courses, and a constant flow of visitors create a stimulating research environment. He was also a tireless organizer of professional conferences and workshops, where he always made sure to provide opportunities for interactions between experienced scientists and younger researchers.

The importance of Peter Hammer’s scientific contribution was acknowledged by the award of numerous international distinctions, including the “George Tzitzeica” prize of the Romanian Academy of Science (1966), the Euler Medal of the Institute of Combinatorics and its Applications (1999), and honorary degrees from the Swiss Federal Institute of Technology in Lausanne (1986), the University of Rome “La Sapienza” (1998), and the University of Liège (1999). He was a Fellow of the American Association for the Advancement of Science since 1974, and a Founding Fellow of the Institute of Combinatorics and its Applications. Several conferences were organized in his honor, including the First International Colloquium on Pseudo-Boolean Optimization (Chexbres, Switzerland, 1987), the Workshop and Symposia Honoring Peter L. Hammer (Caesarea Rothchild Institute, University of Haifa, 2003), and the International Conference on Graphs and Optimization (GO V, Leukerbad, Switzerland, 2006).

Peter Hammer was not only an outstanding scholar and a tireless organizer, but also a kind, generous and humorous human being. He relished the interaction with students and colleagues, and made everybody feel comfortable to work with him, be it on a mathematical question (which he was always keen to formulate) or on planning a conference. He supervised numerous graduate students with respect and fatherly understanding, considering each one of them as his “best student”. He was also a true “citizen of the world”: born in Romania from a Hungarian family, he subsequently took the Canadian citizenship, then the US one, wrote joint papers with co-authors of 28 different countries, fluently spoke 6 languages (or more), traveled the world extensively, spent extended periods of time in Belgium, France, Israel, Italy, Russia, Switzerland and many other countries, and developed an extended network of friends and coworkers on all continents.

Finally, last but certainly not least, Peter Hammer was a loving husband, father and grandfather. He is survived by his wife, Anca Ivanescu, whom he married in 1961 and whose family name he assumed for a few years, by his two sons Alexander and Maxim, and by four beloved grandchildren, Isabelle, Madeline, Annelise, and Oliver.

He will be missed by everyone who knew him, always and forever.

Endre Boros\(^1\), Yves Crama\(^2\) and Bruno Simeone\(^3\)

---

\(^1\)RUTCOR, Rutgers Center for Operations Research, Piscataway, NJ 08854-8003, USA
\(^2\)HEC Management School, University of Liège, 4000 Liège, Belgium
\(^3\)Department of Statistics, University of Rome La Sapienza, 00185 Rome, Italy
Speculations on Biology, Information and Complexity

G. J. Chaitin, IBM Research, New York

Abstract

It would be nice to have a mathematical understanding of basic biological concepts and to be able to prove that life must evolve in very general circumstances. At present we are far from being able to do this. But I’ll discuss some partial steps in this direction plus what I regard as a possible future line of attack.

Can Darwinian evolution be made into a mathematical theory? Is there a fundamental mathematical theory for biology?

To me this does not seem so marvelous, since mathematics and physics co-evolved. That however does not diminish the miracle that at a fundamental level Nature is ruled by simple, beautiful mathematical laws, that is, the miracle that Nature is comprehensible.

I personally am much more disturbed by another phenomenon, pointed out by I.M. Gel’fand and propagated by Vladimir Arnold in a lecture of his that is available on the web, which is the stunning contrast between the relevance of mathematics to physics, and its amazing lack of relevance to biology!

Indeed, unlike physics, biology is not ruled by simple laws. There is no equation for your spouse, or for a human society or a natural ecology. Biology is the
domain of the complex. It takes \(3 \times 10^9\) bases = \(6 \times 10^9\) bits of information to specify the DNA that determines a human being.

Darwinian evolution has acquired the status of a dogma, but to me as a mathematician seems woefully vague and unsatisfactory. What is evolution? What is evolving? How can we measure that? And can we prove, mathematically prove, that with high probability life must arise and evolve?

In my opinion, if Darwin’s theory is as simple, fundamental and basic as its adherents believe, then there ought to be an equally fundamental mathematical theory about this, that expresses these ideas with the generality, precision and degree of abstractness that we are accustomed to demand in pure mathematics.

Look around you. We are surrounded by evolving organisms, they’re everywhere, and their ubiquity is a challenge to the mathematical way of thinking. Evolution is not just a story for children fascinated by dinosaurs. In my own lifetime I have seen the ease with which microbes evolve immunity to antibiotics. We may well live in a future in which people will again die of simple infections that we were once briefly able to control.

Evolution seems to work remarkably well all around us, but not as a mathematical theory!

In the next section of this paper I will speculate about possible directions for modeling evolution mathematically. I do not know how to solve this difficult problem; new ideas are needed. But later in the paper I will have the pleasure of describing a minor triumph. The program-size complexity viewpoint that I will now describe to you does have some successes to its credit, even though they only take us an infinitesimal distance in the direction we must travel to fully understand evolution.

**A software view of biology: Can we model evolution via evolving software?**

I’d like to start by explaining my overall point of view. It is summarized here:

\[
\text{Life} = \text{Software} \quad ?
\]

- **program** \(\rightarrow\) **COMPUTER** \(\rightarrow\) **output**
- **DNA** \(\rightarrow\) **DEVELOPMENT/PREGNANCY** \(\rightarrow\) **organism**

\[
\text{Size of program in bits} \approx (\text{Amount of DNA in bases}) \times 2
\]

So the idea is firstly that I regard life as software, biochemical software. In particular, I focus on the digital information contained in DNA. In my opinion,
DNA is essentially a programming language for building an organism and then running that organism.

More precisely, my central metaphor is that DNA is a computer program, and its output is the organism. And how can we measure the complexity of an organism? How can we measure the amount of information that is contained in DNA? Well, each of the successive bases in a DNA strand is just 2 bits of digital software, since there are four possible bases. The alphabet for computer software is 0 and 1. The alphabet of life is A, G, C, and T, standing for adenine, cytosine, guanine, and thymine. A program is just a string of bits, and the human genome is just a string of bases. So in both cases we are looking at digital information.

My basic approach is to measure the complexity of a digital object by the size in bits of the smallest program for calculating it. I think this is more or less analogous to measuring the complexity of a biological organism by 2 times the number of bases in its DNA.

Of course, this is a tremendous oversimplification. But I am only searching for a toy model of biology that is simple enough that I can prove some theorems, not for a detailed theory describing the actual biological organisms that we have here on earth. I am searching for the Platonic essence of biology; I am only interested in the actual creatures we know and love to the extent that they are clues for finding ideal Platonic forms of life.

How to go about doing this, I am not sure. But I have some suggestions.

It might be interesting, I think, to attempt to discover a toy model for evolution consisting of evolving, competing, interacting programs. Each organism would consist of a single program, and we would measure its complexity in bits of software. The only problem is how to make the programs interact! This kind of model has no geometry, it leaves out the physical universe in which the organisms live. In fact, it omits bodies and retains only their DNA. This hopefully helps to make the mathematics more tractable. But at present this model has no interaction between organisms, no notion of time, no dynamics, and no reason for things to evolve. The question is how to add that to the model.

Hopeless, you may say. Perhaps not! Let’s consider some other models that people have proposed. In von Neumann’s original model creatures are embedded in a cellular automata world and are largely immobile. Not so good! There is also the problem of dissecting out the individual organisms that are embedded in a toy universe, which must be done before their individual complexities can be measured. My suggestion in one of my early papers that it might be possible to use the concept of mutual information—the extent to which the complexity of two things taken together is smaller than the sum of their individual complexities—in order to accomplish this, is not, in my current opinion, particularly fruitful.

In von Neumann’s original model we have the complete physics for a toy cellular automata universe. Walter Fontana’s ALChem = algorithmic chemistry
project went to a slightly higher level of abstraction. It used LISP S-expressions to model biochemistry. LISP is a functional programming language in which everything—programs as well as data—is kept in identical symbolic form, namely as what are called LISP S-expressions. Such programs can easily operate on each other and produce other programs, much in the way that molecules can react and produce other molecules.

I have a feeling that both von Neumann’s cellular automata world and Fontana’s algorithmic chemistry are too low-level to model biological evolution. So instead I am proposing a model in which individual creatures are programs. As I said, the only problem is how to model the ecology in which these creatures compete. In other words, the problem is how to insert a dynamics into this static software world.

Since I have not been able to come up with a suitable dynamics for the software model I am proposing, I must leave this as a challenge for the future and proceed to describe a few biologically relevant things that I can do by measuring the size of computer programs. Let me tell you what this viewpoint can buy us that is a tiny bit biologically relevant.

**Pure mathematics has infinite complexity and is therefore like biology**

Okay, program-size complexity can’t help us very much with biological complexity and evolution, at least not yet. It’s not much help in biology. But this viewpoint has been developed into a mathematical theory of complexity that I find beautiful and compelling—since I’m one of the people who created it—and that has important applications in another major field, namely metamathematics. I call my theory *algorithmic information theory*, and in it you measure the complexity of something $X$ via the size in bits of the smallest program for calculating $X$, while completely ignoring the amount of effort which may be necessary to discover this program or to actually run it (time and storage space). In fact, we pay a severe price for ignoring the time a program takes to run and concentrating only on its size. We get a beautiful theory, but we can almost never be sure that we have found

---

1A model with perhaps the opposite problem of being at too high a level, is Douglas Lenat’s AM = Automated Mathematician project, which dealt with the evolution of new mathematical concepts.

2Thomas Ray’s *Tierra* project did in fact create an ecology with software parasites and hyper-parasites. The software creatures he considered were sequences of machine language instructions coexisting in the memory of a single computer and competing for that machine’s memory and execution time. Again, I feel this model was too low-level. I feel that too much micro-structure was included.
The smallest program for calculating something. We can almost never determine the complexity of anything, if we chose to measure that in terms of the size of the smallest program for calculating it!

This amazing fact, a modern example of the incompleteness phenomenon first discovered by Kurt Gödel in 1931, severely limits the practical utility of the concept of program-size complexity. However, from a philosophical point of view, this paradoxical limitation on what we can know is precisely the most interesting thing about algorithmic information theory, because that has profound epistemological implications.

The jewel in the crown of algorithmic information theory is the halting probability $\Omega$, which provides a concentrated version of Alan Turing’s 1936 halting problem. In 1936 Turing asked if there was a way to determine whether or not individual self-contained computer programs will eventually stop. And his answer, surprisingly enough, is that this cannot be done. Perhaps it can be done in individual cases, but Turing showed that there could be no general-purpose algorithm for doing this, one that would work for all possible programs.

The halting probability $\Omega$ is defined to be the probability that a program that is chosen at random, that is, one that is generated by coin tossing, will eventually halt. If no program ever halted, the value of $\Omega$ would be zero. If all programs were to halt, the value of $\Omega$ would be one. And since in actual fact some programs halt and some fail to halt, the value of $\Omega$ is greater than zero and less than one. Moreover, $\Omega$ has the remarkable property that its numerical value is maximally unknowable. More precisely, let’s imagine writing the value of $\Omega$ out in binary, in base-two notation. That would consist of a binary point followed by an infinite stream of bits. It turns out that these bits are irreducible, both computationally and logically:

- You need an $N$-bit program in order to be able to calculate the first $N$ bits of the numerical value of $\Omega$.
- You need $N$ bits of axioms in order to be able to prove what are the first $N$ bits of $\Omega$.
- In fact, you need $N$ bits of axioms in order to be able to determine the positions and values of any $N$ bits of $\Omega$, not just the first $N$ bits.

Thus the bits of $\Omega$ are, in a sense, mathematical facts that are true for no reason, more precisely, for no reason simpler than themselves. Essentially the only way to determine the values of some of these bits is to directly add that information as a new axiom.

And the only way to calculate individual bits of $\Omega$ is to separately add each bit you want to your program. The more bits you want, the larger your program
must become, so the program doesn’t really help you very much. You see, you can only calculate bits of Ω if you already know what these bits are, which is not terribly useful. Whereas with π = 3.1415926… we can get all the bits or all the digits from a single finite program, that’s all you have to know. The algorithm for π compresses an infinite amount of information into a finite package. But with Ω there can be no compression, none at all, because there is absolutely no structure.

Furthermore, since the bits of Ω in their totality are infinitely complex, we see that pure mathematics contains infinite complexity. Each of the bits of Ω is, so to speak, a complete surprise, an individual atom of mathematical creativity. Pure mathematics is therefore, fundamentally, much more similar to biology, the domain of the complex, than it is to physics, where there is still hope of someday finding a theory of everything, a complete set of equations for the universe that might even fit on a T-shirt.

In my opinion, establishing this surprising fact has been the most important achievement of algorithmic information theory, even though it is actually a rather weak link between pure mathematics and biology. But I think it’s an actual link, perhaps the first.

**Computing Ω in the limit from below as a model for evolution**

I should also point out that Ω provides an extremely abstract—much too abstract to be satisfying—model for evolution. Because even though Ω contains infinite complexity, it can be obtained in the limit of infinite time via a computational process. Since this extremely lengthy computational process generates something of infinite complexity, it may be regarded as an evolutionary process.

How can we do this? Well, it’s actually quite simple. Even though, as I have said, Ω is maximally unknowable, there is a simple but very time-consuming way to obtain increasingly accurate lower bounds on Ω. To do this simply pick a cut-off t, and consider the finite set of all programs p up to t bits in size which halt within time t. Each such program p contributes 1/2^p, 1 over 2 raised to p’s size in bits, to Ω. In other words,

$$\Omega = \lim_{t \to \infty} \left( \sum_{|p| \leq t \text{ & halts within time } t} 2^{-|p|} \right).$$

This may be cute, and I feel compelled to tell you about it, but I certainly do not regard this as a satisfactory model for biological evolution, since there is no apparent connection with Darwin’s theory.
The classical work on a theoretical mathematical underpinning for biology is von Neumann’s posthumous book [2]. Interestingly enough, Francis Crick—who probably contributed more than any other individual to creating modern molecular biology—for many years shared an office with Sydney Brenner, who was aware of von Neumann’s thoughts on theoretical biology and self-reproduction. This interesting fact is revealed in the splendid biography of Crick [3].

For a book-length presentation of my own work on information and complexity, see [4], where there is a substantial amount of material on molecular biology. This book is summarized in my recent article [5], which however does not discuss biology. A longer overview of [4] is my Alan Turing lecture [6], which does touch on biological questions.

For my complete train of thought on biology extending over nearly four decades, see also [7,8,9,10,11].

For information on Tierra, see Tom Ray’s home page at http://www.his.atr.jp/~ray/. For information on ALChemy, see http://www.santafe.edu/~walter/ALChemy/papers.html. For information on Douglas Lenat’s Automated Mathematician, see [12] and the Wikipedia entry http://en.wikipedia.org/wiki/Automated_Mathematician. For Vladimir Arnold’s provocative lecture, the one where Wigner and Gel’fand are mentioned, see http://pauli.uni-muenster.de/~munsteg/arnold.html. Wigner’s entire paper is itself on the web at http://www.dartmouth.edu/~matc/MathDrama/reading/Wigner.html.


---

3An earlier account of von Neumann’s thinking on this subject was published in [1], which I read as a child.
REPORTS FROM CONFERENCES
**Report on CS&P 2006**

**Workshop on Concurrency, Specification and Programming**

**27–29 September 2006, Wandlitz, Germany**

Manfred Kudlek

CS&P 2006 was the 14th in this series of workshops on **Concurrency, Specification, and Programming**, founded by Peter Starke and Ludwik Czaja in Berlin in 1992. It took place at **Wandlitz**, about 30 km north of the centre of Berlin, from September 27-29, 2006. The conference site was **Waldhotel Wandlitz**, which was also where most of the participants stayed and is very near the former residence area WALDHEIDLING of the DDR ruling class, which is protected as a **wild animal research area**.

It was organized by **Humboldt Universität zu Berlin**, **Uniwersytet Warszawski**, and **Wyszà Szkolà Informatyki i Zarządzania w Rzeszowie** (WSIZ, The University of Information Technology and Management in Rzeszów). The organizing and program committee included Hans-Dieter Burkhard, Ludwik Czaja, Gabriela Lindemann, Wojciech Penczek, Andrzej Salwicki, Andrzej Skowron, Holger Schlingloff, and Zbigniew Suraj. CS&P 2006 was supported by **Deutscher Akademischer Austausch-Dienst** and **Humboldt-Universität zu Berlin**.

It was attended by 46 participants from 6 countries, mostly from Poland and Germany. The following table gives the details:

<table>
<thead>
<tr>
<th></th>
<th>PL</th>
<th>DE</th>
<th>NL</th>
<th>RU</th>
<th>SK</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>22</td>
<td>17</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The scientific program consisted of 1 invited lecture and 43 contributions, held in 2 parallel sessions, one (A) consisting of 19, the other (B) of 20 presentations, and 2 plenary sessions. The following tables give the statistics about distribution by number of authors and countries (C for country):

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18</td>
<td>12</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>C</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>1</td>
<td>PL 18 5</td>
</tr>
<tr>
<td>DE</td>
<td>13</td>
<td>RU 6 5</td>
</tr>
<tr>
<td>NL</td>
<td>5</td>
<td>SE 1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SK 1</td>
</tr>
<tr>
<td>UA 2</td>
</tr>
<tr>
<td>UK 2</td>
</tr>
</tbody>
</table>

*Bulletin of the EATCS no 91, pp. 241–242, February 2007*

© European Association for Theoretical Computer Science
CS&P 2006 covered the fields:

**Section A**
- Workflow Nets and an Application
- Logics
- Object nets
- Programming Languages
- Similarity and Nearness

**Section B**
- Mathematical Structures
- (Rough Sets and Categories)
- Mathematical Structures
- (Algebra and Graphs)
- Verification
- Model-based Design
- Multi-Agent systems and Applications
- Robotics and Simulation

The workshop was opened on Wednesday morning with a talk on the history of CS&P, by Hans-Dieter Burkhard and Ludwik Czaia. This was followed by the excellent guest lecture by Antoni Mazurkiewicz on *Compositional Systems over Reducible Networks*, celebrating the 50th anniversary of scientific research in the field of combining graphs, algebra, and algorithms. Worth a mention were the good presentations by the same author on *Local Properties of Triangular Graphs*, by Ludwik Czaia on *Interpreted Nets*, by Andrzej Skowron on *Nearness in Approximation Spaces*, dedicated to the passing away of Zdzislaw Pawlak on April 7, 2006, and on *Analysis of Conflict Dynamics by Risk Patterns*.

The workshop was brought to a close on Thursday late afternoon by Bernd-Holger Schlingloff, who thanked all organizers, program committee members, authors, speakers, and participants, and by Ludwik Czaia, who presented Gabriela Lindemann, Trzebiatowski a big bunch of flowers for her excellent organization.

The proceedings, edited by Hans-Dieter Burkhard, Ludwik Czaia, Gabriela Lindemann, Woiwech Penczek, Andrzej Salwicki, Andrzej Skowron, and Bernd-Holger Schlingloff, containing all 44 contributions, were published in 3 + ε volumes as INFORMATIK-BERICHTE 206 of HUMBOLDT-UNIVERSITÄT ZU BERLIN: Volume 1 – Concurrency; Volume 2 – Specification; Volume 3 – Programming; and ε – three articles missing at the end of Volume 3.

On Friday we had an excursion, first visiting the hunting castle in Gross-Schönebeek 20 km north of Wandlitz, formerly used by the Prussian kings and German emperors, as well as by the DDR high society, the last time on November 9, 1989, when Erich Honecker shot his last deer. After that we had a one hour walk to the wild animal park at Schorfheide, where we had lunch and could view wild animals, among them wolves, horses, boars, and wisents (European bisons). The weather was warm and friendly, with highest temperatures around 25°C.

CS&P was successful again, of high level and well organized. The program can be found at [http://www2.informatik.hu-berlin.de/ki/CSP2006](http://www2.informatik.hu-berlin.de/ki/CSP2006).
REPORT ON NWPT 2006
Nordic Workshop on Programming Theory 2006
18–20 October 2006, Reykjavík, Iceland

Luca Aceto and Anna Ingolfsdottir

The 18th Nordic Workshop on Programming Theory took place from Wednesday, 18 October, till Friday, 20 October, at Reykjavík University. The workshop was held under the auspices of the Icelandic Centre of Excellence in Theoretical Computer Science (ICE-TCS) and of the IFIP TC1 Working Group 1.8 on Concurrency Theory, and was partly sponsored by Reykjavík University.

The NWPT series of annual workshops is a forum bringing together programming theorists from the Nordic and Baltic countries (but also elsewhere). The previous workshops were held in Uppsala (1989, 1999 and 2004), Aalborg (1990), Gothenburg (1991 and 1995), Bergen (1992 and 2000), Turku (1993, 1998, and 2003), Aarhus (1994), Oslo (1996), Tallinn (1997 and 2002), Lyngby (2001), and Copenhagen (2005). Thus, this was the first ever NWPT workshop held in Iceland. The event was attended by 45 scientists from the Nordic countries, but participants came from as far as south as Italy. In addition, several local MSc and advanced BSc students enjoyed some of the presentations, which were of consistently high quality. (Here are a couple of quick observations on the geographical distribution of the participants.

1. There were no contributed talks from Sweden, and David Sands was therefore the only representative of Swedish computer science at the workshop.
2. There was a good number of participants from Germany.)

The scientific programme consisted of four invited presentations and 26 contributed ones. The four invited talks were:

- Gerd Behrmann, Aalborg University. How to become a successful tool builder: The dirty tricks;
- Matthew Hennessy, University of Sussex. Some remarks on testing probabilistic processes;
- Hanne Riis Nielson, Technical University of Denmark. Analysis of process interactions;
- David Sands, Chalmers University of Technology and University of Göteborg. Specifying and verifying dynamic information flow properties.
Hanne Riis Nielson gave the workshop the best of starts by offering an excellent overview of her work on using static analysis techniques to validate models of computational scenarios vis-à-vis the actual “reality” that is being modelled. As a motivating question she asked:

“How can we be sure that process calculi descriptions of, say, biological processes/pathways are faithful to reality?”

At the end of her clear and well-paced lecture, the audience was left feeling that static analysis can indeed help in addressing the all important motivating question underlying her presentation.

Gerd Behrmann’s talk offered a thought provoking and stimulating analysis of the role of tool development in algorithmic verification. Gerd gave the audience many good reasons for building tools, but also pointed out that empirical studies in this field are often of questionable quality, with results that are not reproducible and unfounded conclusions. He pledged for this situation to be improved if tool building is to become a respected scientific activity.

Not surprisingly, Gerd’s talk gave rise to a lively discussion (both during and after the presentation). Flemming Nielson made some very interesting remarks after the talk, explaining how a tool developer can obtain credit for his/her work at his institution (Technical University of Denmark), e.g., by means of innovation schemes and the writing of books about the lessons learned during tool building. He also defended, in case there was any need to do so, the hard work of theorists, and uttered the following eminently quotable sentence:

“The purpose of theory is insight, not theorems!”

Matthew Hennessy delivered a talk on testing probabilistic processes that presented work he did while visiting NICTA in Australia. This was a one hour version of a shorter talk he had delivered earlier at the symposium in honour of Gordon Plotkin’s sixtieth birthday. Matthew gave a typically well polished lecture, which managed to present a lot of technical work without ever giving his audience the feeling of being overwhelmed by the mathematics. We are looking forward to reading the paper on which the talk was based.

The last invited talk for the workshop was delivered by David Sands. Static verification of secure information flow has been a popular theme in recent programming language research, but the information flow policies considered are based on a static view of security levels. In his talk, Dave provided a road map of the main directions of current research, and introduced a simple mechanism, called flow locks, for specifying dynamic information flow policies, and a type-and-effect system for statically verifying flow lock policies. The talk was based on joint work with Niklas Broberg.
The Bulletin of the EATCS

The 26 contributed talks were mostly of excellent quality, presenting work at different stages of development; some of it was definitely in progress, some other was “in infancy”, and some was instead very mature. There was a fair amount of discussion during and after most presentations. This is what a workshop should be like!

We refer interested readers to the URL

http://www.ru.is/nwpt06/programme.html

for the full programme for the workshop, with links to the abstracts for the contributed talks and to the slides for some of the presentations.

Some photos from the workshop, courtesy of MohammadReza Mousavi, are available at the URL

http://picasaweb.google.com/smrmousavi/NWPT06.

The 2007 edition of the workshop will be held in Oslo, Norway. We trust that it’ll be just as successful as we felt this one was. Good luck to Olaf Owe, who will take over the mantle of organizer from us.
Pictures from DCM 2006 (by M. Kudlek)

Maribel Fernández  Kohei Honda  Rajagopal Nagarajan

Luigi Liquori  Nikolaos Siafakis  Luca Fossati

Mircea-Dan Hernest  Alexander Summers  Robert K. Meyer
Pictures from MeCBIC 2006 (by M. Kudlek)
BEATCS no 91

REPORTS FROM CONFERENCES

Verena Wolf
Camilo Rueda
Antonio Vitale

Allessandro Romanel
Rudolf Freund
Gabriel Ciobanu

Xian Xu
Giancarlo Mauri, Nadia Busi, Claudio Zandron

248
Pictures from ICALP 2006  (by M. Kudlek)

Noga Alon  Oleg Verbitsky  Amin Coja-Oghlan

Douglas E. Carroll  Vladimiro Sassone  Ben Reichardt

Magnus Halldórsson  Ronald de Wolf  Harold Ollivier
BEATCS no 91
REPORTS FROM CONFERENCES

Michele Bugliesi, opening ICALP

Chia-Jung Lee

Jaikumar Radhakrishnan

Magnus Bordewich

Birgit Pfitzmann

Svetlana Golonetsky

Tom Früwirth (PPDP)

Bart Preneel

Cynthia Dwork
The Bulletin of the EATCS

Michal Kunc  Turlough Neary  Christos Kapoutsis

Osamu Watanabe  Mathias Samuelides  Yevgeniy Dodis

Tomoyuki Yamakami  Danny Harnik, presenting best C-paper  Sébastien Zimmer
REPORTS FROM CONFERENCES

Ricardo Corin  Michael Maher  Simon Peyton Jones

Wiesław Zielonka  Mihalis Yannakakis  Eryk Kopczyński

Colin Stirling  Prakash Panangaden  Kohei Honda
The Bulletin of the EATCS

Paul Blain Levy  John Hitchcock  Aranyak Mehta

Stefano Guerrini  Esfandiar Haghverdi  Filip Murlak, presenting best B-paper

Pierre-Alain Reynier  Tomasz Jurdziński  Rasmus Ejlers Mögelberg
BEATCS no 91

REPORTS FROM CONFERENCES

Corin Pitcher  Juhani Karhumäki  Taolue Chen

Bas Luttik  Markus Lohrey  Wong Karioa

Lenore Zuck  Aniello Murano  Michele Bugliesi
The Bulletin of the EATCS

Michael Benedikt
Blaise Genest
Radu Iosif

Qiqi Yan, presenting best student paper
Mogens Nielsen, opening award session
Pierre-Louis Curien

Manindra Agrawal, receiving Gödel prize
Mariangiola Dezani-Ciancaglini
Mike Paterson, receiving EATCS award
Michael Stewart Paterson  Giorgio Ausiello  Ingo Wegener

Bart Preneel  Qiqi Yan, receiving best student paper award  Paul Spirakis

San Servol, corridor  Mogens Nielsen, receiving present  A distinguished speaker

256
ABSTRACTS OF
PhD THESES
Abstract of PhD Thesis

Author: Peter Leupold
Title: Languages Generated by Iterated Idempotencies
Language: English
Supervisor: Juhani Karhumäki
Victor Mitrana
Institute: Universitat Rovira i Virgili, Tarragona, Spain
Date: 22 November 2006

Abstract

Much of the current work in Formal Language Theory has been inspired by mechanisms observed in molecular biology. Most prominently, the computational power of recombinations occurring in DNA is investigated, when applying these operations on general strings. Also this thesis has its origin in such a DNA operation, namely in duplication, i.e. the replacement of a factor $u$ by $uu$ within a word.

Chapter 2 outlines the original motivation for introducing the formal language operation of duplication in context with other DNA-inspired string operations. Then its generalization to idempotency languages is described; here we replace the rewriting rule $u \rightarrow uu$ from duplication by $u^m \rightarrow u^n$ for arbitrary but fixed natural numbers $m$ and $n$. A few spotlights are shed on the history of idempotencies in the parts of Algebra related to formal languages, most mentionable on the famous Burnside problem and the problem of non-counting classes. After this, the actual investigations on idempotency languages are presented.

Starting out from a few results on special cases treated in earlier work of other authors, we mainly focus on two types of questions. For one thing we try and determine, which rewrite relations are confluent. Secondly, we examine whether the languages generated by them are regular.

First off, we treat the most restricted variant, uniformly bounded idempotencies. Here all rewrite rules must have the same length. This makes the problems quite resolvable, and the conditions for confluence and regularity are fully characterized for all possible combinations of parameters.

Already for the following variant, bounded idempotencies, where only an upper bound is imposed on the rules’ length, more cases are left open; while all the generated languages are still context-free, many are not regular. Finally, for unrestricted idempotency relations we present on the one hand results that carry over
from the restricted cases, on the other hand most cases are solved over a restricted alphabet of only two letters. Interesting questions like the context-freeness of general duplication languages remain open.

Contrary to the chronological development we then come from general idempotency languages to duplication languages in Chapter 3. Some results are presented, which have not been generalized to general idempotencies and which seem especially interesting in the context of the original motivation for duplication from DNA computing, for example decision problems.

Section 3.2 then introduces the concept of idempotency root. This is motivated by recalling the primitive root of words, then some results concerning duplication roots. The main interest is on the finiteness of roots and the decidability of this property.

In Section 3.3 we define a type of code, which is robust under uniformly bounded duplications in the sense that such duplications occurring in the code words do not affect the uniqueness of factorization. Among other things the conditions are characterized, under which infinite such codes exist, and the density of languages generated by these codes is investigated.

Finally, in Section 3.4 we examine the closure of the classes of regular and context-free languages under duplication in its differently length-bounded variants. Mainly bounded duplication is treated, for example the closures of regular and context-free languages under this operation is established.
The Bulletin of the EATCS

2.4 Uniformly Bounded Idempotency .................................. 38
2.5 Bounded Idempotency .............................................. 44
2.6 General Idempotency ................................................ 51

3 Duplication ................................................................. 63
3.1 General Duplication .................................................. 63
3.2 Roots ........................................................................ 66
3.3 Duplication Codes ....................................................... 77
3.4 Closure of Language Classes ......................................... 89

Concluding Thoughts ....................................................... 97
Interesting Problems Left Open ........................................... 99

Author’s correspondence address
Peter Leupold
Grup de Recerca de Linguística Matemàtica
Plaça Imperial Tàrraco 1
43000 Tarragona
Catalunya
Spain
Abstract

The thesis scrutinizes the power of restricted tree series transducers. A tree series is a mapping from the set of trees into some semiring. We compare the classes of tree series transformations computed by several classes of tree series transducers that use either pure or o-substitution. The analysis is very detailed (in the sense that we consider several syntactic restrictions on tree series transducers and several classes of weight semirings) for the classes of transformations computed by deterministic tree series transducers. We also consider classes of transformations computed by polynomial tree series transducers albeit in less detail. Finally, we also investigate compositions of transformations computed by bottom-up and top-down tree series transducers.

Tree series transducers were introduced as a joint generalization of tree transducers and weighted tree automata. They thereby serve as the transducing devices corresponding to weighted tree automata. Both historical predecessors of tree series transducers have successfully been motivated from and applied in practice. Recently, particular tree series transducers (called extended weighted top-down tree transducers) are successfully applied in machine translation.

In Chapter 3 we introduce pure and o-substitution formally and study them with respect to the fundamental properties of distributivity, linearity, and associativity. Finally, we consider preservation of recognizability for both substitutions. The main result of this chapter, apart from the lemmata on distributivity, linearity, and associativity, states that o-substitution preserves recognizable tree series in the tropical semiring \( (\mathbb{N} \cup \{\infty\}, \min, +) \), whenever the tree series are linear. In general, the result holds for all semirings that are continuous and additively idempotent.

In Chapter 4 we introduce tree series transducers. Intuitively, a tree series transducer is a tree transducer in which the transitions carry a weight; a weight is an element of some semiring. The rewrite semantics works as follows. Along a successful computation on some input tree, the weights of the involved transitions...
are combined by means of the semiring multiplication; if there is more than one successful computation for some pair of input and output trees, then the weights of these computations are combined by means of the semiring addition.

In the same way as tree transducers, also tree series transducers can have particular properties. For example, tree series transducers can be deterministic, total, linear, nondeleting, or a homomorphism. We investigate the effects of these restrictions on the transformational power in detail in the following chapters.

We start our investigation of the power of tree series transducers with deterministic devices. For such devices there is at most one successful computation for every input tree. Deterministic and total top-down tree transducers formalize a restricted class of functional programs. Consequently, they and their generalizations were intensively studied, in particular in the area of functional programming and syntax-directed semantics. These applications could potentially benefit from the additional information (i.e., the weight) that is attached to the output tree of a deterministic tree series transducer. In the functional programming application, we could count reduction steps using the weights, and this could enable us to study efficiency effects of constructions in a uniform setting. Moreover, we can perform a multitude of statistical computations while processing the input tree. Such features are relevant in natural language and speech processing.

We compare the transformational power of deterministic bottom-up and top-down tree series transducers in Chapter 5. We also study the effect of the two types of substitution. The investigation is detailed in the sense that we consider all classes of transformations computed by restricted deterministic tree series transducers where the restriction is any combination of nondeletion, linearity, totality, and homomorphism. Our results are presented in Hasse-diagrams that convey the full relation of all the displayed classes. Such diagrams are presented for “most” commutative semirings.

In Chapter 6 we consider polynomial tree series transducers. Since equality and inclusion results already exist in sizable number, we concentrate on incomparability results. Nondeterministic tree series transducers are interesting because they can capture several (combinatorial) possibilities that yield the same final output tree. For example, in natural language processing the parse tree of an input sentence is usually not uniquely determined because natural language is ambiguous. Thus the output consists of several trees each annotated with a likelihood.

Our main result in this chapter concerns additively idempotent and weakly growing semirings. We show that the classes of transformations computed by bottom-up tree series transducers that use pure substitution and o-substitution are incomparable. The classes of transformations computed by bottom-up and top-down tree series transducers that use pure substitution are also incomparable.

In the final chapter we consider compositions of transformations. They arise naturally because the strategy of breaking down a transformation into several
stages is well-known and well-established in software development. The implementation of a single stage is often easier to understand and easier to validate. The final result of the transformation is then obtained by running the stages one after the other where the output of one stage becomes the input of the next stage. This strategy introduces an overhead of communication between the stages, which might deteriorate efficiency. In our setting, the specification of a stage is a tree series transducer with some properties (e.g., deterministic or linear).

In Chapter 7 we present several composition constructions for tree series transducers. In particular, we show that the class of bottom-up tree series transformations is closed under left-composition with linear bottom-up tree series transformations. Moreover, we show that the class of bottom-up tree series transformations is actually closed under right-composition with boolean, deterministic bottom-up tree series transformations. For top-down tree series transducers we obtain the following results. We show that the class of top-down tree series transformations is closed under right-composition with nondeleting, linear top-down tree series transformations. Secondly, we show that the composition of a boolean, deterministic, total top-down tree series transformation with a linear top-down tree series transformation is a top-down tree series transformation.

Table of Contents

1 Introduction ...................................................... 1
2 Preliminaries ..................................................... 15
3 Tree Series Substitution ......................................... 29
4 Tree Series Transducers .......................................... 57
5 Deterministic Tree Series Transducers ......................... 77
6 Polynomial Tree Series Transducers ........................... 115
7 Compositions of Tree Series Transducers ...................... 151

Author’s correspondence address  Andreas Maletti
Technische Universität Dresden
Fakultät Informatik
Theoretical Computer Science
D-01062 - Dresden
Germany
European Association for Theoretical Computer Science
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.
The Bulletin of the EATCS

Sites of ICALP meetings:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, Great Britain 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Náfplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, Great Britain 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- 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;

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:
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

Università 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